Evolutionary Adaptation in Natural and Artificial Systems

Jean-Baptiste Mouret

Université Pierre et Marie Curie
Institut des Systèmes Intelligents et de Robotique

Evolutionary Adaptation in Natural and Artificial Systems

Habilitation à Diriger des Recherches
spécialité " Informatique"

by Jean-Baptiste Mouret

HDR presented and defended June 16, 2015
Jury:

Dr.	François Charpillet	Research Director at Inria Nancy-Grand Est	(examiner)
Pr.	Stéphane Doncieux	Professor at Université Pierre et Marie Curie (UPMC/CNRS)	(examiner)
Dr.	Pierre-Yves Oudeyer	Research Director at Inria Bordeaux	(reviewer)
Pr.	Kenneth O. Stanley	Associate Professor at University of Central Florida	(reviewer)
Pr.	Richard A. Watson	Associate Professor at University of Southampton	(examiner)
Pr.	Alan Winfield	Professor at University of Bristol	(reviewer)

UPMC SORBONNE UNIVERSITÉS

Acknowledgments

This manuscript is the result of fruitful collaborations with many others.

First and foremost are the Ph.D. students and post-docs who took a very active part in all the experiments, analyses, and text of this manuscript. The present work is certainly more theirs than mine and I am immensely indebted to Paul Tonelli (chapter 2), Sylvain Koos (chapter 4), Antoine Cully (chapter 4), and Danesh Tarapore (chapters 4 and 5). Thank you all: working with all of you has been a pleasure and a permanent enrichment.

Jeff Clune has been my main "external" collaborator for the last three years and I cannot understate how he helped me to become the scientist I am now. Thank you Jeff: it has been successful!

The present work would not have been possible without the support of my friends and colleagues at ISIR. Special thank go to Stéphane Doncieux, who gave me the freedom to develop my own research program: I think this freedom has been a key to my "maturation". Thank you Stéphane: I truly appreciate everything you did for me. Talking about the ISIR pyramid, I also want to thank Nicolas Bredeche, Benoît Girard, and Mehdi Khamassi for the daily scientific discussions.

I had the opportunity to spend 3 months at the University of Vermont, thanks to Josh Bongard, two months at Cornell University, thanks to Hod Lipson, and five months at TU Darmstadt, thanks to Jan Peters. These stays have been eye-opening for me and I am very grateful for these wonderful opportunities.

The ANR (Creadapt project), the DGA (for the scholarship of A. Cully), and now the European Commission (ERC ResiBots) supports our work since 2013: none of the result in this manuscript would have been possible without their financial support.

I thank the member of the jury of this HDR. I admire each of them and I am proud that they accepted to play a role in this important moment of my career. Thanks to: François Charpillet, Stéphane Doncieux, Pierre-Yves Oudeyer, Ken Stanley, Richard Watson, and Alan Winfield.

Last but not least, I thank Serena Ivaldi for her enthusiasm and the everlasting support.

Résumé / *French Summary*

Tout a commencé avec une idée simple : d'après Darwin, l'évolution a dessiné toutes les espèces que l'on connait ; en tant que processus itératif, elle ressemble à un algorithme qui pourrait être exécuté sur un ordinateur ; par conséquent, pourquoi ne pas utiliser de l'*évolution artificielle* pour dessiner des *animaux artificiels*, en d'autres termes, des robots ? Cette idée simple a inspiré les scientifiques dès l'apparition des premiers ordinateurs (Turing 1950 ; Holland 1975), et a donné naissance dans les années 90 au domaine de la « robotique évolutionniste » (*Evolutionary Robotics*) (Cliff et al. 1993 ; Meyer, Husbands et al. 1998 ; Nolfi et Floreano 2001 ; Bongard 2013 ; Doncieux, Bredeche et al. 2015).

Armée d'une abstraction appropriée de l'évolution naturelle, la robotique évolutionniste devrait mener à des machines créatrices de machines (ou au moins à des algorithmes générateurs de robots) qui égaleraient les capacités de l'évolution naturelle pour concevoir des systèmes complexes. Avoir accès à de telles techniques serait très utile pour la robotique car concevoir des robots est très difficile, en particulier des robots mobiles (Nolfi et Floreano 2001 ; Bongard 2013). Il est même parfois argumenté que concevoir des robots « avancés » est si difficile que les humains pourraient bientôt être incapables d'en concevoir la prochaine génération. Par exemple, les robots marcheurs sont un défi important de la robotique « traditionnelle » (Raibert 1986 ; Kajita et Espiau 2008) alors que les animaux excellent à la locomotion à pattes. Des algorithmes évolutionnistes pourraient concevoir des contrôleurs de robot qui seraient plus efficaces que les approches classiques, puisqu'un processus similaire a mené aux impressionnantes capacités des animaux. L'évolution artificielle pourraient même dessiner le robot dans sa globalité, c'est à dire à la fois son contrôleur et son corps, de telle manière à ce que la stratégie de contrôle et la moprhologie soient intimement liées (Pfeifer et Bongard 2007 ; Pfeifer, Lungarella et al. 2007). Un telle intégration permettrait certainement à ces robots automatiquement conçus d'être plus efficace que n'importe quel robot conçu par un humain.

Cependant, un jour de 2008, une vidéo du robot Big Dog est apparue en ligne (Raibert et al. 2008). Alors que la robotique évolutionniste se penchait sur des grandes questions fondamentales sur l'évolution et l'*embodiment* (encorporation ou incarnation), la robotique *mainstream* a continué de progresser. Pour beaucoup, le robot Big Dog marque un tournant symbolique à cause de ses ses capacité locomotrices sans précédent, presque animales, et ce sur des terrains aussi divers que les sous-bois, les rochers, ou même des routes verglacées. Néanmoins, au grand désespoir de la robotique évolutionniste, la conception de Big Dog n'a pas impliqué d'évolution artificielle : ce robot s'appuie sur une actuation très réactive combinée avec des techniques de commande assez classiques mais très bien réglées (Raibert 1986 ; Raibert et al. 2008).

On pourrait argumenter que si l'entreprise qui a conçu Big Dog souhaitait créer un nouveau robot, elle aurait à recommencer tout depuis le début, ce qui lui coûterait beaucoup plus de temps et d'argent que si elle avait passé son temps à travailler sur un processus de conception plus automatisé. Peut-être. Mais l'ingénierie s'appuie sur une organisation en modules réutilisables et sur beaucoup de « bonnes pratiques » qui rendent la réutilisation possible, et parfois même facile. Dans les faits, Boston Dynamics, l'entreprise qui a conçue le Big Dog, vient juste de sortir une version plus petite de leur robot (appelée Spot), qui est basée sur des technologies similaires. On pourrait aussi argumenter que leur approche n'est utilisable que pour des robots quadrupèdes, mais Boston Dynamics a aussi démontré qu'ils pouvaient faire un robot humanoïde s'inspirant de leurs travaux sur les robots quadrupèdes (Atlas).

Globalement, Boston Dynamics n'a pas complètement résolu le problème de la locomotion, et, évidemment, elle ne s'intéresse pas directement au problème de l'intelligence artificielle « généraliste » (*general artificial intelligence*). Néanmoins, la robotique évolutionniste n'a pas non plus résolu le problème de la locomotion, en dépit d'un intérêt persistant depuis ses débuts (Lewis et al. 1992 ; Kodjabachian et Meyer 1998 ; Ijspeert, Hallam et al. 1999 ; Clune, Beckmann, Ofria et al. 2009 ; Yosinski et al. 2011), et l'intelligence artificielle généraliste apparaît toujours hors de portée. Dans tous les cas, étant donné l'état de l'art en robotique évolutionnsite, il semble qu'un long chemin reste à parcourir pour atteindre le niveau de Big Dog au niveau de la locomotion. Cet exemple des capacités locomotrices oblige à s'interroger sur les motivations initiales de la robotique évolutionniste : est-ce que la robotique est vraiment si difficile que la conception automatique sera la seule voie possible ? Et sinon, que reste-t-il à la robotique évolutionniste ?

Il y a au moins un défi majeur de la robotique qui est loin d'être résolu : l'adaptation *en ligne* à des situations vraiment imprévues, que ce soit des pannes matérielles ou des environnements particuliers. Malgré plus de 50 ans de recherche en robotique, les robots sont toujours des systèmes fragiles qui s'arrêtent facilement de fonctionner dans les conditions difficiles (Carlson et Murphy 2005). Dans les systèmes actuellement déployés, l'approche pour gérer les situations inattendue est essentiellement héritée de l'ingénierie des systèmes critiques (e.g. les centrales nucléaires ou les vaisseaux spatiaux) (Koren et Krishna 2007). Dans ces contextes, la fiabilité est généralement atteinte grâce à systèmes experts (Koren et Krishna 2007) ou des algorithmes de *machine learning* qui établissent un diagnostic de la situation, couplés à des procédures d'urgence prédéfinies (Kluger et Lovell 2006), des algorithmes de planification (Russell et Norvig 2009), ou des algorithmes d'apprentissage (Bongard 2013). Ces approches basées sur les diagnostics sont fragiles car elles combinent deux sources potentielles d'erreurs : les erreurs dans le diagnostic, qui dépend fortement de capacités sensorielles avancées, et le nombre nécessairement fini de plans possibles, qui ne peuvent couvrir tous les cas possibles qu'un robot peut rencontrer.

Les robots « bas coûts » exacerbent ces problèmes car ils peuvent être cassés de très nombreuses manières (les matériaux de bonne qualité sont chers), car ils ont des capacités sensorielles limitées (les capteurs sont chers et augmentent la complexité), et parce qu'ils visent typiquement des environnements non contrôlés (e.g. les habitations, par opposi-

tion aux usines, où les robots sont protégés de la plupart des perturbations externes).

A première vue, l'apprentissage, et non l'évolution, apparaît être la solution intuitive pour créer des robots capables de découvrir par eux-même des comportements nouveaux dans des situations imprévues. Les animaux clairement s'appuient sur l'apprentissage quand leur inhérente robustesse est insuffisante pour faire face à la situation (Wolpert et al. 2001), et non pas sur l'évolution, qui ne peut agir pendant leur vie. Néanmoins, les algorithmes évolutionnistes ont une importante caractéristique qui les distinguent de la plupart des algorithmes d'apprentissage existants : ils sont potentiellement plus créatifs. En d'autres termes, nous considérons ici les algorithmes évolutionnistes comme des *algorithmes d'apprentissage créatifs*.

Les algorithmes d'apprentissage ont une longue histoire qui s'étale sur l'ensemble de l'histoire de l'informatique (Turing 1950). Ils sont traditionnellement répartis dans trois catégories (Haykin 1998) : l'apprentissage supervisé (Barlow 1989), pour lequel il y a une base de données d'exemples d'entrées/sorties, l'apprentissage non supervisé, pour lequel il y a seulement une base de donnée d'entrées, et l'apprentissage par renforcement, (Sutton et Barto 1998), dans lequel l'agent reçoit des récompenses qui dépendent de son comportement. Certaines idées plus récentes entrent difficilement dans ce cadre, comme l'apprentissage semi-supervisé (Chapelle et al. 2010), pour lequel seulement un sous-ensemble des exemples sont étiquetés, l'apprentissage actif (Cohn et al. 1996 ; Baranes et Oudeyer 2013), dans lequel le robot joue un rôle actif dans la construction de sa base d'apprentissage, et l'apprentissage intrinsèquement motivé (Oudeyer et al. 2007 ; Baldassarre et Mirolli 2013), dans lequel le robot est plus dirigé par la curiosité que par la recherche de la performance.

L'adaptation à des situations inconnues est un problème typique d'apprentissage renforcement, car la plupart des robots ont une mission pour laquelle ils essaie de maximiser leur performance. Bien que l'apprentissage par renforcement et la robotique évolutionniste soient inspirés par des principes différents, ils s'attaquent tous deux à un problème similaire (Togelius et al. 2009 ; Whiteson 2012) : des agents (e.g., des robots) obtiennent des récompenses (resp. une *fitness*) par leur comportement dans leur environnement, et on souhaite trouver la politique (resp. le comportement) qui correspond à la somme de récompenses maximum (resp. la *fitness* maximum). Au contraire des algorithmes basés sur la différence temporelle (Sutton et Barto 1998), mais comme les algorithmes dits de *direct policy search*, la robotique évolutionniste utilise la valeur globale de la politique, et ne construit pas d'estimation pour des paires états-actions. Cette approche holistique rendent les algorithmes évolutionnistes potentiellement moins efficaces que les algorithmes d'apprentissage par renforcement classiques car ils ignorent toute l'information contenue dans les transitions entre états durant la « vie » de l'agent. Cependant, cette approche permet aussi à la robotique évolutionniste de mieux gérer les problèmes issus de l'observabilité partielle et de l'apprentissage avec des systèmes continus, comme le contrôle de robot, où les actions et les états sont souvent difficiles à définir.

De manière plus importante, l'avantage principal de la robotique évolutionniste sur l'apprentissage par renforcement est qu'il est possible d'évoluer non seulement les paramètres d'une politique, mais aussi sa structure (Stanley et Miikkulainen 2002 ; Stanley et Miikkulainen 2003 ; Whiteson 2012), ce qui permet d'éviter la difficulté (et le biais) liée au choix d'une représentation de la politique (e.g., *dynamic movement primitives* : Schaal 2006 ; Ijspeert 2008, ou espace état-action discret : Sutton et Barto 1998). Cette capacité à explorer des structure de politiques est ce qui rend les algorithmes évolutionnistes plus créatifs que la plupart des autres approches : en théorie, un algorithme évolutionniste pourrait construire une structure de contrôle dont la complexité serait limitée uniquement par la puissance de calcul disponible. Bien sûr, cet espace de recherche infini rend les algorithmes évolutionnistes plus lents que d'autres algorithmes plus contraints. L'évolution des espèces a nécessité des milliards d'années et des milliards de milliards d'individus ; on ne peut clairement pas utiliser le même temps pour permettre à un robot d'apprendre !

La biologie pose presque la question symétrique à « comment concevoir des algorithmes évolutionnistes qui rendraient les robots créatifs ? » : qu'est-ce qui fait que l'évolution naturelle est un algorithme de recherche si créatif et efficace ? Pour réaliser la vision de la robotique évolutionniste, les scientifiques ont besoin de comprendre la biologie de l'évolution avec suffisamment de détails pour pouvoir la simuler sur un ordinateur. Ils ont besoin d'une compréhension *mécanistique*. Cependant, après 150 ans de recherche, l'évolution est encore mal connue. Bien sûr, on comprend la génétique mieux que jamais (Roff 1990 ; West-Eberhard 2003), les expériences avec les bactéries donnent beaucoup d'informations sur les principes de l'évolution (Barrick et al. 2009), et les fossiles sont une précieuse fenêtre sur les espèces disparues (Morris 1993) ; mais on ne comprend toujours pas vraiment ce qui rend les espèces si *evolvable*, c'est à dire, comment elles peuvent rapidement et créativement s'adapter à de nouveaux environnements. Une conséquence directe est que les expériences avec l'évolution artificielle ont l'air d'être plus lente que l'évolution naturelle (Wagner et Altenberg 1996 ; Banzhaf et al. 2006 ; Hu et Banzhaf 2010).

Ainsi, comprendre l'evolvabilité est un défi majeur de la biologie qui une influence critique sur la robotique évolutionniste. Il semble que le développement soit une clé pour que les mutations affectent les phénotypes d'une manière qui soit à la fois exploratoire et non mortelle (West-Eberhard 2003 ; Banzhaf et al. 2006 ; Kirschner et Gerhart 2006 ; Gerhart et Kirschner 2007 ; Müller 2007), mais on ne sait pas quelles parties du processus de développement doivent être copiées, ni celles qui doivent être abstraites, ou celles qui doivent être ignorées. Il semble aussi que la modularité et la hiérarchie sont des éléments clés de l'évolvabilité car ces « bons principes d'ingéniérie » permettent à l'évolution de facilement réutiliser des modules fonctionnels dans de nouveaux contextes (Simon 1991 ; Lipson 2007). Cependant, comment la modularité et la hiérarchie ont eux-même évolué est une question ouverte (Wagner, Pavlicev et al. 2007). De plus, une meilleure compréhension des pressions de sélection est requise pour répliquer les merveilles de l'évolution naturelle dans un système artificiel. On ne sait toujours pas vraiment comment les ailes sont apparues chez les oiseaux (Dial 2003 ; Dyke et al. 2013), mais, surtout, la théorie de ce qui permet à l'évolution de créer de nouvelles caractéristiques en est encore balbutiante (Kirschner et Gerhart 2006 ; Moczek 2008). Pour toutes ces questions

classiques, mais ouvertes, les scietifiques ont besoin de séparer les concepts qui doivent être inclus dans l'évolution artificielle de ceux qui ajoutent une complexité inutile.

Dans ce manuscrit, nous soutenons la thèse que l'évolution artificielle peut contribuer à la fois à la robotique et à la biologie évolutionniste, en fournissant des nouveaux algorithmes à la robotique et des nouveaux outils à la biologie. L'adaptation est ainsi une question aux multiples facettes que nous avons explorée à partir de différents points de vue, quelques fois en biologie computationelle, et plus souvent en intelligence artificielle et en robotique. Cette approche nous mène à deux questions symmétriques :

- *Qu'est ce qui fait que les organismes naturels sont si « evolvable » ?*
- *Comment exploiter l'évolution pour rendre les robots plus adaptatifs ?*

PRINCIPALES CONTRIBUTIONS

Évolution de réseaux de neurones plastiques

Contribution publiée dans : Tonelli P., and Mouret, J.-B. (2013). *On the Relationships between Generative Encodings, Regularity, and Learning Abilities when Evolving Plastic Artificial Neural Networks.* PLoS ONE 8(11) : e79138. doi :10.1371/journal.pone.0079138

L'évolution est un processus adaptatif important, mais les adaptation comportementales mettent en jeu beaucoup d'autres mécanismes et, en particulier, l'apprentissage. En terme de capacités adaptatives pour des agents artificiels, il serait utile de combiner les adaptations dues à l'évolution, assez lente mais très exploratoires, avec celles de l'apprentissage, plus rapide mais plus contraint.

Ces deux processus sont inévitablement entrelacés car tous deux améliorent les capacités des individus à survivre dans leur environnement : beaucoup d'adaptations comportementales peuvent à la fois être le résultat de l'apprentissage et de l'évolution. Cependant, l'évolution et l'apprentissage sont presque toujours étudiés indépendamment, que ce soit en biologie (neuroscience contre biologie évolutionniste) ou en intelligence artificielle (*machine learning* contre *evolutionary computation*).

Dans cette première contribution, on s'intéresse à une question à l'intersection de l'évolution et de l'apprentissage : comment est-ce que l'évolution peut mener à des individus qui peuvent apprendre dans beaucoup de situations différentes ? Plus précisément, comment est-ce que des réseaux de neurones artificiels évolués par évolution artificielle peuvent-ils être capables d'apprendre dans des situations pour lesquelles ils n'ont jamais été directement sélectionnés ? Ou, avec un point de vue *machine learning* : comment l'évolution peut-elle éviter le danger de sur-spécialiser le mécanisme d'apprentissage, ce qui le limiterait à un sous-ensemble particulier de situations ?

Une hypothèse classique est que les capacités d'apprentissage généralistes évoluent en raison d'une possible pression de sélection directe à être capable de survivre dans des environnements changeant fréquemment. Néanmoins, cette pression pourraient ne pas expliquer complètement l'ubiquité des capacités d'apprentissage généralistes chez les animaux.

Dans cette première contribution, nous avons exploré une hypothèse alternative, qui est que le processus de développement des réseaux de neurones augmente significativement la probabilité d'obtenir des réseaux capables d'apprendre dans beaucoup de situations différentes. L'intuition sous-jacente est que ce processus de développement rend plus probable la répétition de la même structure d'apprentissage – même si certaines répétitions n'ont aucun intérêt à court terme – que la spécialisation de la structure d'apprentissage.

En utilisant une tâche de conditionnement opérant simple (4 associations à apprendre), nous avons comparé trois manières d'encoder des réseaux de neurones plastiques : un encodage direct (Mouret et Doncieux 2012a), un encodage développemental inspiré par les modèles de neuroscience (Mouret, Doncieux et Girard 2010), et un encodage développemental inspiré par les gradients morphogènes, similaire à HyperNEAT (Stanley, D'Ambrosio et al. 2009). Nous avons lancé l'évolution de ces réseaux de neurones en faisant varier le nombre d'exemples d'associations utilisés dans la fonction de fitness, puis nous avons évalué les capacités d'apprentissage en demandant aux meilleurs réseaux d'apprendre d'autres associations, c'est à dire des associations que le réseau n'a pas été sélectionné pour apprendre. Nous évaluons la régularité des réseaux en comptant le nombre d'automorphismes, une technique que nous avons introduite dans cette contribution.

Les résultats montrent qu'utiliser un encodage développemental améliore significativement les capacités d'apprentissage des réseaux de neurones plastiques évolués. Les analyses complémentaires révèlent que quel que soit l'encodage, les réseaux les plus réguliers sont ceux qui ont les meilleurs capacités d'apprentissage généralistes.

Évolution de la modularité

Contribution publiée dans : Clune* J, Mouret* J-B, Lipson H. (2013) *The evolutionary origins of modularity.* Proceedings of the Royal Society : B. 280 : 20122863. http ://dx.doi.org/10.1098/rspb.2012.2863 (* equal contribution).

Une question centrale en biologie est comment les populations sont capables de s'adapter rapidement à des nouveaux environnements, une caractéristique appelée *evolvabilité* (Pigliucci 2008). Un contributeur majeur à l'évolvabilité est le fait que beaucoup des entités biologiques sont modulaires, en particulier les processus biologiques qui peuvent être modelés avec des réseaux (e.g., les réseaux métaboliques, les réseaux régulateurs de gènes, les réseaux d'interaction de protéines, les réseaux de neurones, etc.) (Mountcastle 1997 ; Carroll 2001 ; Guimera et Amaral 2005 ; Alon 2006 ; Wagner, Pavlicev et al. 2007 ; Hintze et Adami 2008 ; Pigliucci 2008).

Les réseaux sont modulaires s'ils contiennent des groupes de nœuds hautement connectés entre eux qui sont connectés par uniquement quelques connections aux autres groupes (Striedter 2005 ; Lipson 2007 ; Wagner, Pavlicev et al. 2007). Malgré son importance et des dizaines d'années de recherche, il n'y a pas de consensus sur *pourquoi* la modularité a évolué dans la nature (Wagner, Mezey et al. 2001 ; Wagner, Pavlicev et al. 2007 ; Espinosa-Soto et Wagner 2010).

La modularité est probablement causée par plusieurs forces qui agissent avec des forces différentes (Wagner, Pavlicev et al. 2007). L'hypothèse la plus classique est que

la modularité émerge en réponse à des changements d'environnements fréquents (Kashtan et Alon 2005 ; Kashtan, Noor et al. 2007) : Intuitivement, les systèmes modulaires semblent plus adaptables, une leçon bien connue des ingénieurs (Suh 1990), car il est plus facile de « recabler » un réseau modulaire qu'un réseau « emmêlé » (Kashtan et Alon 2005 ; Kashtan, Noor et al. 2007) ; ils devraient donc être favorisés par l'évolution. Cependant, cette évolvabilité ne donne qu'un avantage sélectif à long terme, ce qui fait de la sélection pour l'évolvabilité au mieux une pression indirecte (Wagner, Mezey et al. 2001 ; Wagner, Pavlicev et al. 2007). Il est, de plus, difficile à déterminer si les environnements changent suffisamment rapidement et de manière suffisamment modulaire pour que cette pression ait une influence importante. Une théorie voisine est que la modularité évolue pour permettre de modifier des sous-composants sans affecter les autres (Espinosa-Soto et Wagner 2010), une capacité utile en environnements très variables. Il y a d'autres hypothèses (voir Wagner, Pavlicev et al. 2007), en particulier que les duplications de gènes créent un biais vers la modularité (Wagner, Pavlicev et al. 2007).

Dans cette contribution, nous étudions une hypothèse alternative qui a été suggérée précédemment mais jamais testée : la modularité évolue non pas parce qu'elle améliorer l'évovabilité, mais comme une conséquence de la sélection pour réduire les coûts de connexion dans les réseaux (Ramón y Cajal 1899 ; Striedter 2005).

Nous avons utilisé des simulations basées sur de l'évolution artificielle multi-objectif (Deb 2001). Le problème principal consiste à faire évoluer la topologie d'un réseau recevant des entrées sur une « rétine » de 8 pixels. Ce réseau doit reconnaître des motifs sur la partie droite, d'autres motifs sur la partie gauche, et répondre « vrai » s'il reconnaît à la fois un objet à gauche et à droite. Les expériences ont été répliquées sur d'autres tâches impliquant des décisions booléennes en fonction des entrées. La modularité est évaluée avec la mesure de Newman (Newman 2006) étenduee aux réseaux orientés (Leicht et Newman 2008).

Après 25 000 générations dans un environnement non variable, les expériences où les réseaux sont sélectionnés à la fois pour la performance sur la tâche et le coût de connexion mènent à des réseaux significativement plus modulaires que ceux où les réseaux ne sont sélectionnés que pour la performance.

Les réseaux évolués avec une pression sur le coût des connections sont aussi plus « evolvable » : lorsque l'on change la tâche pour une tâche voisine, ils s'adaptent plus vite que ceux évolués sans coût de connexion.

Enfin, la minimisation du coût des connexion peut agir en conjonction avec d'autres forces pour augmenter la modularité. Notamment, la modularité est encore plus haute lorsque l'on combine la pression au coût des connexions avec un environnement variant très souvent.

Nous avons continué d'étudier cette question en testant notre hypothèse avec un encodage génératif (Stanley, D'Ambrosio et al. 2009). Les intérêts de la modularité sont en effet décuplés lorsqu'elle est combinée avec la régularité, c'est à dire quand les modules peuvent être dupliqués et réutilisés. Nos résultats montrent qu'il est possible d'obtenir des réseaux réguliers et modulaires quand on combine un encodage génératif avec un coût de connexion.

Enfin, nous avons testé l'influence d'une pression sélective sur le coût de connexion dans le cas de l'évolution réseaux plastiques basés sur l'apprentissage hebbien neuromodulé (Soltoggio, Bullinaria et al. 2008). Dans ce cas, on observe que l'évolution tend à séparer le réseau en un module d'apprentissage, gérant la récompense, et un module capable d'apprendre la tâche.

Résistance aux dommages sans diagnostic

Contribution publiée dans :

- Cully, A., Clune, J., Tarapore, D., and Mouret, J.-B. (2015). *Robots that can adapt like Natural Animals*. Nature. (to appear).
- Koos, S. and Cully, A. and Mouret, J.-B. (2013). *Fast Damage Recovery in Robotics with the T-Resilience Algorithm.* International Journal of Robotics Research. Vol. 32 :14. pp 1700-1723

Lorsque que les robots quittent l'environnement contrôlé des usines pour fonctionner en autonomie dans des environnements plus complexes et plus naturels (Bellingham et Rajan 2007 ; Yoerger 2008 ; Broadbent et al. 2009), ils doivent répondre au fait qu'ils vont très probablement être endommagés (Carlson et Murphy 2005 ; Sanderson 2010). Cependant, alors que les animaux peuvent rapidement s'adapter à de nombreuses blessures, les robots actuels ne peuvent pas « penser créativement » pour trouver un comportement compensatoire lorsqu'ils sont endommagés : ils sont limités par leur senseurs proprioceptifs, qui ne peuvent diagnostiquer que les problèmes pour lesquels ils ont été conçus, et doivent ensuite déterminer un nouveau comportement en s'appuyant sur ce diagnostic, ce qui rend tout erreur de diagnostic critique (Carlson et Murphy 2005 ; Sanderson 2010).

Nous avons proposé deux nouveaux algorithmes pour résoudre ce problème. Tous deux reposent sur les mêmes concepts : (1) l'apprentissage par essai-erreur (et donc les algorithmes évolutionnistes) permet de trouver des comportements compensatoires sans nécessiter de diagnostic ou de plans prédéfinis, et (2) un simulateur du robot *intact* peut être utilisé pour guider la recherche d'un comportement compensatoire, même si le robot est endommagé. L'intuition sous-jacente est que les comportements qui n'utilisent pas les parties endommagées auront des résultats similaires en simulation et en réalité. Dans les deux cas, notre plateforme expérimentale principale est un robot hexapode cassé de 5 manières différentes, en particulier des pattes cassées et manquantes.

Notre premier algorithme, appelé T-Resilience, est basée sur l'approche par transférabilité, que nous avons initialement introduite pour traverser le *reality gap* qui sépare les résultats obtenus par évolution en simulation de ceux obtenus sur le robot. Cette approche consiste à utiliser un algorithme de regression (ici une *Support Vector Machine*, Smola et Vapnik 1997) pour apprendre les limites de la simulation, c'est à dire quand la simulation sera correcte et quand elle sera incorrecte. Pour un contrôleur, plus la simulation est proche de la réalité, plus le contrôleur est dit transférable. Concrètement, le robot utilise un algorithme évolutionniste multi-objectif (Deb et al. 2002) qui optimise deux critères : la performance en simulation et la transférabilité estimée par le SVM. Pour estimer la transférabilité, l'algorithme transfère periodiquement un individu de la population et enregistre les différences entre réalité et simulation.

Nous avons testé cette méthode sur le robot hexapode avec 5 dommages différents et sur le robot intact. Dans tous les cas, le robot a trouvé des comportements satisfaisant en moins de 25 tests sur le robot et un total de 20 minutes (en fonction de la puissance de calcul disponible, ce temps peut être plus long). Nous avons comparé notre approche à un algorithme de *policy gradient* (Kohl et Stone 2004), un algorithme de recherche locale stochastique (Hoos et Stützle 2005), et l'approche de Bongard et al. 2006. La T-Resilience mène à des résultats significativement meilleurs que ces approches dans toutes les comparaisons.

Le deuxième algorithme, appelé « intelligent trial and error », permet aux robots de s'adapter aux mêmes dommages en moins de 2 minutes et une dizaine de tests.

Avant le déploiement, le robot exploite un nouvel algorithme pour créer une carte des comportements performants. Nous avons baptisé cet algorithme « MAP-Elites ». En pratique, le robot hexapode trouve 13000 « bonnes » manières différentes de marcher et les organise sous la forme d'une carte multi-dimensionnelle. Cette carte représente les intuitions du robot à propos de quels comportements il peut utiliser et à quelle performance il peut s'attendre. L'algorithme MAP-Elites est dérivé d'idées que nous avons développées lorsque nous étudions l'évolution de la modularité et avions besoin de visualiser l'espace de recherche projeté dans les dimensions coûts de connexion / modularité.

Cette carte constitue un *prior* pour un algorithme d'optimisation bayesienne (introduction : Brochu et al. 2010 ; Borji et Itti 2013 ; Mockus 2013 ; application à la robotique : Lizotte et al. 2007 ; Calandra et al. 2014), un type d'algorithme d'optimization « boite noire » qui modélise la fonction à optimiser avec un *Gaussian process* (processus gaussien). Le robot commence donc avec un fonction de performance correspondant à ce qu'il a obtenu en simulation, et accorde à cette connaissance a priori une confiance faible. Pour trouver un nouveau comportement, le robot sélectionne le comportement le plus prometteur, puis met à jour le processus gaussien pour refléter son estimation de performance pour chaque comportement potentiel.

Dans nos expériences, le robot trouve systématiquement un bon comportement compensatoire en moins de 10 tests et 2 minutes. Nous avons réalisé des expériences similaires avec un bras robot plan à 8 degrés de liberté, que nous avons endommagé de 14 façons différentes. Le but du robot est d'atteindre un point dans le plan, typiquement pour y déposer un objet. Là encore, notre algorithme permet au robot de s'adapter en quelques minutes.

Signature d'évolvabilité

Contribution publiée dans : Tarapore, D., and J.-B. Mouret. *Evolvability signatures of generative encodings : beyond standard performance benchmarks.* Information Sciences (2015).

Les deux algorithmes de la contribution précédente permettent à des robots de trouver un nouveau comportement en quelques tests. Cependant, ils ont été – à dessein – testés avec des contrôleurs très simples (des sinusoides) afin de permettre à notre contribution de se concentrer sur l'algorithme d'adaptation, et non sur la stratégie de contrôle. Il y a deux problèmes avec ces contrôleurs : (1) ils sont en boucle ouverte, c'est à dire qu'ils ne peuvent pas s'adapter à des petits changements d'environnement, et (2) ils contraignent fortement la cinématique de la marche, ce qui peut empêcher de bien compenser certains dommages.

Les deux algorithmes précédents peuvent fonctionner avec n'importe quel type de contrôleur qui peut être évolué avec un algorithme évolutionniste. Néanmoins, il y a beaucoup de manières différentes d'implémenter des contrôleurs de marche, et encore plus de manières de les faire évoluer avec un algorithme évolutionniste. Comment choisir l'approche la plus prometteuse ?

Traditionnellement, les méthodes pour évoluer des contrôleurs de robots sont évaluées uniquement par le score de *fitness* obtenu. Cependant, le score de fitness ne fournit que des informations limitées sur la méthode. Par exemple, il ne permet pas de prédire comment le système réagira si la fonction de fitness change (e.g. suite à un dommage du robot), ou comment l'encodage explore l'espace de recherche, indépendamment de la pression de sélection.

Pour mieux comparer les encodages de contrôleurs pour la robotique évolutionniste, nous avons proposé une nouvelle méthode, appelée « signatures d'évolvabilité », qui dessine la distribution statistique de la diversité des comportements et de la fitness après une étape de mutation. Cette méthode s'appuie sur la définition de l'évolvabilité qui inclut à la fois la variabilité et la viabilité des comportements générés, c'est à dire à quel point ils sont différents et à quel point ils restent performants.

Nous avons testé la pertinence de notre concept en évoluant des contrôleurs pour la marche hexapode. Nous avons utilisé 5 encodages différents : encodage direct de paramètres d'un Central Pattern Generator (CPG, Ijspeert 2008), encodage par CPPN des paramètres d'un CPG en boucle ouverte, encodage par CPPN (Stanley 2007) des paramètres d'un CPG en boucle fermée, et le contrôleur SUPG (Morse et al. 2013), aussi basé sur des CPPN. Nous avons observé une relation prédictive entre la signature d'évolvabilité de chaque encoding et le nombre de générations requis pour qu'un robot hexapode s'adapte à un dommage. Notre étude révèle aussi que le contrôleur SUPG a la signature d'évolvabilité la plus prometteuse car il est capable de générer des comportements différents avec peu de perte de fitness. Dans nos expériences, cet encodage est aussi celui qu permet de trouver le plus rapidement un nouveau comportement après la perte d'une patte.

Plus généralement, les signatures d'évolvabilité apportent un complément aux mesures basées uniquement sur des benchmarks en offrant une vision plus riche du comportement de chaque encodage. Elles nous permettront d'affiner notre choix du meilleur type de contrôleur/genotype pour nos robots.

DISCUSSION

Bio-inspiration et outils communs

Nous avons commencé ce manuscrit avec une contribution qui visait l'intelligence artificielle, mais nos expériences nous ont montré à quel point nous étions loin de pouvoir utiliser l'évolution pour créer automatiquement des réseaux de neurones capables d'apprendre. On peut légitimement se demander si de telles méthodes pourront un jour créer des réseaux de neurones capables d'approcher l'état de l'art en apprentissage par renforcement (e.g., Deep Q-Learning : Mnih et al. 2015).

De manière inattendue, cette contribution ouvre peut-être plus possibilité pour comprendre l'évolution des capacités cognitive. En particulier, elle donne des outils pour analyser les compromis entre la flexibilité, c'est-à-dire la capacité à apprendre dans beaucoup de situations, l'entrainabilité, c'est-à-dire la capacité à apprendre facilement et rapidement, et la compacité, c'est-à-dire la tendance générale à supprimer les connexions inutiles (Clune* et al. 2013).

Notre deuxième contribution, les origines évolutionnistes de la modularité, visait explicitement la biologie. Néanmoins, notre travail sur cette contribution nous a mené à introduire notre algorithme de cartographie des élites, qui est la pierre angulaire de notre troisième contribution, la résistance aux dommages, qui est explicitement dédiée à la robotique.

Dans l'approche traditionnelle de la robotique bio-inspirée, une découverte est faite en biologie, un modèle abstrait est implémenté sur un robot, et le robot est amélioré grâce aux idées issues de la biologie (Pfeiffer 2007 ; Meyer et Guillot 2008 ; Kovač 2013). Par contraste, dans notre approche, la biologie évolutionniste et la robotique évolutionniste s'attaquent à des questions différentes mais les *outils* sont similaires. En d'autres termes, *l'inspiration n'est pas dans le résultat, mais plutôt dans le chemin qui a mené au résultat.*

Cette inspiration est bi-directionnelle, puisque nous avons à la fois utilisé des outils de la robotique évolutionniste pour offrir un nouveau regard sur des questions de biologie, et des outils utilisés en biologie évolutionniste pour contribuer à la robotique. Cette convergence des outils est illustrée par le fait que toutes les expériences de ce manuscrit utilisent le même *framework* logiciel, sferes$_{v2}$ (Mouret et Doncieux 2010).

De meilleurs outils pour de meilleures fondations

Malgré ses racines en informatique, la robotique évolutionniste est une science expérimentale. L'approche classique pour savoir si une idée est plus prometteuse qu'une autre est de lancer un ensemble d'expériences avec différentes approches, mesurer la meilleure fitness obtenue, et conclure. Cette méthodologie peut mener à des améliorations incrémentales quand le but est de résoudre le problème utilisé dans le *benchmark* ou un problème très proche. Cependant, en robotique évolutionniste, les « benchmarks » sont en général des étapes potentielles sur le chemin d'autres problèmes plus ambitieux. Dans cette situation, simplement analyser la meilleure fitness est de courte vue : pour la même idée, il est très probable que la conclusion de la comparaison change si l'expérimentateur utilise d'autres paramètres (souvent nombreux) et d'autres tâches (Karafotias et al. 2014).

En conséquence, il y a un besoin clair d'outils d'analyse en robotique évolutionniste et, plus généralement, en « evolutionary computation » ; c'est pourquoi une constante des travaux de ce manuscrit est de proposer de nouvelles manières d'analyser et de visualiser les résultats obtenus en robotique évolutionniste.

Mesurer la régularité dans les réseaux. Dans notre première contribution (évolution de réseaux de neurones plastiques), nous avons introduit une nouvelle manière de mesurer la régularité des réseaux. Notre approche est fondée sur des résultats classiques de théorie des graphes, les automorphismes, et elle est rapide, notamment grâce aux bibliothèques existantes pour énumérer les automorphismes des graphes (McKay 1981 ; Junttila et Kaski 2007 ; Katebi et al. 2012). Notre approche est une des rares approches permettant de mesurer la régularité d'un réseau. L'alternative classique est de compresser la matrice de connectivité (Clune, Stanley et al. 2011), ce qui est infaisable exactement car il faudrait pouvoir compresser tous les arrangements de nœuds dans la matrice (en plus d'avoir un algorithme de compression optimal).

Dessiner des réseaux de neurones. Pour mesurer la longueur des connexions lorsque nous étudions les origines de la modularité, nous avons eu besoin de placer certains neurones à des positions fixes et de placer les autres à la position qui minimise la somme du coût des connexions. Ce problème peut être résolu exactement par programmation quadratique (Chklovskii 2004). Incidemment, cette approche permet de dessiner des réseaux de neurones *feed-forward* et nous l'avons utilisée plusieurs fois dans ce manuscrit.

Visualiser les elites d'un espace de recherche. Bien que les espaces de recherche et les paysages de fitness soient au centre de beaucoup de discussion en optimisation et biologie, on peut rarement les *voir* car ils sont généralement de trop haute dimension. La littérature en statistique offre de nombreuses options pour réduire la dimension d'un jeu de données et visualiser des données de haute dimension (Andrews 1972 ; Haykin 1998 ; Tenenbaum et al. 2000 ; Kohonen 2001). Néanmoins, ce sont des algorithmes « passifs » qui commencent avec un jeu de données et cherchent la meilleure représentation. Ils ne s'attaquent pas au problème de générer ce jeu de données.

Par contraste, pour identifier les pics de fitness, nous devons les chercher *activement*. Il n'est pas suffisant d'échantillonner aléatoirement des millions de solutions puis de les afficher, pour la même raison que l'échantillonnage aléatoire est généralement un mauvais algorithme d'optimisation : trouver par chance un pic de fitness est très improbable pour n'importe quel espace de recherche un peu grand.

C'est pourquoi l'algorithme MAP-Elites, initialement développé pour visualiser la relation coût/modularité/performance puis utilisé en robotique, est un algorithme de recherche et non un algorithme de réduction de dimension classique ; il pourrait être vu comme un algorithme de réduction de dimension actif, une catégorie d'algorithme qui reste encore à créér.

Signatures d'évolvabilité. Ces signatures, que nous avons déjà décrites précédemment partagent plusieurs caractéristiques avec les autres méthodes que nous avons proposées :

- comme MAP-Elites, c'est un outil visuel qui dessine des images en 2 dimensions ;
- comme MAP-Elites et notre travail sur la diversité comportementale (Mouret et Doncieux 2012a), il est basé sur un concept de distance comportementale ;
- comme notre mesure de régularité, les signatures révèlent des caractéristiques intrinsèques des encodages sans nécessiter de choisir si ces caractéristiques sont bonnes ou mauvaises.

Construire des fondations solides

La robotique évolutionniste est un projet à long terme qui nécessite des fondations solides pour pouvoir continuer à

progresser. Au delà du manque d'outils d'analyse, la robotique évolutionniste souffre d'un manque de connaissances générales : après des dizaines d'années de recherche, peu de faits sont établis.

Par essence, une expérience de robotique évolutionniste met en jeu trois composants principaux : une tâche, un encodage, une pression de sélection, et un algorithme évolutionniste. Beaucoup d'articles testent plusieurs tâches, mais seulement une poignée d'entre eux essaye de généraliser dans les 3 autres dimensions.

Dans notre travail sur la diversité comportementale (Mouret et Doncieux 2012a), nous avons testé 2 encodages, 3 tâches, et 3 méthodes différentes pour appliquer une pression de sélection pour la diversité. Grâce à cette approche, nous avons pu atteindre une conclusion assez générale : une pression de sélection qui encourage la diversité comportementale a beaucoup plus d'impact sur le succès d'une expérience que l'encodage, quel que soit la technique utilisée pour encourager la diversité.

Dans notre contribution sur la plasticité synaptique, nous avons utilisé 3 encodages différents, ce qui nous a permis de dresser des conclusions qui devraient être valable pour n'importe quel encodage biaisé vers la régularité, y compris ceux qui n'ont pas encore été conçus.

Dans notre contribution a propos des signatures d'évolvabilité, nous avons étudié les encodages indépendamment de l'algorithme évolutionniste et de la pression de sélection.

Globalement, ces contributions montrent qu'il est possible de poser des fondations à condition de développer de nouveaux outils d'analyse et d'aller au delà d'articles qui comparent uniquement la fitness obtenue par des systèmes complets (encodage + pression de sélection + algorithme évolutionniste + paramètres + tâche).

PROJET DE RECHERCHE

Pour les cinq prochaines années, je vais principalement me concentrer sur les applications à la robotique et plus spécialement sur la résistance aux dommages. Je vais continuer de collaborer avec des biologistes, mais ce ne sera pas central dans mon travail.

Vers des robots hautement résilients

Mon objectif principal est de poursuivre la voie ouverte par l'approche « Intelligent Tirla and Error » pour proposer de nouveaux algorithmes qui permettront à des robots autonomes de découvrir de nouveaux comportements lorsqu'ils sont endommagés ou lorsqu'ils sont face à une situation inattendue. Nous continuerons à nous appuyer sur un algorithme évolutionniste type « MAP-Elites » et sur des méthodes de recherche comme l'optimisation bayesienne.

Dans cinq ans, nous devrions être capables de mettre un robot cassé dans une pièce normale (e.g. un appartement) et il devra découvrir un nouveau moyen de poursuivre sa tâche. Par exemple, un robot marcheur avec une patte cassée aura à découvrir en toute autonomie comment marcher à nouveau, pour toutes le directions, et en prenant en compte les obstacles pendant son apprentissage.

Notre algorithme idéal aura les caractéristiques suivantes :

Général : le même algorithme devrait être capable de fonctionner avec des modifications mineures sur n'importe quel robot et n'importe quelle tâche, en particulier pour les robots marcheurs, les manipulateurs mobiles, et les robots humanoïdes.

Rapide : le processus ne devrait pas prendre plus de quelques minutes, en particulier pour éviter d'endommager le robot encore plus.

Créatif : le processus doit être aussi créatif que possible.

Multi-tâche : apprendre à faire une unique tâche est rarement suffisant. Par exemple, apprendre à marcher dans une unique direction est inutile.

Deployable : les robots n'effectuent pas leur mission dans des pièces expérimentales vides ; ils doivent faire avec leur environnement pendant qu'ils apprennent.

Multi-objectif : la plupart des robots doivent optimiser plusieurs objectifs antagonistes, par exemple maximiser la vitesse de déplacement et minimiser la consommation énergétique.

Ce projet est financé pour 2015-2020 par la commission européenne (ERC ResiBots).

Les origines évolutionnistes de la généralisation comportementale

Cette partie de mon projet sera réalisée en collaboration avec Jean-Baptiste André (Institut des Sciences de l'Evolution - CNRS - Montpellier) et Nicolas Bredeche (ISIR, Université Pierre and Marie Curie).

L'objectif de ce projet est de proposer un nouveau regard sur les innovations évolutionnistes dans le cadre des innovations comportementales en se concentrant sur le rôle de la généralisation (Parter et al. 2008 ; Watson et al. 2014) – la capacité pour les individus de se comporter de manière adaptative dans des environnements qui n'ont jamais été rencontrés par leur espèce. Pour cela, nous nous appuierons sur nos travaux sur la plasticité synaptique, pour l'apprentissage en ligne basé sur des neurones, et sur l'optimisation bayesienne, pour l'apprentissage en ligne rapide. Nous combinerons ces techniques avec l'évolution guidée par l'environnement (*environment-drive evolution*) car c'est un modèle intéressant de l'évolution naturelle qui peut prendre en compte l'environnement explicitement (Bredeche et al. 2012).

Nous nous attaquerons à deux grands défis :

- proposer un modèle d'évolution *environment-driven* qui prend en compte les capacités d'apprentissage ;
- tester les effets des facteurs environnementaux sur l'émergence et la maintenance des capacités de généralisation et/ou d'apprentissage.

CONCLUSION

J'ai commencé ma thèse avec l'envie de rendre les robots plus adaptatifs en s'inspirant de l'évolution, mais je me suis finalement concentré l'amélioration des techniques d'évolution de réseaux de neurones (Mouret et Doncieux 2008 ; Mouret et Doncieux 2009, 2012a). L'essentiel du travail que j'ai conduit depuis ma thèse est un retour à ma motivation initiale : faire des robots capables de survivre dans un monde dynamique et imprédictible.

En effectuant des aller-retour entre des questions de biologie évolutionniste et de robotique, nous avons maintenant une meilleure idée de :

- comment faire évoluer des réseaux de neurones plastiques avec capacités d'apprentissages en ligne généralistes ⇒ en utilisant des encodages génératifs ;
- comment les réseaux biologiques peuvent-ils adapter leur comportements en quelques générations ⇒ en étant modulaires, grâce notamment à la pression de sélection qui pousse à minimiser les coûts de connexion ;
- quels encodages pour les réseaux de neurones combinent créativité et viabilité des solutions ⇒ grâce aux signatures d'évolvabilité, nous pouvons conclure que SUPG est le meilleur encodage de notre ensemble de test pour notre tâche (la locomotion) ;
- comment exploiter l'évolution artificielle pour permettre l'adaptation rapide, en ligne, et créative en robotique ⇒ en remplissant beaucoup de niches en simulation avec un algorithme évolutionniste et cherchant parmi celles-ci avec une optimisation bayesienne.

Chacune de ces étapes nous rapproche de robots qui peuvent réagir de manière créative et rapide à des situations inattendues. Dans le futur, nous ferons de notre mieux pour :

- aller le plus loin possible vers des expériences réalistes ⇒ mettre un robot endommagé dans un appartement, attendre 2 minutes, et le robot devrait avoir découvert un moyen de compenser le dommage ;
- faire que nos résultats soient aussi généraux que possible ⇒ tester toutes nos idées sur plusieurs robots différents, et, si possible, pour différents encodages/algorithmes/techniques ;
- continuer à s'intéresser à la fois à des questions théoriques en biologie évolutionniste et des questions appliquées en robotique ;
- continuer de développer de nouveaux outils d'analyses qui vont au delà d'une simple mesure de performance.

Contents

Introduction & state-of-the-art

It all started with a simple idea: according to Darwin, evolution shaped all the species we know; as an iterative process, it looks like an algorithm that could be executed by a computer; therefore, why not using *artificial* evolution to design *artificial* animals, that is, robots? This simple idea inspired scientists since the fifties (Turing 1950; Holland 1975) and gave birth in the nineties to the field called "Evolutionary Robotics" (ER) (Cliff et al. 1993; Meyer, Husbands, et al. 1998; Nolfi and Floreano 2001; Bongard 2013; Doncieux, Bredeche, et al. 2015).

Armed with the appropriate abstraction of natural evolution, evolutionary robotics should lead to robot-creating machines (or at least robot-generating algorithms) that would equal the abilities of natural evolution to design complex system. Having such techniques would be very useful for robotics because designing robots, especially mobile robots, is very difficult (Nolfi and Floreano 2001; Bongard 2013). Actually, one can think that designing complex robots is so difficult than humans could soon be unable to design the next generation of machines. In this case, algorithms will have to replace or at least assist humans in the design process. For instance, walking robots is a open challenge for "traditional robotics" (Raibert 1986; Kajita and Espiau 2008) but for which animals excel. Evolutionary algorithms could design robot controllers that would be more efficient than classic approaches, since a similar process led to amazingly high-performing animals. Artificial evolution could even design the whole robot, that is, both its controller and its body, so that its control strategy and its morphology are tightly integrated (Pfeifer and Bongard 2007; Pfeifer, Lungarella, et al. 2007). Such a tight integration would undoubtedly make the robot much more efficient than any human-made design.

But, one day of 2008, a video of the Big Dog robot appeared online (Raibert et al. 2008) (Fig. 1.1). While evolutionary robotics was asking big and fundamental questions about evolution and embodiment, "mainstream" robotics continued to progress. For many, the Big Dog robot is a symbolic turning point because of its unprecedented, almost life-like, walking performance in natural terrains, from forest to rocks and icy roads. However, to evolutionary robotics' utter despair, the design of Big Dog did not involve artificial evolution: this robot uses very reactive actuation combined with finely tuned but classic control techniques (Raibert 1986; Raibert et al. 2008).

One could argue that if the company that designed the Big Dog wanted to create a new robot, they would have to start again from scratch and that this would cost them much more money than if they had spent their time designing an automatic design method. Maybe. But engineered modules and technical knowledge are precisely designed to be as transferable as possible. Thousands of years of engineering led to many good practices to make reuse possible, if not easy. In fact, Boston Dynamics, the company that designed the Big Dog, just unveiled a smaller version of their robot (*Spot*, a dog-sized robot), which is based on similar technology. One could also argue than their approach is only good for quadruped robots, but they successfully demonstrated a biped robot (Atlas) inspired by their work on the quadruped.

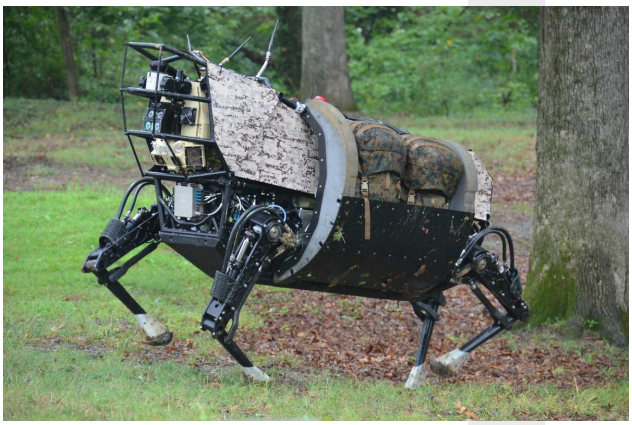


Figure 1.1. LS3, the successor of Big Dog. (image (c) 2012, Tactical Technology Office, Defense Advanced Research Projects Agency, U.S. Department of Defense). The Big Dog did not need evolution or learning to achieve impressive locomotion abilities. However, it might need evolution and/or learning to cope with some unexpected situations, like, for instance, a partially broken leg.

Overall, Boston Dynamics did not fully solve the problem of robotic locomotion (yet) and obviously did not even tackle the question of general artificial intelligence. Nonetheless, evolutionary robotics did not solve the locomotion problem either, in spite of a continuous interest since its beginnings (Lewis et al. 1992; Kodjabachian and Meyer 1998; Ijspeert, Hallam, et al. 1999; Clune, Beckmann, Ofria, et al. 2009; Yosinski et al. 2011), and general artificial intelligence is still out of reach. At any rate, given the state of the art in evolutionary robotics, there appears to be a long journey before reaching the level of Big Dog for locomotion skills. The locomotion example therefore casts doubts on the initial motivation of evolutionary robotics: is robotics really so difficult that automatic design will be the only possible approach? And, if not, what is left for evolutionary robotics?

There is at least one challenge in robotics that is far from being solved: online adaptation to truly unforeseen situations. Despite over 50 years of research in robotics, robots are still fragile systems that easily stop functioning in difficult conditions. In currently deployed systems, the approach to handle unexpected situations is inherited from engineering of safety-critical systems (e.g. nuclear plants or spaceships) (Koren and Krishna 2007). In these contexts, reliability is typically achieved thanks to expert systems (Koren and Krishna 2007) or machine learning algorithms (Polycarpou and Helmicki 1995) that diagnose the situation, coupled with predefined contingency plans (Kluger and Lovell 2006), planning algorithms (Russell and Norvig 2009), or learning algorithms (Bongard 2013). Such diagnosis-based approaches are brittle because they combine two sources of potential errors: errors in the diagnosis, which critically depends on extensive sensing abilities, and the necessary limited number of possible plans, which hardly cover all the possible situa-

tions that a robot may encounter. Low-cost robots exacerbate these issues because they can be broken in many ways (high-quality material is expensive), because they have low sensing abilities (sensors are expensive and increase the overall complexity), and because they are typically targeted towards non-controlled environments (e.g. houses, by opposition to factories, in which robots are protected from most unexpected events).

At first, learning, and not evolution, appears to be the intuitive solution to make robots that can discover novel behaviors in unforeseen situations. Animals certainly rely on learning when their inherent robustness is not sufficient to cope with the situation (Wolpert et al. 2001), and not on evolution, which cannot act during their lifetime. Nonetheless, evolutionary algorithms have important feature that distinguishes it from most learning *algorithms*: it has the potential of being much more creative. In other words, we here see evolutionary algorithms as *creative learning algorithms*.

Learning algorithms have a long history that spans over the whole history of computer science (Turing 1950). They are traditionally divided into three categories (Haykin 1998): supervised learning, for which there is a database of input/output examples, unsupervised learning (Barlow 1989), for which there is only a database of inputs, and reinforcement learning (Sutton and Barto 1998), for which the agent receives rewards that depends on its behavior. Some more recent ideas do not fit well this framework, like semi-supervised learning (Chapelle et al. 2010), in which only a subset of the examples are labeled, active learning (Cohn et al. 1996; Baranes and Oudeyer 2013), in which the robot robot plays an active role in building its own database of experiences, and intrinsically motivated learning (Oudeyer et al. 2007; Baldassarre and Mirolli 2013), in which the robot is driven more by curiosity than by performance.

The adaptation to unforeseen situations is a typical reinforcement learning *problem* because most robots have a mission for which they need to maximize their performance. Although reinforcement learning and evolutionary robotics are inspired by different principles, they both tackle a similar challenge (Togelius et al. 2009; Whiteson 2012): agents (e.g., robots) obtain rewards (resp. fitness values) while behaving in their environment, and we want to find the policy (resp. the behavior) that corresponds to the maximum reward (resp. fitness). Contrary to temporal difference algorithms (Sutton and Barto 1998), but like direct policy search algorithms (Kober and Peters 2013), evolutionary robotics only uses the global value of the policy and does not construct value estimates for particular state-action pairs. This holistic approach makes evolutionary robotics potentially less powerful than classic reinforcement learning because it discards all the information in the state transitions observed during the "life" (the evaluation) of the individual. However, this approach also allows evolutionary robotics to cope better than reinforcement learning with partial observability and continuous domains like robot control, where actions and states are often difficult to define.

More importantly, the main advantage of evolutionary robotics over reinforcement learning is that it can evolve not only the parameters of a policy but also its structure (Stanley and Miikkulainen 2002; Stanley and Miikkulainen 2003; Whiteson 2012), hence avoiding the difficulty (and the bias) of choosing a policy representation (e.g., dynamic movement primitives (Schaal 2006; Ijspeert 2008)) or a discrete state-action space (Sutton and Barto 1998)). This ability to explore the structure of a policy is what makes evolutionary algorithms more creative than most other approaches: in theory, an evolutionary algorithm could construct a control structure of unbounded complexity and be limited only by the available computational power. Of course, this unbounded search space makes evolutionary algorithm slower than more constrained learning algorithm. Natural evolution took billions of years and billions of billions of individuals; we certainly cannot use the same amount of time to allow a robot to learn!

Biology almost mirrors the question of how to design evolutionary algorithms that would make robots creative: what makes natural evolution such a creative and efficient "search algorithm"? To realize the vision of evolutionary robotics, scientists need to understand evolutionary biology in enough detail to simulate it on a computer. They need a *mechanistic* understanding of evolution. However, after 150 years of research, evolution is still poorly understood. Of course, we understand genetics better than ever (Roff 1990; West-Eberhard 2003), experiments with bacteria provides many insights into the working principle of evolution (Barrick et al. 2009), and fossil records are invaluable windows into ancient species (Morris 1993); but we still do not fully understand what make species so evolvable (Kirschner and Gerhart 1998; Carroll 2001; Kirschner and Gerhart 2006; Pigliucci 2008; Pavlicev and Wagner 2012), that is, how they can quickly and creatively adapt to new environments. A direct consequence is that experiments with artificial evolution does not seem to have the same evolution speed as natural evolution (Wagner and Altenberg 1996; Banzhaf et al. 2006; Hu and Banzhaf 2010).

Hence, understanding evolvability is a major challenge of biology that has a critical impact on evolutionary robotics. It seems that development is a key to make mutations effect phenotypes in a way that is both exploratory and non-lethal (West-Eberhard 2003; Banzhaf et al. 2006; Kirschner and Gerhart 2006; Gerhart and Kirschner 2007; Müller 2007), but we do not know which parts of the developmental process need to be copied, which ones can be abstracted, and which ones can be ignored. It also seems that modularity and hierarchy are key drivers of evolvability because these "good engineering principles" allow evolution to easily re-use functional modules in new contexts (Simon 1991; Lipson 2007). However, how modularity and hierarchy themselves evolved is an open question (Wagner, Pavlicev, et al. 2007). In addition, a better understanding of selective pressures is needed to replicate the wonders of natural evolution in artificial systems. We still do not know how wings appeared in birds (Dial 2003; Dyke et al. 2013), but, more importantly, the theory about what drives the evolution of novel features is still in its infancy (Kirschner and Gerhart 2006; Moczek 2008). For all these classic but open questions, scientists need to separate concepts that need to be included in artificial evolution from those that are unnecessarily adding complexity.

It is contended here that artificial evolution can contribute to both robotics and evolutionary biology by providing novel algorithms to robotics and novel tools to biology. Adaptation is thus a multi-faceted question that we explored from different points of view, a few times in computational biology, and more often in artificial intelligence

and robotics. This approach led us to two symmetrical questions:

- biology: *"what makes natural organisms so evolvable?"*
- robotics: *"how to harness evolution to make robots more adaptive?"*

While the answers to the biology question could feed the robotics one, we did not systematically establish a direct link in our work because our contributions in biology were too preliminary to make a difference in robotics. We obviously hope to establish a stronger link in the future, although it is not our main goal. Nonetheless, we re-used the *tools* developed for biology in our robotics experiments. In particular, the algorithm that we designed to picture the search landscape when we were studying the origins of modularity (chapter 3) is the basis of our algorithm for damage recovery in robotics (chapter 4). Reciprocally, the techniques developed for evolutionary robotics and evolutionary intelligence are useful to understand biology. For instance, our models to study the evolution of modular networks (chapter 3) is based on multi-objective optimization (Deb 2001), which comes from engineering, and takes inspiration from the ongoing discussions in evolutionary robotics about selective pressures (Mouret 2011b; Mouret and Doncieux 2012a; Doncieux and Mouret 2014). This bi-directional technical connection between our contributions is illustrated by the fact that *all* the experiments reported in this manuscript rely on the same software framework for evolutionary computation (Sferes$_{v2}$, see Mouret and Doncieux (2010)).

Contributions (chapters):
– Evolving plastic neural networks
– Evolving modular neural networks
– Diagnosis-free adaptation to damage
– Evolvability signatures

The following chapters are condensed versions my main articles to evolutionary adaptation in nature and robots. Thus, the most of the text is adapted from the original articles. Please see them (listed at the beginning of each chapter) for the full methods, technical details. These chapters do not constitute an exhaustive view of my work, but more a snapshot of the advances of my research project since the end of my PhD. In particular, I purposely did not include the work that is a direct continuation of my PhD (mainly publications about behavioral diversity, in collaboration with my PhD advisor – S. Doncieux).

The next chapter (chapter 2) investigates how an evolutionary algorithm could design large neural networks with life-time learning abilities. It combines two concepts at the same time, Hebbian learning and generative encodings for neural networks, and it shows that they can benefit from each other. This chapter contributes to the question of how an algorithm could design an "artificial nervous system" with online adaptation abilities, an artificial intelligence question, and suggests discussions about the evolution of learning abilities in nature.

The third chapter investigates the evolutionary origins of modularity, a central question to understand how species adapt to new environments. We believe that a better understanding of the origins of modularity is likely to improve the adaptive abilities of an artificial evolution system in dynamic scenarios. This chapter explores a new hypothesis to explain the evolutionary origins of modularity in biological networks, which is that modularity emerges because of the ubiquitous selective pressure to minimize the cost of connections.

The fourth chapter reports our experiments with evolution for on-line damage recovery with real robots. It introduces an algorithm that leverages a novel evolutionary algorithm (MAP-Elites) and Gaussian processes to allow a hexapod robot to recover from many damage conditions in less than 2 minutes, that is, much faster than the state of the art.

The fifth chapter is about the evolution of closed- and open-loop neuro-controllers for hexapod locomotion, with the underlying goal of using such controllers with the algorithm of the previous chapter. We needed new tools to compare different encodings and we decided to introduce a novel method to compare them. As a consequence the contribution of this chapter is twofold: an extensive benchmark of possible methods to evolve neuro-controllers for locomotion, and a novel approach to compare the evolvability provided by encodings.

The sixth chapter provides some additional discussions on our methodology and our results.

The seventh chapter describes my research project for the next years.

STATE-OF-THE-ART

Each chapter includes its own state-of-the-art. We here briefly introduce the questions and techniques that are shared by several chapters.

Nature-like evolvability in artificial evolution

To make evolution possible, random mutations need to be reasonably likely to produce viable (non-lethal) phenotypic variations (Kirschner and Gerhart 1998; Gerhart and Kirschner 2007). Viable variations are consistent with the whole organism. For instance, a variation of beak size in a finch requires consistent adaptation of both mandibles (lower and upper), as well as an adaptation of the musco-skeletal structure of the head (West-Eberhard 2003). In addition, viable variations do not break down the vital functions of the organism: an adaptation of the beak should not imply a change in the heart. Likewise, a mutation that would add a leg to a robot would require a new control structure to make it move, but the same mutation should not break the vision system. This kind of consistency and weak linkage is impossible to achieve in a simple artificial evolutionary system in which each gene is associated to a single phenotypic trait (Wagner and Altenberg 1996; Hu and Banzhaf 2010).

In biology, the development process that links genes to phenotypes, called the genotype-phenotype map, is thought to be largely responsible for organisms' ability to generate such viable variations (Wagner and Altenberg 1996; Kirschner and Gerhart 2006; Gerhart and Kirschner 2007; Müller 2007). Inspired by the progress made by the evo-devo community (Müller 2007), many researchers in ER have focused on the genotype-phenotype map for neural networks(e.g. SGOCE: Kodjabachian and Meyer 1998; Modnet: Doncieux and Meyer 2004; MENNAG: (Mouret and Doncieux 2008); HyperNEAT: Stanley and Miikkulainen 2003; Stanley, D'Ambrosio, et al. 2009; Verbancsics and Stanley 2011) or complete robots (e.g. GRNs: Bongard 2002; Cussat-Blanc and Pollack

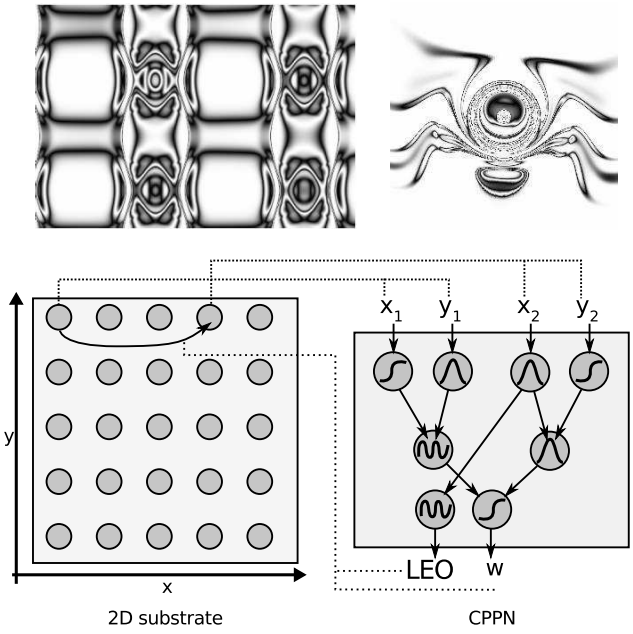


Figure 1.2. **Concept of the HyperNEAT encoding.** Top: two examples of patterns generated with a 2-dimensional CPPN (2 inputs, which correspond to the x,y coordinate of each pixel, and 1 output, which corresponds to the grey level of the pixel). Images from picbreeder (Secretan et al. 2008): http://www.picbreeder.org. Bottom: principle of HyperNEAT, which evolves a CPPN that is queried to obtain both the connectivity (LEO) and the weights (w) of a neural network.

2012, L-systems (Hornby and Pollack 2002), CPPNs: Auerbach and Bongard 2014; Cheney, Clune, et al. 2014). Essentially, they propose genotype-phenotype maps that aim to reproduce as many features as possible from the natural genotype-phenotype map, and, in particular, consistency, regularity (repetition, repetition with variation, symmetry), modularity (functional, sparsely connected subunits), and hierarchy (Lipson 2007).

HyperNEAT (Stanley, D'Ambrosio, et al. 2009; Gauci and Stanley 2010; Clune, Stanley, et al. 2011) is a particular encoding that we use several times in this manuscript because it captures most of the previously mentioned properties while being abstract enough to be easy to implement. It is broadly inspired by the chemical gradients in biological cells (Stanley 2007). To generate a network, HyperNEAT requires a geometric layout of nodes that are typically related to the morphology of the robot[1]. To know how a particular neuron is connected to another one, HyperNEAT uses the coordinates of the source and target neuron to query a network of functions called a Compositional Pattern Producing Network (CPPN) (Stanley 2007).

The CPPN is what is evolved by the evolutionary algorithm and constitutes the genome of an individual, that is, each individual has a different CPPN. A typical CPPN for neural networks has 4 inputs (x,y coordinate for the target and the source) and 2 outputs (the synaptic weight and a Link-Expression Output (LEO), which tells whether the connection exists), although many other variants exist (in more than 2 dimensions, without the LEO, etc.). CPPNs are feed-forward networks that is very similar to a classic neural networks, therefore they can be evolved with classic direct encodings used to evolve neural networks (Stanley and Miikkulainen 2002; Mouret and Doncieux 2012a). The only difference between a CPPN and a classic neural network is that each node can have a different activation function, instead of the classic sigmoid used in neural networks. Typical functions are Gaussian (symmetry), sigmoids (non-linearity) and sine waves (repetition). By evolving the topology and the parameters of the network (the weights, the biases,and the activation function), it is possible to express many nature-like patterns (figure 1.2). For instance, a CPPN for which $f(x_1, y_1, x_2, y_2) = f(x_2, y_1, x_1, y_2)$ encodes a symmetry along the x axis. HyperNEAT often outperforms other encodings for problems that are highly regular (Clune, Stanley, et al. 2011) or for which there is a large input space, like a camera or boardgames (D'Ambrosio and Stanley 2008; Stanley, D'Ambrosio, et al. 2009; Gauci and Stanley 2010; Gauci and Stanley 2011).

Generative encodings like HyperNEAT can generate networks that can be modular, hierarchical and regular. Intuitively, these properties should improve evolvability and should therefore be selected by the evolutionary process: if the "tools" to create, repeat, organize, and combine modules are available to evolution, then evolution should exploit them because they are beneficial. However, evolvability only matters in the long term, and evolution mostly selects on the short term: long-term evolvability is a weak, second-order selective pressure (Wagner, Pavlicev, et al. 2007; Pigliucci 2008; Woods et al. 2011). As pointed out by Pigliucci (2008), "whether the evolution of evolvability is the result of natural selection or the by-product of other evolutionary mechanisms is very much up for discussion, and has profound implications for our understanding of evolution in general."

The ongoing discussions about the evolutionary origins of modularity illustrate some of the direct and indirect evolutionary mechanisms at play: the current leading hypotheses suggest that modularity might evolve to specialize gene activity (Espinosa-Soto and Wagner 2010), or, more directly, to adapt in rapidly varying environments (Kashtan and Alon 2005; Clune, Beckmann, McKinley, et al. 2010). In chapter 3, we explore the hypothesis that it evolved as a by-product of the pressure to minimize connection costs. These questions about the importance of selective pressures in understanding evolvability echo the recent results that suggest that classic objective-based fitness function may hinder evolvability: nature-like artificial evolvability might require more open-ended approaches to evolution than objective-based search (Lehman and Stanley 2013).

Overall, despite efforts to integrate ideas about selective pressures and genotype-phenotype maps to improve evolvability, much work remains to be done on the way to understanding how the two interact (see chapter 3) and how they should be exploited in evolutionary robotics.

Combining evolution and life-time learning

The ability of animals to adapt during their lifetime may be what most clearly separates them from current machines. In evolutionary robotics, evolved robots with learning abilities could be more able to cope with environmental and morphological changes (e.g., a damage leg) than non-plastic robots (Urzelai and Floreano 2001): evolution would deal with the long-term changes, whereas learning would deal with the short-term ones. In addition, many evolved robots

[1] Extensions of HyperNEAT exist to automatically discover the layout, but we did not use them in our work (Risi and Stanley 2011).

rely on artificial neural networks, which have initially been designed with learning in mind: there is a good fit between evolving neural networks in ER and evolving robots with online learning abilities.

Most papers in this field used on various forms of Hebbian learning (Abbott and Nelson 2000; Urzelai and Floreano 2001). In our work, we used on neuro-modulated Hebbian plasticity because they allow the neural networks to take into account a reward (Soltoggio, Bullinaria, et al. 2008; Soltoggio and Jones 2009; Risi and Stanley 2010). This rule is a classic Hebbian rule which can be activated/deactivated by modulatory neurons, which are special types of neurons. We therefore distinguish two types of neurons: "modulatory neurons" and "standard neurons". Inputs of each neuron are divided into modulatory inputs I_m and standard I_s inputs. The output a_i of a neuron i is then defined as follows:

$$a_i = \varphi_1\Big(\sum_{j \in I_s} w_{ij} a_j + b_i\Big) \tag{1.1}$$

where I_s is the set of standard input neurons of neuron i, a_i its output, b_i its bias, $\varphi_1(x) = \frac{1}{1+\exp(-\lambda x)}$ a sigmoid with its output in $[0, 1]$, w_{ij} the synaptic weight between neurons i and j. Each non-modulatory synaptic weight w_{ij} is modified with regards to the sum of modulatory inputs (m_i, computed from the modulatory neurons in I_m) and a constant coefficient η ($\eta = 0.04$ in our experiments):

$$m_i = \varphi_2\Big(\sum_{j \in I_m} w_{ij} a_j\Big) \tag{1.2}$$

$$\Delta w_{ij} = \eta \cdot m_i \cdot a_i \cdot a_j \tag{1.3}$$

$$w_{ij}(t+\delta t) = \begin{cases} \min(\max(w_{ij}(t) + \Delta w_{ij}, 0), K_{max}) \\ \text{if } w_{ij}(t) \geq 0 \\ \min(\max(w_{ij}(t) - \Delta w_{ij}, K_{min}), K_{\varepsilon}) \\ \text{if } w_{ij}(t) < 0 \end{cases}$$

where $\varphi_2(x) = \frac{2}{1+\exp(-\lambda x)} - 1$ is a sigmoid with its output in $[-1, 1]$ (to allow positive and negative modifications of synaptic weights), K_{max} is the maximum value that a synaptic weight can take (in our experiments, $K_{max} = 30$), K_{min} the minimum (in our experiments, $K_{min} = -30$, and K_{ε} a small constant (in our experiments, $K_{\varepsilon} = -10^-5$). When evolving plastic neural networks, a single mutation can transform a modulatory neuron into a standard neuron, and vice-versa.

Despite its promises (Gruau and Whitley 1993; Nolfi and Floreano 2001; Urzelai and Floreano 2001), mixing evolution and online learning has proven difficult. First, many papers about learning in ER address different challenges while using the same terminology (e.g., "learning", "robustness," and "generalization"), making it difficult to understand the literature. In particular, the distinction between "behavioral robustness" (the robot maintains the same qualitative behavior in spite of environmental/morphological changes; no explicit reward/punishment system is involved) and "reward-based behavioral changes" (the behavior of the robot depends on rewards, which are set by the experimenters) is often unclear (Mouret and Tonelli 2014). There is also a critical difference between evolving a neural network that can use rewards to *switch* between two behaviors, on the one hand, and being able to learn a new behavior in a situation that was not used in the fitness function, on the other hand. Surprisingly, only a handful of papers evaluate the actual *learning abilities* of evolved neural networks in scenarios that were not tested in the fitness function (Chalmers 1990; Tonelli and Mouret 2011; Tonelli and Mouret 2013; Coleman et al. 2014).

Second, crafting a fitness function is especially challenging: to evaluate the fitness of robots that can learn, the experimenter needs to give them the time to learn and test them in many different situations, but this kind of evaluation is prohibitively expensive (many scenarios, each of them having to last for a long time), especially with physical robots. In addition, evolving neural networks with learning abilities appears to be a deceptive problem (Risi, Hughes, et al. 2010; Risi and Stanley 2010) in which simple, non-adaptive neural networks inititally outperforms networks with learning abilities, since the latter are more complex to discover and tune. Novelty search (Lehman and Stanley 2011a) could help mitigating this issue (Risi and Stanley 2010).

Overall, evolution and learning are two adaptive processes that interact with each other; they form a complex system that is hard to tame. The Baldwin effect (Hinton and Nowlan 1987; Dennett 2003) suggests that learning might facilitate evolution by allowing organisms (or robots) to pretest the efficiency of a behavior thanks to their learning abilities, which will create a selective pressure to make it easier to discover the best behaviors, which in turn should result in the inclusion of these behaviors in the genotype (so that it is not required to discover the behavior again and again). Hence the Baldwin effect can smooth a search landscape and accelerate evolution (Hinton and Nowlan 1987). Nonetheless, from a pure computational perspective, the ideal balance between spending time on the fitness function, to give individuals the time to learn, and using more generations, which gives evolution more time to find good solutions, is hard to find. In many cases, it might be more computationnaly efficient to use more generations than using learning to benefit from the Baldwin effect. More generally, evolution and learning are two processes that optimizes a similar quantity – the performance of the individual. In nature, there is a clear difference in time scale (a lifetime versus millions of years), therefore the two processes do not really compete with each other. However, ER experiments last only a few days and often solve problems that could also be solved by learning: given the time scale of current experiments, learning and evolution might not be as complementary as in nature, and they might interact in a different way.

Resilience and adaptation in robotics

Fault tolerance and robustness. The most classic approaches to fault tolerance are based on a diagnosis followed by the activation of a contingency plan (Frank 1990; Visinsky et al. 1994; Koren and Krishna 2007). For instance, if a hexapod robot detects that one of its legs is not reacting as expected, it can drop it and adapt the position of the other legs accordingly (Jakimovski and Maehle 2010; Mostafa et al. 2010). Considerable efforts have been put into automatic diagnosis (Frank 1990; Frank and Köppen-Seliger 1997; Blanke and Schröder 2006) with techniques like fuzzy logic (Isermann 1998), learning algorithms (Polycarpou

and Helmicki 1995), or genetic algorithms (Bongard et al. 2006). Similar techniques have been used to find contingency plans automatically (Polycarpou and Helmicki 1995; Bongard et al. 2006). Diagnosis-based methods undoubtedly proved their usefulness in space, aeronautics and numerous complex systems, but they also lead to systems that are expensive to operate and to design: (1) diagnosis requires both advanced proprioceptive capabilities and reasoning systems, which increase the cost and the complexity of the machine; (2) defining contingency plans increases the time required to design and test a new machine, therefore it also increases the cost. Importantly, these approaches require the identification of the faulty subsystems and a procedure to bypass them, whereas both operations are difficult for many kinds of faults. For example, many mechanical failures (e.g. a broken leg) are difficult to diagnose because there is no direct way to observe them (most robots are unable to "see" themselves and the image processing required for such diagnosis is especially difficult).

Another classic approach to fault tolerance is to employ robust controllers that can work in spite of damaged sensors or hardware inefficiencies (e.g., Goldberg and Chen 2001; Caccavale and Villani 2002; Qu et al. 2003; Lin and Chen 2007). Such controllers usually do not require diagnosing the damage, but this advantage is tempered by the need to integrate the reaction to all faults in a single controller. They may also encounter a situation for which they are not robust, typically when their designers did not considered it beforehand.

Learning in robotics Discovering a new behavior in an unexpected situation is a particular case of *learning* a new behavior, a question that generates an abundant literature in artificial intelligence since its beginnings (Turing 1950). We are here interested in reinforcement learning algorithms because we consider scenarios in which evaluating the performance of a behavior is possible but the optimal behavior is unknown. However, classic reinforcement learning algorithms are primarily designed for discrete states and discrete actions (Sutton and Barto 1998; Kober and Peters 2013), whereas autonomous robots have to solve many continuous problems, in particular for motor control. The prevalent alternative in robotics is to view reinforcement learning as an optimization of the parameters of a policy (a controller) (Kober and Peters 2013; Stulp and Sigaud 2013). This optimization problem can be tackled with gradient-based methods (Kohl and Stone 2004; Kober and Peters 2013), evolutionary algorithms (Mouret and Doncieux 2012a; Bongard 2013; Koos, Cully, et al. 2013; Doncieux, Bredeche, et al. 2015), or other optimization methods like Bayesian optimization (Lizotte et al. 2007; Mockus 2013; Calandra et al. 2014).

Most policy-search methods (Kober and Peters 2013) iteratively optimize locally around existing policies by computing changes in the policy parameters that will increase the expected reward. They essentially differ in the way the policy is updated, with techniques ranging from gradient estimation with finite differences (Kohl and Stone 2004) to advanced methods such as the Powell method (Sproewitz et al. 2008). To make learning tractable, most of the successful reinforcement learning experiments include prior knowledge through demonstrations (Kober and Peters 2013), which removes the need for global optimization. They also employ effective, but often ad-hoc, representations that minimize the number of policy parameters (Kober and Peters 2013).

Gradient-based policy search algorithms have been successfully applied to many motor control scenarios, for instance to teach a robot to play the ball-in-the-cup game by exploiting human demonstrations and representing the policy with dynamic motor primitives (Kober and Peters 2013). They have also been applied to locomotion tasks for quadruped (tables 1.1 and 1.2) (Kimura et al. 2001; Kohl and Stone 2004) and biped robots (Tedrake et al. 2005), but they require numerous evaluations on the robot (often more than 1000 trials in a few hours, see table 1.1).

Evolutionary algorithms have also been employed to learn a policy (Grefenstette et al. 1999; Heidrich-Meisner and Igel 2009; Kober and Peters 2013). They are less prone to local optima than most policy-search algorithms and they can optimize structures (neural networks, fuzzy rules, programs, ...) (Mouret and Doncieux 2012a; Bongard 2013). However, evolutionary algorithms are slower than policy gradient methods: Learning experiments with evolutionary algorithms are reported to require many hundreds of trials on the robot and to last from two to tens of hours (tables 1.1 and 1.2).

To minimize learning time, many authors proposed to learn or evolve using a physical simulator, then transfer the learned behavior to the real robot. Unfortunately, it is now widely documented that behaviors learned in simulation very often do not work well on the physical robot (Jakobi 1997; Mouret and Doncieux 2012b; Bongard 2013; Koos, Mouret, et al. 2013). This issue is called the reality gap. Our group recently introduced a new method, called the transferability approach (Mouret and Doncieux 2012b; Koos, Cully, et al. 2013; Koos, Mouret, et al. 2013) to cross this gap. This method combines machine learning (a support vector machine), experiments with the physical robot, and a multi-objective evolutionary algorithm to focus the search on behaviors that are accurately simulated. It has been successfully applied to quadruped locomotion (Mouret and Doncieux 2012b; Koos, Mouret, et al. 2013) (10 physical trials), hexapod locomotion (25 physical trials) (Koos, Cully, et al. 2013), T-maze navigation (Koos, Mouret, et al. 2013) (10 physical trials) and humanoid locomotion (Oliveira et al. 2013).

As a illustration, we surveyed the methods that have been used for learning locomotion in legged robots (Table 1.1). For this simple task, learning times are between 20 minutes and 25 hours. Orders of magnitude are similar for other tasks. For instance, about 100 trials (episodes) are needed to learn to put a ball in a cup when starting with demonstrations (Kober and Peters 2013).

Learning/evolving for damage recovery. Learning algorithms have been used in a few studies dedicated to damage recovery, in particular on a snake-like robot with a damaged body (Mahdavi and Bentley 2003) (about 600 trials/10 hours) and on a quadrupedal robot that breaks one of its leg (Berenson et al. 2005) (about 670 trials/2 hours). A recent experiment showed adaptation to damage in a modular robot in about 15 minutes, however the search space is small (each parameter can take only two values, and there are only 4 parameters in the experiments performed on the real robots) (Christensen et al. 2013).

Table 1.1 How long many previous robot learning algorithms take to run. While comparisons between these algorithms are difficult because they vary substantially in their objective, the size of the search space, and the robot they were tested on, we nonetheless can see that learning times are rarely below 20 minutes, and often take hours.

approach/article	starting behavior *	learning time	robot	DOFs†	param.‡	reward¶
Policy Gradient Methods						
Kimura et al. 2001	n/a	80 min.	quadruped	8	72	internal
Kohl and Stone 2004	walking	3 h	quadruped	12	12	external
Tedrake et al. 2005	standing	20 min.	biped	2	46	internal
Geng et al. 2006	n/a	4-5 min.	bipedal	4	2	internal
Christensen et al. 2013	n/a	15+ min	quadruped	8	8	external
Evolutionary Algorithm						
Chernova and Veloso 2004	random	5 h	quadruped	12	54	external
Hornby, Takamura, et al. 2005	non-falling	25h	quadruped	19	21	internal
Barfoot et al. 2006	random	10 h	hexapod	12	135	external
Yosinski et al. 2011	random	2 h	quadruped	9	5	external
Bongard et al. 2006[1]	random	4 h	hexapod	12(18)	30	internal
Koos, Cully, et al. 2013	random	20 min.	hexapod	12(18)	24	internal
Bayesian optimization						
Lizotte et al. 2007	center§	2h	quadruped	12	15	internal
Calandra et al. 2014	random	46 min.	biped	4	8	external
Others						
Weingarten et al. 2004 [2]	walking	$>$ 15 h	hexapod (Rhex)	6	8	external
Sproewitz et al. 2008 [3]	random	60 min.	quadruped	8	5	external
Hemker et al. 2009 [4]	walking	3-4 h	biped	24	5	external
Barfoot et al. 2006 [5]	random	1h	hexapod	12	135	external
Erden and Leblebicioğlu 2008 [6]	standing	15-25min.	hexapod	18	n/a	internal

*Behavior used to initialize the learning algorithm.

† DOFs: number of controlled degrees of freedom.

‡ param: number of learned control parameters.

§ center: center of the search space.

¶ reward: "internal" means that the reward used by the learning algorithm is/can be computed onboard by the robot; "external" means that an external tracking system is used (e.g. camera or motion capture system).

[1] The authors do not provide time information, reported values come from the implementation of Koos, Cully, et al. (2013).

[2] Nelder-Mead descent. [3] Powell method. [4] DACE (Design and Analysis of Computer Experiments).

[5] Multi-agent reinforcement learning. [6] Free-State generation with reinforcement learning.

All policy gradient and evolutionary algorithms spend most of their running time in evaluating the quality of controllers by testing them on the target robot. Since, contrary to simulation, reality cannot be sped up, their running time can only be improved by finding strategies to evaluate fewer candidate solutions on the robot. In their "starfish robot" project, Bongard et al. (Bongard et al. 2006) mixed evolutionary algorithms, simulation, and actions on the robot to recover from physical damage. In their algorithm, the robot chooses simple motor actions and measure their consequences to update a simulator of the robot so that it reflects the damage. This updated simulator is then used to find a new policy with an evolutionary algorithm. Our own tests with Bongard's algorithm on a hexapod (Koos, Cully, et al. 2013) revealed that (1) several hours are required to find a good model (i.e. the right diagnosis) and (2) the algorithm often fails to find the right model. In addition, in the original Bongard et al's paper (Bongard et al. 2006), only half of the experiments managed to find the right model.

CONCLUSION

This brief state-of-the-art reveals several open issues and promising leads. For evolutionary biology and evolutionary intelligence:

- modularity, hierarchy, and regularity are three important keys to understand what makes natural organisms so evolvable;
- the evolution of modularity, hierarchy, and regularity are likely to involve both specific genotype-phenotype maps and specific selective pressures;
- CPPN-based genotype-phenotype maps are good candidates to model genotype-phenotype map in an abstract way;
- online-learning in evolved neural network can rely on neuro-modulated Hebbian plasticity.

For robotics and machine learning:

- the traditional approach for adapting to unforeseen situations is to first perform a diagnosis and, second, to exploit it to select a new behavior;
- reinforcement learning algorithms can provide a shortcut because they do not require a diagnosis to find a behavior that work in the new conditions; this is has the potential to make this learning-based methods simpler and more robust than traditional approaches;
- the prevalent method for reinforcement learning in robotics is called *direct policy search*, which is close to continuous optimization;
- current learning algorithms take 15+ minutes even for relatively small search spaces (e.g. 6-12 parameters, requiring 15-30 minutes); algorithms without these lim-

Table 1.2 How long many previous robot damage recovery algorithms take to run. While comparisons between these algorithms are difficult because they vary substantially in their objective, the size of the search space, and the robot they were tested on, we nonetheless can see that damage recovery times are rarely below 20 minutes, and often take hours.

approach/article	starting behavior *	learning time	robot	DOFs†	param.‡	reward¶
Policy Gradient Methods						
Christensen et al. 2013	n/a	15+ min	quadruped	8	8	external
Evolutionary Algorithm						
Berenson et al. 2005	random	2 h	quadruped	8	36	external
Mahdavi and Bentley 2006	random	10 h	snake	12	1152	external
Bongard et al. 2006[1]	random	4 h	hexapod	12(18)	30	internal
Koos, Cully, et al. 2013	random	20 min.	hexapod	12(18)	24	internal
Bayesian optimization						
Reinforcement learning						
Erden and Leblebicioğlu 2008 [2]	standing	15-25min.	hexapod	18	n/a	internal

*Behavior used to initialize the learning algorithm.

† DOFs: number of controlled degrees of freedom.

‡ param: number of learned control parameters.

¶ reward: "internal" means that the reward used by the learning algorithm is/can be computed onboard by the robot; "external" means that an external tracking system is used (e.g. camera or motion capture system).

[1] The original authors do not provide time information, reported values come from the implementation of Koos, Cully, et al. (2013).

[2] Free-State generation with reinforcement learning.

itations take several hours.

Evolving plastic neural networks

Scientific context. Evolution is a powerful adaptive process in nature, but behavioral adaptation involves many other mechanisms and, in particular, lifetime learning. In term of adaptive abilities for artificial agents, it would be useful to combine the slow paced, but highly exploratory, adaptation provided by evolution with the faster, but more constrained, adaptation provided by lifetime learning.

These two processes are inevitably interleaved because they both improve the ability of individuals to survive in their environment: many behavioral adaptations can either be the result of lifetime learning or or the result of evolution. However, evolution and life-time learning are mostly studied independently, be it in biology (neuroscience vs evolutionary biology) or in artificial intelligence (machine learning vs evolutionary optimization).

In this chapter, we tackle a question at the intersection of evolution and learning: how can evolution lead to individuals that can learn in many different situations? More precisely, how can evolved neural networks learn in situations for which they have never been directly selected? Or, with a machine learning point of view: how can evolution avoid the pitfalls of overfitting the learning mechanism so that it can only work in a specific subset of situations?

One hypothesis is that general learning abilities evolve because there is an explicit pressure to be capable of surviving in varying environments. Nevertheless, this pressure might not fully explain the ubiquity of general learning abilities in animals. We here explore another hypothesis, which is that the developmental process of neural networks makes general learning abilities more likely than specific learning abilities (i.e., abilities to learn only in situations that agents encounter during their lifetime).

Using a simple operant conditioning task and a classic evolutionary algorithm, we compare three ways to encode plastic neural networks: a direct encoding, a developmental encoding inspired by computational neuroscience models, and a developmental encoding inspired by morphogen gradients (similar to HyperNEAT). Our results suggest that using a developmental encoding improves the learning abilities of evolved plastic neural networks. Complementary experiments reveal that this result is likely the consequence of the bias of developmental encodings towards regular structures: (1) in our, experimental setup, encodings that tend to produce more regular networks yield networks with better general learning abilities; (2) whatever the encoding is, networks that are the more regular are statistically those that have the best learning abilities.

Human context. The main article (in PLoS One) is the principal contributions of the PhD of Paul Tonelli, who I co-supervised (co-supervision with S. Doncieux - JBM: 75%, SD:25%). It is an extended version of a paper which won the "best paper award" of the "Generative and Developmental Systems" track at the GECCO'2011 conference.

Main articles:

- Tonelli P., and Mouret, J.-B. (2013). *On the Relationships between Generative Encodings, Regularity, and Learning Abilities when Evolving Plastic Artificial Neural Networks.* PLoS ONE 8(11): e79138. doi:10.1371/journal.pone.0079138
- Tonelli, P., and Mouret, J.-B. (2011) *On the Relationships between Synaptic Plasticity and Generative Systems*. Proceedings of the 13th Annual Conference on Genetic and Evolutionary Computation (GECCO).

Related articles:

- Ellefsen, K. O., Mouret, J.-B., and Clune, J. (2015). *Neural modularity helps organisms evolve to learn new skills without forgetting old skills.* Plos Computational Biology.
- Mouret, J.-B. and Tonelli, P. (2014). *Artificial Evolution of Plastic Neural Networks: a few Key Concepts.* Growing Adaptive Machines: combining Development and Learning in Artificial Neural Networks, Studies in Computational Intelligence (Springer), publisher. Vol 557 Pages 251-261.

Other contributors:

- Paul Tonelli (PhD student)

Author contributions (for the main article): performed the experiments: PT; wrote the code: PT and JBM; analyzed the results: JBM; wrote the paper: JBM

It is commonly believed that the key for understanding the evolution of large and organized neural networks is the developmental process that links genes to nervous systems (Hornby and Pollack 2002; Stanley and Miikkulainen 2003; Pfeifer and Bongard 2007; Clune and Lipson 2011). The genotype of animals does not encode each synapse individually, it instead describes rules of development that are executed multiple times to give birth to networks with regular patterns of connection. Influenced by this concept, many researchers proposed *artificial developmental systems* with diverse inspirations including chemical gradients (Stanley, D'Ambrosio, et al. 2009; Clune and Lipson 2011), gene regulatory networks (Bongard 2002; Mattiussi and Floreano 2007), cell divisions (Gruau and Whitley 1993), computational neuroscience models (Mouret, Doncieux, and Girard 2010) and L-systems (Hornby and Pollack 2002).

Nonetheless, most networks evolved so far with developmental systems cannot change during the "lifetime" of the controlled agent, whereas animal nervous systems are continuously changing to enable on-line behavioral adaptation and learning (Abbott and Nelson 2000). The basis of most of these changes seems to be provided by *synaptic plasticity*, that is, by the ability of synapses to strengthen or weaken over time (Hebb 1949; Abbott and Nelson 2000). Several papers report experiments in which neu-

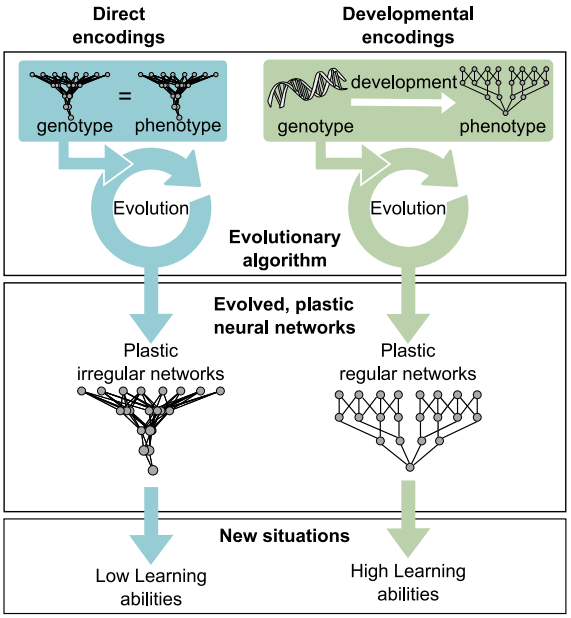


Figure 2.1. Main hypothesis. Using developmental encodings should facilitate the evolution of plastic ANNs with high learning abilities.

ral networks with synaptic plasticity are evolved (Gruau and Whitley 1993; Urzelai and Floreano 2001; Niv et al. 2002; Soltoggio, Dürr, et al. 2007; Floreano et al. 2008; Soltoggio, Bullinaria, et al. 2008; Soltoggio and Jones 2009; Risi, Hughes, et al. 2010; Risi and Stanley 2010). Yet, only a handful of them use developmental systems (Gruau and Whitley 1993; Soltoggio, Dürr, et al. 2007; Risi and Stanley 2010).

The present chapter shows that these two topics–developmental systems and synaptic plasticity–are actually deeply connected.

One of the main challenge when designing ANNs with learning abilities is to make them capable of learning in a large class of situations, that is, designing them so they can adapt their behavior to maximize a reward signal (or minimize an error) in as many situations as possible. For instance, it has been famously shown that single layer perceptrons are only capable of learning linearly separable patterns (Minsky and Papert 1987), whereas multi-layer perceptrons can learn any non-linear function (provided enough neurons are available) (Cybenko 1989). Single-layer perceptrons therefore possess lower learning abilities than multi-layer perceptrons: their architecture critically constrains what they can learn. When artificial evolution is used to design a plastic ANN, the topology of the networks is the result of the interactions between the fitness function, the encoding and the associated variation operators. As a consequence, the encoding and the fitness function have to be carefully crafted so that plastic neural networks are able to learn in as many situations as possible and, specifically, in situations that are not explicitly tested in the fitness function.

The most classic approach is to design a fitness function that tests each neural network in several test cases and rewards individuals that successfully adapt their behavior to each of them. To ensure that networks possess general learning abilities, it is then required to assess their abilities to learn in a new set of test cases, that is, test cases that have never been encountered by the evolutionary process (Chalmers 1990). The success of this "episodic fitness" approach relies on the assumption that if enough test cases are used, then it should become easier for the evolutionary process to design a generic structure than a specialized one.

Unfortunately, even in simplistic and constrained toy problems, the reported experiments show that many test cases need to be included in the fitness function to obtain general learning abilities[1] (e.g. 10 to 20 test cases in (Chalmers 1990)). For more complex problems, one can expect an exponential growth in the number of required test cases, because the number of possible test cases grows exponentially with the number of inputs/outputs. This approach is, therefore, unlikely to scale-up to life-like neural-networks.

This is where developmental systems have a role to play. These systems evolve short descriptions of large structures by exploiting regularities observed in Nature, such as repetition of useful sub-parts, symmetries, and symmetries with variation (Hornby 2005; Stanley, D'Ambrosio, et al. 2009). They more easily describe regular structures than irregular ones, because the former can be described by a few general rules whereas the latter require describing either each element, or a list of exceptions to general rules. As a consequence, developmental systems bias the search space towards regular structures (Clune and Lipson 2011). *We here propose that this bias towards regularity is critical to evolve plastic neural networks that can learn in a large variety of situations.* Intuitively, this bias makes it more likely to obtain generic networks that apply the same learning rules to whole sets of inputs instead of networks that are finely-tuned to only solve the test cases used in the fitness function. A direct consequence is that *using developmental systems to evolve plastic neural networks should facilitate the evolution of plastic ANNs with general learning abilities.*

METHODS

This hypothesis is tested using a simulated "Skinner box" (Figure 2.2), a classic experimental setup for operant conditioning in which a caged animal must learn to associate stimuli (e.g. lights) to actions (e.g. pushing a lever). If the animal executes the correct action, it is rewarded (e.g. by some food); if it chooses the wrong one, it is punished (e.g. by an electric shock). There is no delay in the reward, so there is no credit assignement problem (Sutton and Barto 1998). We consider only one-to-one associations so that, for each simple stimulus (each light), there is a different action to perform. Four stimuli and four actions are used; there are therefore 256 possible sets of stimulus/action ($4^4 = 256$; see Appendix S1 for the list of possible association sets). We formalize the stimulus/action associations using the concept of association sets:

Definition 1 (Association) *An association is a pair* (I, O) *of input/output that leads to the maximum positive reward. In our system* ($n = 4$*: 4 inputs, 4 outputs),* $(1, B)$ *is an association*

[1] It should be emphasized that many authors do not test whether the plastic ANNs they evolve can learn in test cases *that have not been encountered during the evolutionary process*. For instance, (Soltoggio, Bullinaria, et al. 2008; Risi, Hughes, et al. 2010; Risi and Stanley 2011) don't assess how evolved neural networks can cope with an unknown situation; counter-examples are (Urzelai and Floreano 2001) and (Chalmers 1990).

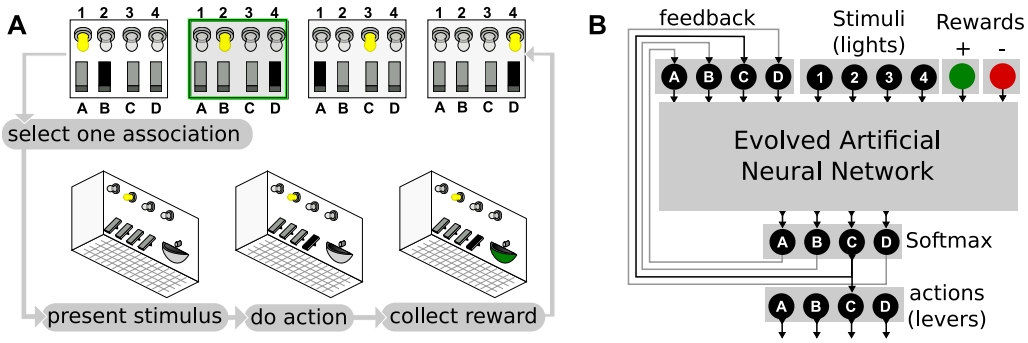


Figure 2.2. **A. Concept of the "Skinner box".** A caged animal must learn to associate stimuli (here lights) to actions (here pushing a lever). The experimenter selects a stimulus/action association, presents it to the animal, record the action, and gives the reward to the animal. The experimenter then chooses another association in the association set and starts the cycle again. The association set is learned once the animal associates the right action to each stimulus. **B. Formalization of the Skinner box as a task for an artificial neural network**. Each stimulus is an input of the neural network. Positive and negative rewards are two additional inputs. The output is selected according to a softmax function (Methods) and the result of the softmax is looped back to the input layer.

that means that the agent must push the B lever when light 1 *is on.*

Definition 2 (Association set) *An association set* $A = \{(I_1, K_1), \cdots, (I_n, K_n)\}$ *is a list of associations that covers all the* n *possible inputs. For instance, the list of associations* $\{(1, B), (2, C), (3, D), (4, A)\}$ *is an association set in our system (*$n = 4$*: 4 inputs, 4 outputs). Several inputs can be associated to the same output. For instance, the association set* $\{(1, A), (2, A), (3, A), (4, A)\}$ *is also valid in our system.*

Definition 3 (Global training set) *The global training set, called* $\mathbb{G}$*, is the the set of all the possible association sets of an experimental setup. In our system, there are* 4 *possible outputs and* 4 *possible inputs (*$n = 4$*), therefore the size of* $\mathbb{G}$ *is* 256 *(*$|\mathbb{G}| = 4^4 = 256$*; the complete list of assotiation sets is available in Appendix S1). The ideal plastic network should be able to learn every association sets of* $\mathbb{G}$*.*

The fitness function (Methods) assesses the ability to learn a subset of the global training set, called the *evolutionary training set*:

Definition 4 (Evolutionary training set) *The evolutionary training set, called* $\mathbb{E}$*, is the set of the association sets used in the fitness function.*

$\mathbb{E}$ is included in $\mathbb{G}$; it does not change during an experiment. Depending of the experiment, the size of $\mathbb{E}$ varies between 1 and 7. The elements of $\mathbb{E}$ have been chosen at random.

The fitness function is normalized by the size of $\mathbb{E}$, so that it actually corresponds to the the number of successfully learned sets divided by $|\mathbb{E}|$. After each evolution experiment, we assess the ability of the network with the best fitness score to learn every possible association set, that is, we evaluate the fitness function on the global training set. We call the success rate of this test the *General Learning Abilities score (GLA score)*. This score reflects how well networks that are selected for their capacity to learn a few association sets are able to learn association sets that have not been encountered during evolution.

The evolved ANNs (Figure 2.2, B) have one input for each possible stimulus (i.e., 4 stimuli inputs), one input for positive rewards and one input for negative rewards. They have 4 outputs, each of them representing the probability of choosing each action. The final action is selected thanks to a "softmax" function that randomly selects an action according to a distribution that gives a higher probability to actions that corresponds to high output values distribution (Sutton and Barto 1998) (Methods). In effect, the neural network can activate any combination of the four available outputs and the softmax function makes sure that only one action is chosen at a time (Figure 2.2, B). Only one light (input) is activated at a time.

Plasticity is implemented in the neural networks using *neuro-modulated Hebbian plasticity* (Abbott and Nelson 2000; Floreano et al. 2008; Soltoggio, Bullinaria, et al. 2008; Risi, Hughes, et al. 2010) (Methods). In this model, neurons are of two kinds, "standard" and "modulatory"; the strength of connection between each pair of neurons is modified using a classic Hebbian rule that is activated only if the sum of inputs from modulatory neurons is above an evolved threshold.

For each association of $\mathbb{E}$, the fitness function first presents the stimuli to the neural network for a few timesteps (Methods). Once the final output is computed by the softmax, it is copied to the input layer (feedback inputs). The reward input (positive or negative) is set at the same time. Such feedback loops are often present in computational models of cortex-basal ganglia-thalamus-cortex loops for action selection (Houk et al. 1995; Frank and Claus 2006; Girard et al. 2008) and are implicit in actor-critic models of reinforcement learning (Sutton and Barto 1998). The neural network is then simulated for a few more timesteps (Methods). It is expected that the evolutionary process will connect one or several modulatory neurons to the reward input and that the ANNs will exploit the copied outputs to strengthen/weaken the connections that correspond to the action that has actually been performed. Nonetheless, it must be emphasized that weight changes can occur at any time, including during the first step of the evaluation of the ANN. Only the topology and the synaptic weights of the ANNs, which are designed by evolution, determine when and how synaptic weights change.

The ANNs that solve this task may seem trivial at first sight. However, the evolutionary process needs to add at least one modulatory neuron (inputs cannot be modulatory in our system) and we never found any solution with less than two hidden neurons (one of them being modulatory).

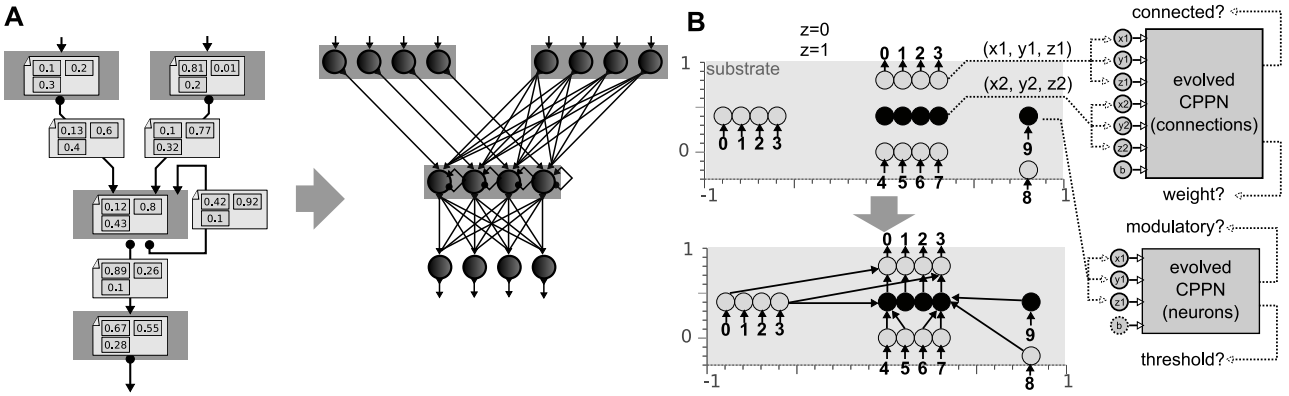


Figure 2.3. **A. Principle of the map-based, developmental encoding.** The neural network is encoded as a labeled graph (left), which is developped to a graph of maps according to the labels (right). (Methods). **B. Principle of the HNN encoding (minimal HyperNEAT).** Neurons are placed in a 3D substrate (top). To know whether two neurons are connected and the synaptic weight of each connection, a Compositional Pattern Producing Network (CPPN) is queried using the 3D coordinates of the two neurons (Methods). This CPPN is evolved using a direct encoding. To know the parameters of each node (neuron type and threshold value), a second CPPN is queried with the 3D coordinates of the neuron (Methods).

Essentially, the challenge raised by this task is to discover learning rules that allow the ANN to exploit a reward to strengthen and weaken the right connections. Typical solutions require three main "discoveries": (1) identifying and correctly connecting the reward inputs, (2) gating the reward with the softmax choice to modify only the connections corresponding to the chosen action, and (3) applying the resulting reinforcement to a link between the inputs and the output.

The topology and the parameters of evolved ANNs are encoded with three encodings (Floreano et al. 2008), with three different levels of expected regularity (Figure 2.3). The first encoding, called the map-based encoding (Mouret, Doncieux, and Girard 2010) (Methods), is inspired by computational neuroscience models in which ANNs are described as graph of single neurons and *neural maps* (spatially-organized identical neurons) that are connected with a few possible connection schemes (usually only one-to-one and one-to-all) (Gurney et al. 2001; Rougier and Vitay 2006; Girard et al. 2008). This encoding produces very regular neural networks because it has to treat each neuron in a map in the exact same way as the other neurons of the same map. The second encoding is a simplified version HyperNEAT (Stanley, D'Ambrosio, et al. 2009), called HNN, for Hyper Neural Network (Methods). HyperNEAT-like encodings are developmental encodings in which morphogen gradients are described as feed-forward networks of mathematical functions that operate in a Cartesian space. This indirect approach allows them to encode large networks with Nature-like connection patterns (symmetry, symmetry with variations, repetition, etc.). The last encoding is a classic direct encoding in which evolution directly acts on the structure and the parameters of the ANN (Methods). This encoding has no bias to produce regular networks.

To understand the relationship between encodings, regularity and learning abilities, we have to assess the regularity of evolved ANNs. According to Lipson (Lipson 2007), regularity is the compressibility of the description of the structure. Regrettably, this value is not computable (Li and Vitányi 2008) and, to our knowledge, there exists no well-recognized approximation for weighted, directed graphs. The few algorithms designed to compress the graph structure are greedy approximations that only work well for sparse, undirected labeled graphs (Peshkin 2007; Hayashida and Akutsu 2010). We follow another method to estimate the regularity of networks: counting the number of symmetry axes (Mowshowitz 1968a,b; Zenil et al. 2013). A graph has an axis of symmetry when two groups of nodes can be swapped without modifying the graph, that is, when there is a repetitive, structural pattern. More axes of symmetry means a better compression because the two groups need to be described only once (Mowshowitz 1968a,b; Zenil et al. 2013). In graph theory, this kind of symmetry is called an automorphism and fast algorithms exist to count them (McKay 1981; Junttila and Kaski 2007; Katebi et al. 2012) (Methods). Figure 2.4 reports the number of automorphisms of a few example networks.

Networks are evolved using the classic multi-objective evolutionary algorithm NSGA-II (Deb 2001; Deb et al. 2002). Two objectives are optimized: the fitness of networks (Methods) and a behavioral novelty objective (Soltoggio and Jones 2009; Risi, Hughes, et al. 2010; Lehman and Stanley 2011a; Mouret 2011b; Mouret and Doncieux 2012b), to mitigate premature convergence (Methods). These two objectives are optimized during a maximum of 4000 generations of 400 individuals. Experiment are stopped as soon as the best individual of the population reaches a perfect fitness value on the evolutionary training set. At the end of each experiment, the novelty objective is discarded and we consider that the best individual is the one with the best fitness value.

We perform 7 series of independent experiments by varying the size of the evolutionary learning set from 1 to 7 (i.e., $|\mathbb{E} = 1, \cdots, 7|$). For each series, the three investigated encodings are tested (direct encoding, map-based encoding and HNN encoding). Each experiment is replicated 30 times to obtain statistics. We therefore launch a total of $3 \times 7 \times 30 = 630$ experiments, each one lasting between 1 and 8 hours on our computers (Intel Xeon E5520@2.27GHz) depending on the time required to converge and the size of the evolutionary training set. Because of this large computational time, we were not able to extend our experiments to harder problems, for instance with more inputs/outputs.

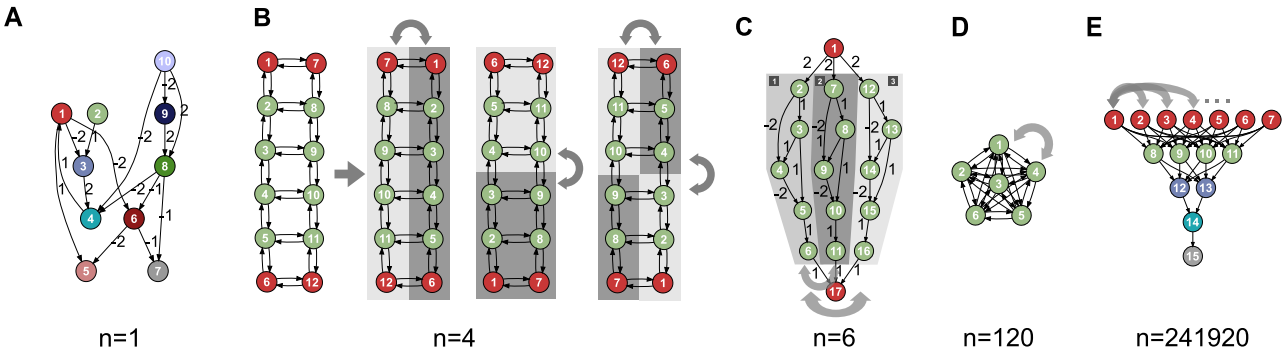


Figure 2.4. **Examples of networks and corresponding number of automorphisms.** (Colors are only here to help seing the symmetry axes, they have no particular meaning). **A.** A random network typically has 1 automorphism (itself). **B.** The central pattern generator of the lamprey (Ijspeert, Crespi, et al. 2007) has 4 automorphisms (2×2) because it has two axial symmetries: top-down and left-right. The structure of the graph implies that the vertex orderings $\{1,2,3,4,5,6,7,8,9,10,11,12\}$, $\{7,8,9,10,11,12,1,2,3,4,5,6\}$, $\{6,5,4,3,2,1,12,11,10,9,8,7\}$ and $\{12,11,10,9,8,7,6,5,4,3,2,1\}$ all lead to the same connectivity matrix. **C.** This network has 6 automorphisms ($3!$) because modules marked $1, 2$ and 3 can be swapped without changing the connectivity of the network. **D.** This fully connected network with uniform synaptic weights has 120 automorphisms ($5!$) because each of its nodes can be swapped with any other node. **E.** This multi-layer perceptron with uniform synaptic weights has 241920 automorphisms($7! \times 4! \times 2!$) because each node of each layer can be swapped with any other node of the same layer.

RESULTS

For each encoding, we compute the GLA score of networks with a perfect fitness on the evolutionary training set and we plot it as a function of the size of the evolutionary training set.

The results show a clear difference in the GLA scores obtained with each encoding (Figure 2.5, A). With a direct encoding, the GLA score grows linearly with the size of the evolutionary training set, which is consistent with previous results (Chalmers 1990), and the GLA scores obtained with small values of $|\mathbb{E}|$ are statistically different from those obtained with larger values (e.g., 1 versus 7: $p = 4 \times 10^{-4}$; 4 versus 7, $p = 2 \times 10^{-3}$, 3 versus 6, $p = 2 \times 10^{-3}$; unless otherwise specified, the statistical test in this chapter is the Mann-Whitney U-test). With the direct encoding, using a fitness that tests at least 6 associations sets ($|\mathbb{E}| > 5$) is required to obtain networks with a GLA-score similar to the one reached with the map-based encoding with only 2 association tests ($p = 0.8$). The HNN encoding appears as a trade-off between the direct encoding and the map-based encoding: for each value of $|\mathbb{E}|$, the GLA score obtained with HNN is consistently higher than the one obtained with the direct encoding, yet it is lower than the one reached with the map-based encoding (for 2, 3 or 4 association sets, HNN versus direct encoding, $p < 0.03$; for 1, 2, 3 or 5 association sets, HNN versus map-based encoding, $p < 6 \times 10^{-3}$)[2].

As expected, each encoding leads to different levels of regularity, and increasing the number of association sets used in the fitness function increases the regularity of evolved neural networks (Figure 2.5, B). All the networks evolved with the map-based encoding are regular: they all have at least one symmetry axis. The HNN encoding also leads to many networks with at least one symmetry axis (from 80% to 100%), whereas the direct encoding leads to substantially fewer regular networks (from 20% to 70%, depending on $|\mathbb{E}|$). These numbers vary with the size of $\mathbb{E}$. With the HNN encoding, three association sets are needed to obtain 100% of regular networks; with the direct encoding, the number of regular networks grows from 20%, when one association set is used during evolution ($|\mathbb{E}| = 1$), to 60-70% when more than 6 association sets are used ($|\mathbb{E}| > 5$).

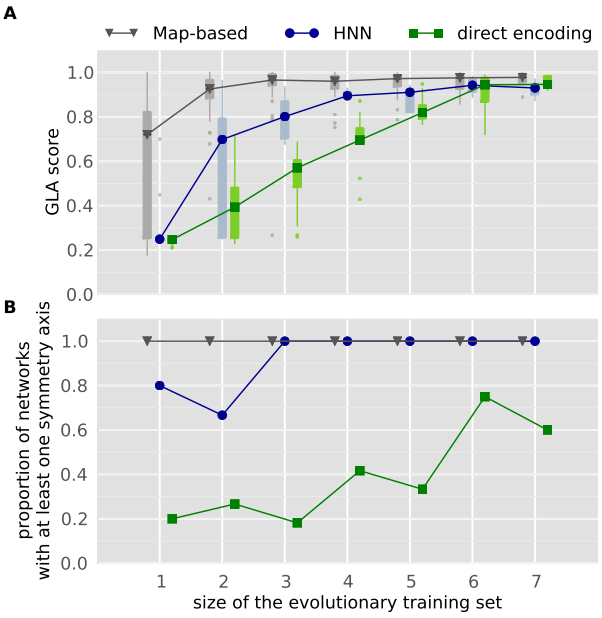


Figure 2.5. **Relationship between encodings, general learning abilities and the size of the evolutionary training set ($|\mathbb{E}|$). A.** Generative encodings yield plastic ANNs with better general learning abilities than those evolved with a direct encoding. Morever, increasing the size of $\mathbb{E}$ increases the general learning abilities. Each box extends from the lower to upper quartile values of the data, with a symbol at the median. Whiskers extend to the most extreme data point within $1.5 \times IQR$, where IQR is the interquartile range. Flier points (outliers) are those past the end of the whiskers. X-values are shifted for the map-based encoding and the direct encoding in order to make the figure readable. **B.** Generative encodings yield more regular networks than a direct encoding, and increasing the size of $\mathbb{E}$ increases the regularity of evolved networks.

To further understand this result, we plot the network with the best learning abilities for each encoding and each size of the evolutionary learning set (figure 2.7). We observe the same overall link between learning abilities and regularity as on figure 2.5, but some networks have good learning abilities with only a few automorphisms, like the network

[2]With 4 association sets and the HNN encoding, there are not enough networks with a perfect fitness score to perform a statistical analysis.

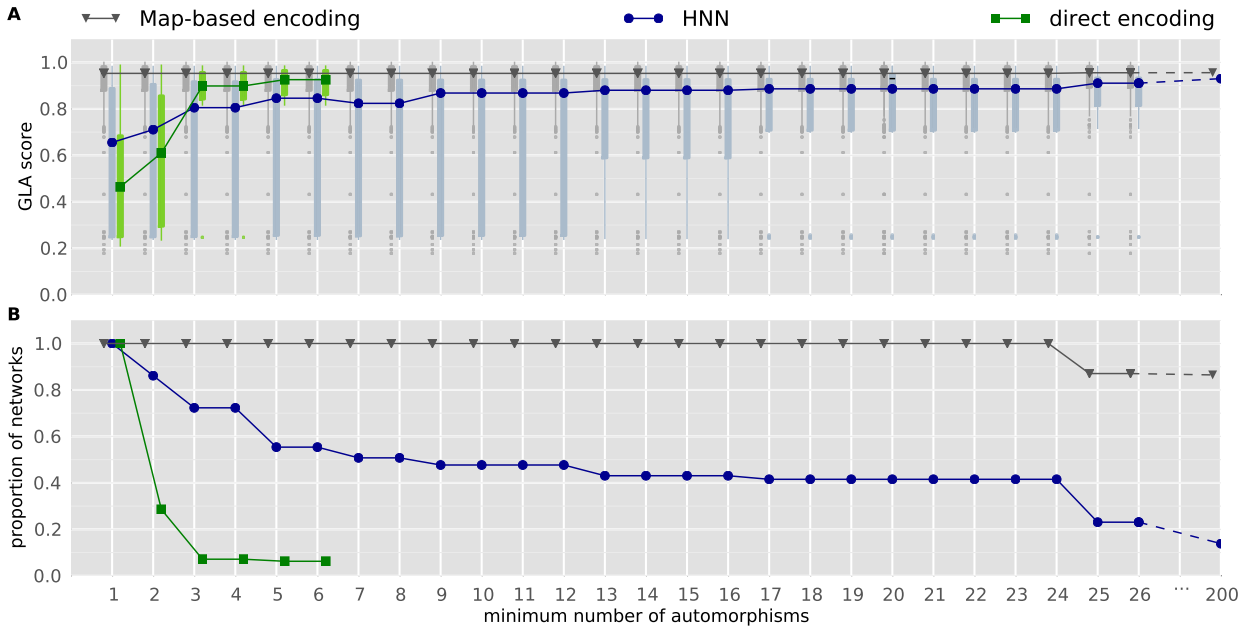


Figure 2.6. Relationship between regularity and general learning abilities. Data are from the same experiments as Figure 2.5. The "minimum number of automorphims" means that if, for example, a network has 4 automorphisms, it is included in columns 1,2,3 and 4. X-values are shifted for the map-based encoding and the direct encoding in order to make the figure readable. **A.** The more automorphisms a network has, the more likely it is to have good general learning abilities (GLA score). Each box extends from the lower to upper quartile values of the data, with a symbol at the median. Whiskers extend to the most extreme data point within $1.5 \times IQR$, where IQR is the interquartile range. Flier points (outliers) are those past the end of the whiskers. **B.** 7% of networks evolved with a direct encoding have more than 3 autormorphisms. 72% of those evolved with HNN have more than 3 automorphisms. 86% of networks evolved with the map-based encoding have more than 200 autmorphisms; 100% of them have at least 10 automorphisms.

evolved with a direct encoding and 6 association sets (GLA score of 0.99, 2 automorphisms). This result is possible because nothing encourages a directly encoded network to duplicate the same sub-structure several times: it may be sometimes easier to either re-invent 4 times the same function but with slight changes, or to design an integrated solution that relies on only one complex structure. This particular network seems to use a centralized structure with only one modulatory neuron that modulates all the plastic connections of the network. Conversely, some regular networks have a low GLA score, such as the network evolved with HNN and one association set (GLA score of 0.70, 24 automorphisms). There is no paradox in this result: the regularities can be at the wrong place to lead to high-learning abilities.

Whatever the encoding and the size of $\mathbb{E}$ are, networks with the best learning abilities are those that are the most regular (Figure 2.6, A; this figure use the same data as Figure 2.5). Hence, among networks evolved with the direct encoding, those that have at least 2 automorphisms (one axis of symmetry) have a better GLA score than those that have no automorphism ($p = 6 \times 10^{-3}$). Those with more than 3 automorphisms also have statistically better learning abilities than those with two automorphisms ($p = 0.05$) and than those without any symmetry axis ($p = 5 \times 10^{-4}$). The same tendency is present with the HNN encoding: networks with at least two automorphisms (i.e., networks with at least one symmetry axis) have a higher GLA score than those that have no symmetry axis (one automorphism, $p = 0.04$); networks with more than 17 automorphisms have a higher GLA score than those with at least two automorphisms ($p = 2 \times 10^{-3}$).

With the HNN encoding, 41% of networks have exactly 24 automorphisms but only 23% of them have 25 or more automorphisms (Figure 2.6, B, blue line). With the map-based encoding, a drop from 100% to 87% occurs at the same number of automorphisms (Figure 2.6, B, grey line). A network with 24 automorphisms is a network in which a sub-network is repeated 4 times ($24 = 4!$, Methods). This number is particular in our experiments because both HNN and the map-based encoding group neurons by 4, therefore the number of automorphisms is expected to be a multiple of 24: a different number means that at least one neuron of a group has a connectivity pattern that is different from the rest of the group. With HNN, this kind irregularity is possible but unlikely. With the map-based encoding, it is not possible, that is why all map-based networks have a number of automorphisms exactly equals to a multiple of 24 (for instance, on figure 2.7, all map-based networks have 24 or $576 = 24 \times 24$ automorphisms).

DISCUSSION

The experiments reported in this chapter add weight to the hypothesis that using a developmental encoding improves the learning abilities of evolved plastic neural networks. Complementary experiments reveal that this result is the consequence of the bias of developmental encodings towards regular structures (Clune and Lipson 2011): (1) encodings that tend to produce more regular networks yielded networks with better general learning abilities; (2) in our experimental setup, whatever the encoding is, networks that are the more regular are statistically those that have the best learning abilities. This second point implies that an indirect encoding that is not biased towards regular network

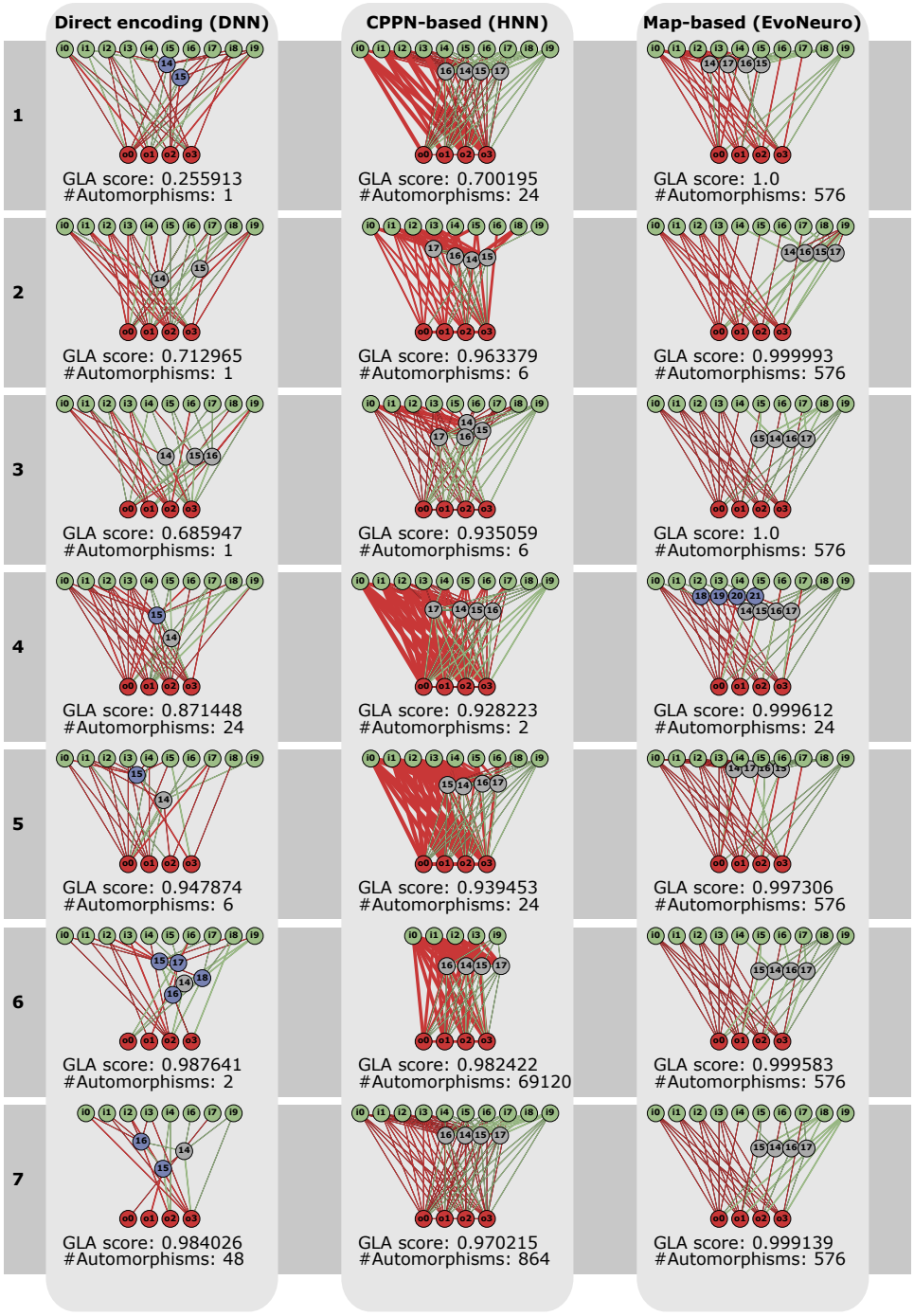


Figure 2.7. **Network with the best learning abilities, for each encoding and each size of the evolutionary learning set.** Each network is the best (in term of learning abilities) of the 30 independent runs. Inhibitory connections are represented as green line and excitatory ones as red lines. The width of the lines is proportional to the corresponding synaptic weight (for modulated connections, the line width is determined after one of the learning phases, for one of the possible association sets). Input neurons are in green, output neurons in red, modulatory neuron in gray and standard neurons in blue. "#automorphisms" means "number of automorphisms" (Methods). Nodes that are not connected (directly or indirectly) to at least one input and one output are not drawn.

should not lead to ANNs with high learning abilities; it also implies that a direct encoding combined with a helper objective that encourages regularity should lead to ANNs with good learning abilities (see (Mouret and Doncieux 2012b) and (Clune* et al. 2013) for examples of helper objectives with a direct encoding). Nonetheless, our experiments show that current generative encodings and neuromodulated Hebbian plasticity make a promising combination to evolve large, plastic neural networks. Future work in this direction should investigate whether this combination holds its promises in other tasks such as learning in a maze (Soltoggio, Bullinaria, et al. 2008; Risi, Hughes, et al. 2010) or visual processing (Stanley, D'Ambrosio, et al. 2009).

According to our results, neural networks evolved with an encoding biased towards regularity could be more flexible than those evolved with an unbiased encoding: they are better at learning association sets that have never been encountered during their evolution. To achieve this flexibility, they have to possess connections that were not directly selected during evolution. In other words, their flexibility stems from "spandrels" (Gould and Lewontin 1979): they

are the byproducts of the bias that make evolution more likely to duplicate a sub-structure than to design a specialized circuit.

These results are in opposition to the general tendency of neural networks to minimize connection costs (Chklovskii et al. 2002; Cherniak et al. 2004; Chen et al. 2006; Clune* et al. 2013) because they show that flexible behaviors require maintaining many "useless" connections. They indicate that a selective pressure for flexibility is likely to favor developmental procedures that would result in connections that do not procure any short-term advantage. In a constant environment, these connections should disappear; but in a constantly changing environment – which puts more pressure on flexibility –, these connections appear critical. This view is consistent with the theory of "variability selection", which posits that flexibility is one of the primary selective pressure that shaped the brains of hominids (Potts 1998; Richerson et al. 2005).

The conflict between flexibility and connection costs also echoes the debate about the modularity/non-modularity of the mammalian brain (Grossberg 2000; Bullmore and Sporns 2009; Meunier et al. 2010), since the minimization of connection costs has been linked with the evolution of modularity (Striedter 2005; Clune* et al. 2013). Our results thus suggest that the parts of the brain that heavily rely on synaptic plasticity to achieve flexible behaviors should be less modular than simpler, less plastic parts. To test this proposition, it is possible to launch computational experiments in which plastic neural networks are evolved with a selective pressure to minimize connection costs and different flexibility requirements.

Pushed to the extreme, the results of our experiments suggest that the best flexibility would be achieved with fully connected networks, since this would be the best possible regularity. In real brains, such a connectivity would be challenging for pure physical reasons (Braitenberg 2001; Chklovskii et al. 2002): if each neuron of a mouse was connected to each other, its brain (about 10 millions neurons) would at least occupy 350 cubic meters (Braitenberg 2001) (about the cranial volume of an Orangutan). Artificial brains do not have such limitations and can be designed as fully connected (Hopfield 1982), but most neural networks used in machine learning are made of layers of neurons, with each layer fully connected to the next one (Cybenko 1989; Haykin 1998). Layers are a very specific structure that prevents some flexibility (non-Markovian tasks cannot be learned), but they make learning easier, because feed-forward networks have no intrinsic dynamics (contrary to recurrent neural networks). These networks are still very regular and flexible. In image processing, convolutional neural networks are classic feed-forward neural networks in which many, well-chosen connections are removed and many synaptic weights are constrained to be equal (LeCun et al. 1998). These networks are much easier to train than classic layered neural networks, but they cannot learn when the input data do not look like images.

These examples highlight a potential trade-off between flexibility and trainability, or, put differently, between learning abilities and learning efficiency: in many situations, it seems beneficial to trade some flexibility to make the system easier to train[3]. Our experiments considered a simple situation in which trainability was not a major concern because the input/output patterns are simple and low-dimensional. In more challenging tasks, the evolutionary process would probably have to find the best trade-off between trainability and flexibility, and therefore between regularity and specialization. Nonetheless, although convolutional networks are less regular than multi-layer perceptrons, they are still very regular and could be generated with a generative encoding. Generative encodings that aim at intermediate regularity might thus be one of the key to explore this trainability/flexibility trade-off.

Overall, the present paper shows that evolution, development and synaptic plasticity are three interleaved processes that are hard to study separately. While an extensive understanding of their interactions is probably out of reach with the current state of knowledge, studies that combine simple models of each of these processes shed light on how one of them – here development – can simplify another – here learning. Such studies appear helpful for both building a global vision of the evolution of intelligent lifeforms as well as harnessing evolution to create intelligent agents.

DETAILED METHODS

Full methods are available in the article in appendix: Tonelli P, Mouret J-B (2013). *On the Relationships between Generative Encodings, Regularity, and Learning Abilities when Evolving Plastic Artificial Neural Networks.* PLoS ONE 8(11): e79138. doi:10.1371/journal.pone.0079138

[3]Flexibility must sometimes be avoided because any change in the behavior might kill the animal or break the robot. This situation would correspond to a very easy learning, because current values are deemed as perfect, and a very low flexibility.

Evolving modular networks

Scientific context. Natural evolution makes it possible for species to adapt to new environments in a few generations; we would like to reproduce these abilities in artificial evolutionary systems to endow robots (and other agents) with similar adaptive abilities. To do so, we first need to understand what make natural organisms so evolvable, which is a central question in evolutionary biology.

A key driver of evolvability (among others) is the widespread modularity of biological networks–their organization as functional, sparsely connected subunits. Intuitively, modular systems seem more adaptable, because it is easier to rewire a modular network with functional subunits than an entangled, monolithic network. Modularity is also a stepping stone to scale systems to many components: to reuse, combine, and duplicate modules, one need to have modules first.

There is no consensus regarding why modularity itself evolved in nature, and therefore no consensus about how to generate modular networks by artificial evolution. We here adopted a computational biology point of view and proposed a new hypothesis to explain the evolution of modular networks. While most hypotheses assume indirect selection for evolvability, we demonstrated that the ubiquitous, direct selection pressure to reduce the cost of connections between network nodes can cause the emergence of modular networks.

In a follow-up study, we combined these findings with a developmental encoding (HyperNEAT) to evolve neural networks that are both modular and regular. We also explored how a connection cost influences the evolution of plastic neural networks.

Human context. Jeff Clune and I met for the second time at the GECCO'2010 conference, in Portland. At this time, we both had unsuccessfully tried to reproduce previous techniques for evolving modular neural networks. Surprised by our results, we subsequently agreed to start a collaboration (by e-mail) to clarify this issue together.

After many failed experiments, we converged on the idea to test a new, simpler hypothesis to explain the evolutionary origins of modularity: modularity might have emerged in nature as a byproduct of the pressure to minimize connection cost. Jeff Clune was at this time working with Hod Lipson at Cornell University, and they invited me to write a paper on this topic during a 1-month stay at Cornell.

A year later, Jeff Clune started his own lab at the university of Wyoming and he supervised two follow-ups of this work (one about using a developmental encoding and one about synaptic plasticity). He invited me to participate in these studies, which are quickly summarized in this chapter.

Main articles:

- Clune* J, Mouret* J-B, Lipson H. (2013) *The evolutionary origins of modularity.* Proceedings of the Royal Society: B. 280: 20122863. http://dx.doi.org/10.1098/rspb.2012.2863 (* equal contribution).
- Huizinga, J. and Mouret, J.-B. and Clune, J. (2014). *Evolving Neural Networks That Are Both Modular and Regular: HyperNeat Plus the Connection Cost Technique.* Proceedings of GECCO. Pages 1-8.
- Ellefsen, K. O., Mouret, J.-B., Clune C. (2015). *Neural modularity helps organisms evolve to learn new skills without forgetting old skills.* Plos Computational Biology.

Related articles:

- Mouret, J.-B. and Doncieux, S. (2009). *Evolving modular neural-networks through exaptation. IEEE Congress on Evolutionary Computation, 2009 (CEC 2009). Pages 1570–1577.*
- Mouret, J.-B. and Doncieux, S. (2008). *MENNAG: a modular, regular and hierarchical encoding for neural-networks based on attribute grammars.* Evolutionary Intelligence. Vol 1 No 3 Pages 187–207.

Other contributors:

- Jeff Clune, University of Wyoming (Assistant Prof.)
- Joost Huizinga, University of Wyoming (PhD student)
- Kai Olav Ellefsen, Norvewian Institute of Science and Technology (PhD student)
- Hod Lipson, Cornell University (Associate Prof.)

Author contributions: *For the main study:* performed the experiments: JBM; wrote the code: JBM; wrote the paper: JBM, JC, and HL; analyzed the results: JBM, JC, and HL. *For the follow-up studies:* performed the experiments: KOE and JH; wrote the code: JBM, KOE, and JH; analyzed the results: KOE, JH, JC, and JBM; wrote the papers: KOE, JH, JC, and JBM.

A long-standing, open question in biology is how populations are capable of rapidly adapting to novel environments, a trait called evolvability (Pigliucci 2008). A major contributor to evolvability is the fact that many biological entities are modular, especially the many biological processes and structures that can be modeled as networks, such as metabolic pathways, gene regulation, protein interactions, and animal brains (Mountcastle 1997; Carroll 2001; Guimera and Amaral 2005; Alon 2006; Wagner, Pavlicev, et al. 2007; Hintze and Adami 2008; Pigliucci 2008). Networks are modular if they contain highly connected clusters of nodes that are sparsely connected to nodes in other clusters (Striedter 2005; Lipson 2007; Wagner, Pavlicev, et al. 2007). Despite its importance and decades of research, there is no agreement on why modularity evolves (Wagner, Mezey, et al. 2001; Wagner, Pavlicev, et al. 2007; Espinosa-Soto and Wagner 2010). Intuitively, modular systems seem more adaptable, a les-

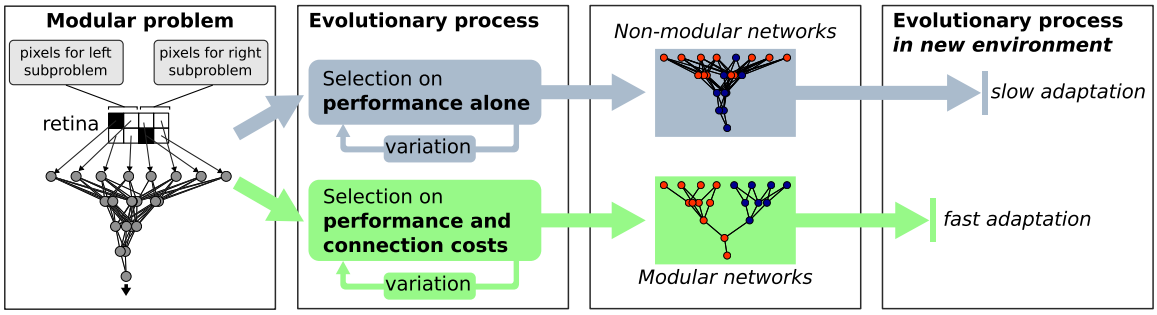


Figure 3.1. **Main hypothesis.** Evolving networks with selection for performance alone produces non-modular networks that are slow to adapt to new environments. Adding a selective pressure to minimize connection costs leads to the evolution of modular networks that quickly adapt to new environments.

son well-known to human engineers (Suh 1990), because it is easier to rewire a modular network with functional subunits than an entangled, monolithic network (Kashtan and Alon 2005; Kashtan, Noor, et al. 2007). However, because this evolvability only provides a selective advantage over the long-term, such selection is at best indirect and may not be strong enough to explain the level of modularity in the natural world (Wagner, Mezey, et al. 2001; Wagner, Pavlicev, et al. 2007).

Modularity is likely caused by multiple forces acting to various degrees in different contexts (Wagner, Pavlicev, et al. 2007), and a comprehensive understanding of the evolutionary origins of modularity involves identifying those multiple forces and their relative contributions. The leading hypothesis is that modularity mainly emerges due to rapidly changing environments that have common subproblems, but different overall problems (Kashtan and Alon 2005; Kashtan, Noor, et al. 2007). Computational simulations demonstrate that in such environments (called modularly varying goals: MVG), networks evolve both modularity and evolvability (Kashtan and Alon 2005; Kashtan, Noor, et al. 2007). In contrast, evolution in unchanging environments produces non-modular networks that are slower to adapt to new environments (Kashtan and Alon 2005; Kashtan, Noor, et al. 2007). Follow-up studies support the modularity-generating force of MVG in nature: the modularity of bacterial metabolic networks is correlated with the frequency with which their environments change (Parter et al. 2007). It is unknown how much natural modularity MVG can explain, however, because it unclear how many biological environments change *modularly*, and whether they change at a high enough frequency for this force to play a significant role (Espinosa-Soto and Wagner 2010). A related theory that also assumes a constantly changing environment and selection for evolvability is that modularity arises to enable modifying one subcomponent without affecting others (Espinosa-Soto and Wagner 2010). There are other plausible hypotheses (reviewed in Wagner 2007), including that variation mechanisms, such as gene duplication, create a bias towards the generation of modular structures (Wagner, Pavlicev, et al. 2007) and that modularity evolves due to selection to make phenotypes robust to environmental perturbations (Wagner, Mezey, et al. 2001).

We investigate an alternate hypothesis that has been suggested, but heretofore untested, which is that modularity evolves not because it conveys evolvability, but as a byproduct from selection to reduce connection costs in a network (Fig. 3.1). Such costs include manufacturing connections, maintaining them, the energy to transmit along them, and signal delays, all of which increase as a function of connection length and number (Cherniak et al. 2004; Striedter 2005; Ahn et al. 2006; Chen et al. 2006). The concept of connection costs is straightforward in networks with physical connections (e.g. neural networks), but costs and physical limits on the number of possible connections may also tend to limit interactions in other types of networks like genetic and metabolic pathways. For example, adding more connections in a signaling pathway might delay the time that it takes to output a critical response; adding regulation of a gene via more transcription factors may be difficult or impossible after a certain number of proximal DNA binding sites are occupied, and increases the time and material required for genome replication and regulation; and adding more protein-protein interactions to a system may become increasingly difficult as more of the remaining surface area is taken up by other binding interactions. Future work is needed to investigate these and other hypotheses regarding costs in cellular networks. The strongest evidence that biological networks face direct selection to minimize connection costs comes from the vascular system (LaBarbera 1990) and from nervous systems, including the brain, where multiple studies suggest that the summed length of the wiring diagram has been minimized, either by reducing long connections or by optimizing the placement of neurons (Chklovskii et al. 2002; Laughlin and Sejnowski 2003; Cherniak et al. 2004; Striedter 2005; Ahn et al. 2006; Chen et al. 2006; Raj and Chen 2011). Founding (Ramón y Cajal 1899) and modern (Striedter 2005) neuroscientists have hypothesized that direct selection to minimize connection costs may, as a side-effect, cause modularity. This hypothesis has never been tested in the context of evolutionary biology. The most related study was on non-evolving, simulated neural networks with a specific within-life learning algorithm that produced more modularity when minimizing connection length in addition to performance (Jacobs and Jordan 1992), although the generality of the result was questioned when it was not replicated with other learning algorithms (Bullinaria 2001). Without during-life learning algorithms, carefully-constructed MVG environments, or mutation operators strongly biased toward creating modules, attempts to evolve modularity in neural networks have failed (Di Ferdinando et al. 2001; Wagner, Mezey, et al. 2001; Clune, Beckmann, McKinley, et al. 2010).

Given the impracticality of observing modularity evolve in biological systems, we follow most research on the subject by conducting experiments in computational systems with

evolutionary dynamics (Kashtan and Alon 2005; Wagner, Pavlicev, et al. 2007; Espinosa-Soto and Wagner 2010). Specifically, we use a well-studied system from the MVG investigations (Kashtan and Alon 2005; Kashtan, Noor, et al. 2007; Clune, Beckmann, McKinley, et al. 2010): evolving networks to solve pattern-recognition tasks and Boolean logic tasks (Methods). These networks have inputs that sense the environment and produce outputs (e.g. activating genes, muscle commands, etc.), which determine performance on environmental problems. We compare a treatment where the fitness of networks is based on performance alone (PA) to one based on two objectives: maximizing performance and minimizing connection costs (P&CC). A multi-objective evolutionary algorithm is used (Deb 2001) with one (PA) or two (P&CC) objectives: To reflect that selection is stronger on network performance than connection costs, the P&CC cost objective affects selection probabilistically only 25% of the time, although the results are robust to substantial changes to this value (Methods). Two example connection cost functions are investigated. The default one is the summed squared length of all connections, assuming nodes are optimally located to minimize this sum (Methods), as has been found for animal nervous systems (Cherniak et al. 2004; Chklovskii 2004; Chen et al. 2006; Pérez-Escudero and Polavieja 2007). A second measure of costs as solely the number of connections yields qualitatively similar results to the default cost function, and may better represent biological networks without connections of different lengths. More fit networks tend to have more offspring (copies that are randomly mutated), and the cycle repeats for a preset number of generations (Fig. 3.1, Methods). Such computational evolving systems have substantially improved our understanding of natural evolutionary dynamics (Lenski, Ofria, Collier, et al. 1999; Lenski, Ofria, Pennock, et al. 2003; Kashtan and Alon 2005; Kashtan, Noor, et al. 2007; Wagner, Pavlicev, et al. 2007; Espinosa-Soto and Wagner 2010).

The main experimental problem involves a network that receives stimuli from eight inputs (Kashtan and Alon 2005). It can be thought of as an eight-pixel retina receiving visual stimuli, although other analogies are valid (Methods), such as a genetic regulatory network exposed to environmental stimuli. Patterns shown on the retina's left and right halves may each contain an 'object', meaning a pattern of interest (Fig. 3.2a). Networks evolve to answer whether an object is present on both the left *and* right sides of the retina (the L-AND-R environment) or whether an object is displayed on *either* side (the L-OR-R environment). Which patterns count as an object on the left and right halves are slightly different (Fig. 2a). Each network iteratively sees all possible 256 input patterns and answers true (≥ 0) or false (< 0). Its performance is the percentage of correct answers, which depends on which neurons are connected, how strongly, and whether those connections are inhibitory or excitatory (Methods). Networks are randomly generated to start each experiment. Their connections stochastically mutate during replication (Methods). Network modularity is evaluated with an efficient approximation (Newman 2006; Leicht and Newman 2008) of the widely used modularity metric Q, which first optimally divides networks into modules then measures the difference between the number of edges within each module and the number expected for random networks with the same number of edges (Newman 2006; Leicht and Newman 2008).

MAIN RESULTS

After 25000 generations in an unchanging environment (L-AND-R), treatments selected to maximize performance and minimize connection costs (P&CC) produce significantly more modular networks than treatments maximizing performance alone (PA) (Fig. 3.2d, $Q = 0.42, 95\%$ confidence interval $[0.25, 0.45]$ vs. $Q = 0.18[0.16, 0.19]$, $p = 8 \times 10^{-09}$ using Matlab's Mann-Whitney-Wilcoxon rank sum test, which is the default statistical test unless otherwise specified). To test whether evolved networks exhibit *functional modularity* corresponding to the left-right decomposition of the task we divide networks into two modules by selecting the division that maximizes Q and color nodes in each partition differently. Left-right decomposition is visually apparent in most P&CC trials and absent in PA trials (Fig. 3.2e,f). Functional modularity can be quantified by identifying whether left and right inputs are in different partitions, which occurrs in 56% of P&CC trials and never with PA (Fisher's exact test, $p = 4 \times 10^{-11}$). Pairs of perfect sub-solution neurons–whose outputs perfectly answer the left and right subproblems–occur in 39% of P&CC trials and 0% of PA trials (Fisher's exact test, $p = 3 \times 10^{-6}$, Supplementary Fig. C.1).

Despite the additional constraint, P&CC networks are significantly higher-performing than PA networks (Fig. 3.2c, Supplementary Fig. C.13). The median-performing P&CC network performs perfectly ($1.0[1.0, 1.0]$), but the median PA network does not ($0.98[0.97, 0.98]$, $p = 2 \times 10^{-05}$). P&CC performance may be higher because its networks have fewer nodes and connections (Supplementary Fig. C.8b,c), meaning fewer parameters to optimize. Modular structures are also easier to adapt since mutational effects are smaller, being confined to subcomponents (Lipson 2007). While it is thought that optimal, non-modular solutions usually outperform optimal, modular designs, such 'modularity overhead' only exists when comparing *optimal* designs, and is not at odds with the finding that *adaptation* can be faster and ultimately more successful with a bias towards modular solutions (Lipson 2007).

To better understand why the presence of a connection cost increases performance and modularity, we searched for the highest-performing networks at all possible combinations of modularity and cost (Methods). For high-performing networks, there is an inverse correlation between cost and modularity, such that the lowest-cost networks are highly modular (Fig. 3.3). Many runs in the P&CC treatment evolved networks in this region whereas the PA treatments never did. There are also many non-modular, high-cost networks that are high-performing, helping to explain why modularity does not evolve due to performance alone (Fig. 3.3). Comparing PA vs. P&CC populations across generations reveals that a connection cost pushes populations out of high-cost, low-modularity areas of the search space into low-cost, modular areas (Fig. 3.2g,h). Without the pressure to leave high-cost, low-modularity regions, many PA networks remain in areas that ultimately do not contain the highest-performing solutions (Fig. 3.3, pink squares in the bottom right), further explaining why P&CC treatments have higher performance.

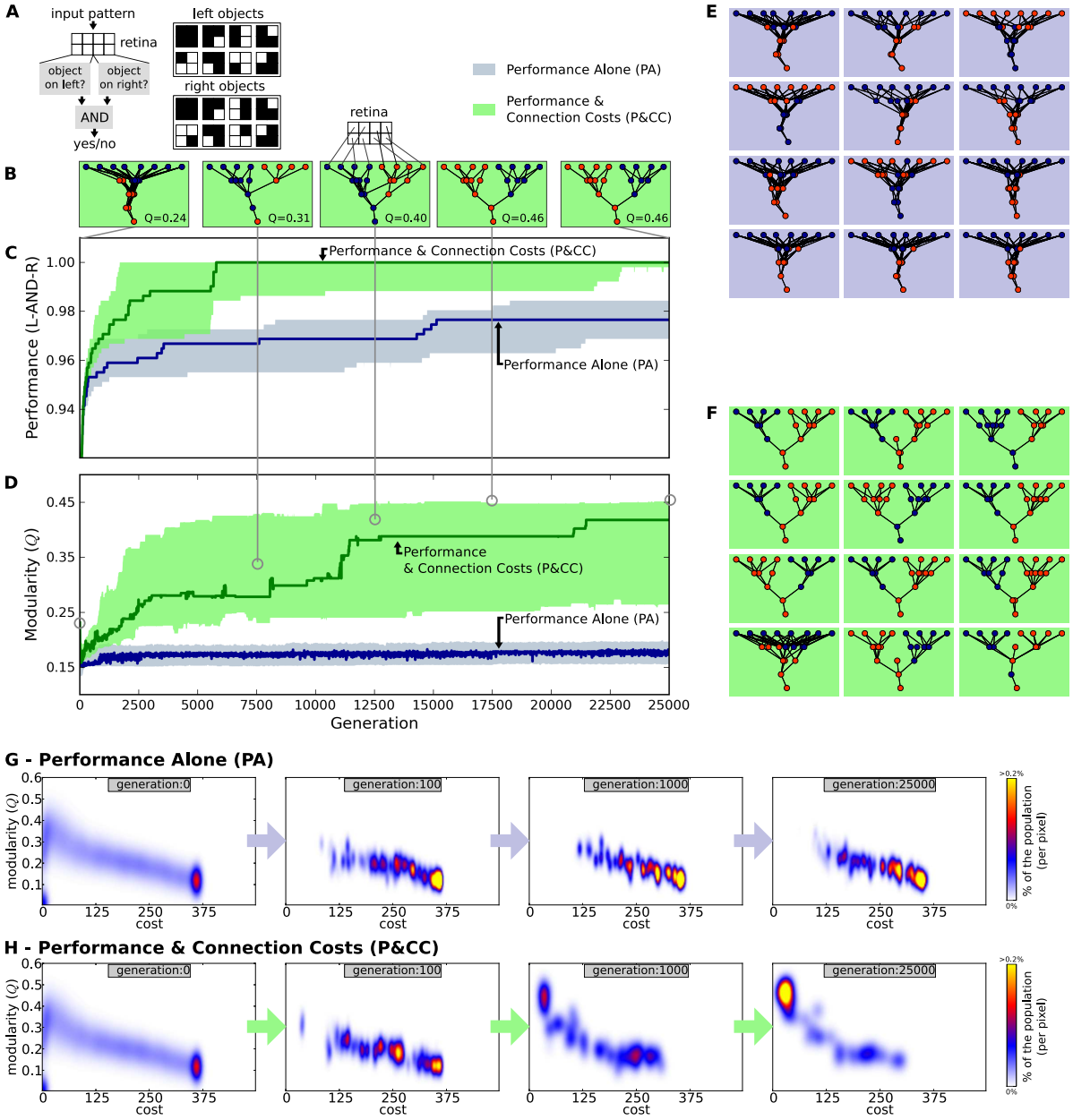


Figure 3.2. The addition of connection costs leads to higher-performing, functionally modular networks. (A) Networks evolve to recognize patterns (objects) in an eight-pixel retina. The problem is modularly decomposable because whether an object exists on the left and right sides can be separately determined before combining that information to answer whether objects exist on *both* sides (denoted by the AND logic function). **(B)** Networks from an example trial become more modular across evolutionary time (see SI for video) with a pressure to minimize connection costs in addition to performance (P&CC). **(C)** Median performance ($\pm$ 95% bootstrapped confidence intervals) per generation of the highest-performing network of each trial, which is perfect only when minimizing connection costs in addition to performance. **(D)** Network modularity, which is significantly higher in P&CC trials than when selecting for performance alone (PA). **(E)** The 12 highest-performing PA networks, each from a separate trial. **(F)** The 12 highest-performing P&CC networks, which are functionally modular in that they have separate modules for the left and right subproblems. Nodes are colored according to membership in separate partitions when making the most modular split of the network (see text). The final networks of all 50 trials are visualized in Supplementary Fig. C.1. **G, H**. Cost and modularity of PA and P&CC populations across generations, pooled from all 50 trials. A connection cost pushes populations out of high-cost, low-modularity regions towards low-cost, modular regions. Fig. 3.3 shows the fitness potential of each map area.

We also found evidence of an inverse correlation between the total cost of a network and modularity in randomly generated networks, irrespective of performance, supporting the intuition that low cost networks are more likely to be modular (Supplementary Fig. C.12).

P&CC networks are also more evolvable than PA networks. We ran additional trials until 50 P&CC and 50 PA trials each had a perfectly performing network (Methods) and transferred these networks into the L-OR-R environment, which has the same subproblems in a different combination (Supplementary Fig. C.6). The presence (P&CC) or absence (PA) of a connection cost remained after the environmental change. We performed 50 replicate experiments for each transferred network. We also repeated the experiment, except first evolving in L-OR-R and then transferring to L-AND-R. In both experiments, P&CC networks exhibit greater evolvability than PA by requiring fewer generations to adapt to the new environment (Fig. 3.4a, L-AND-R$\rightarrow$L-OR-R: $3.0[2.0, 5.0]$ vs. $65[62, 69], p = 3 \times 10^{-78}$; L-OR-R$\rightarrow$L-AND-R: $12.0[7.0, 21.0]$ vs. $222.5[175.0, 290.0], p = 9 \times 10^{-120}$). Modular networks thus evolve because their sparse connec-

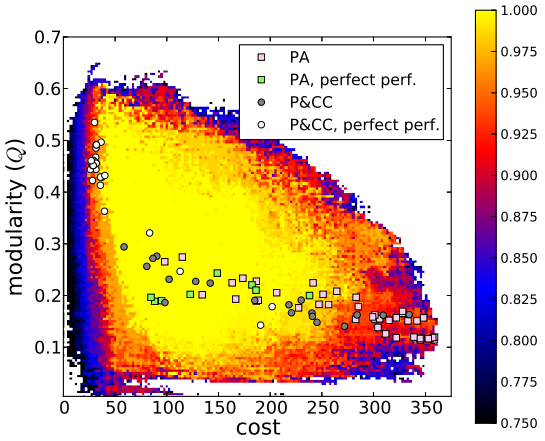


Figure 3.3. The highest-performing networks found for each combination of modularity and cost for the retina problem. Colors indicate the highest-performing network found at that point in the modularity vs. cost space, with yellow representing perfect performance. This map has been generated using the MOLE algorithm (Methods). The best-performing network at the end of each of the 50 PA and P&CC runs are overlaid on the map. Networks with perfect performance exist throughout the space, which helps explain why modularity does not evolve when there is selection based on performance alone. Below a cost threshold of around 125 there is an inverse correlation between cost and modularity for perfectly performing networks. The lowest cost networks–those with the shortest summed lengths–that are high-performing are modular.

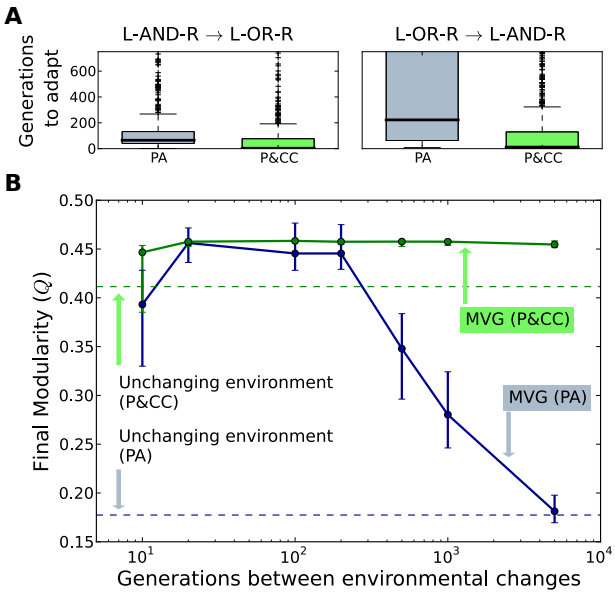


Figure 3.4. Evolving with connection costs produces networks that are more evolvable. (A) P&CC networks adapt faster to new environments than PA networks. Organisms were first evolved in one environment (e.g. L-AND-R) until they reached perfect performance and then transferred to a second environment (e.g. L-OR-R). Thick lines are medians, boxes extend from 25^{th} to 75^{th} data percentiles, thin lines mark $1.5 \times IQR$ (interquartile range), and plus signs represent outliers. Supplementary Fig. C.6 is a zoomed-out version showing all of the data. **(B)** P&CC networks in an unchanging environment (dotted green line) have similar levels of modularity to the highest levels produced by MVG (solid blue line). Combining MVG with P&CC results in even higher modularity levels (solid green line), showing that the forces combined are stronger than either alone.

tivity has lower connection costs, but such modularity also aids performance and evolvability because the problem is modular.

Minimizing connection costs can work in conjunction with other forces to increase modularity. Modularity levels are higher when combining P&CC with MVG environments (Fig. 3.4b: solid vs. dotted green line, $p = 3 \times 10^{-5}$). Overall, P&CC (with or without MVG) yields similar levels of modularity as MVG at its strongest, and significantly more when rates of environmental change are too slow for the MVG effect to be strong (Fig. 3.4: green lines vs. blue solid line).

P&CC modularity is also higher than PA even on problems that are non-modular (Fig. 3.5a, $p = 5.4 \times 10^{-18}$). As to be expected, such modularity is lower than on modular problems ($p = 0.0011$, modular retina vs. non-modular retina). This non-modular problem involves answering whether any four pixels were on (black), which is non-modular because it requires information from all retina inputs. As mentioned previously, performance and modularity are also significantly higher with an alternate connection cost function based on the number of connections (P&CC-NC) instead of the length of connections (Fig. C.7). We also verified that modularity and performance are not higher simply because a second objective is used (Fig. C.7). We further tested whether modularity arises even when the inputs for different modules are not geometrically separated, which is relevant when cost is a function of connection length: Even in experiments with randomized input coordinates (Methods), a connection cost significantly increased performance ($1.0[0.98, 1.0], p = 0.0012$) and modularity ($Q = 0.35[0.34, 0.38], p = 1 \times 10^{-9}$).

All the results presented so far are qualitatively similar in a different model system: evolving networks to solve Boolean logic tasks. We tested two fully separable problems: one with five "exclusive or" (XOR) logic modules (Fig. 3.5b), and another with hierarchically nested XOR problems (Fig. 3.5c). P&CC created separate modules for the decomposed problems in nearly every trial, whereas PA almost never did (Figs. SC.2, SC.3). P&CC performance was also significantly higher (Fig. 3.5b,c), and there was an inverse correlation between cost and modularity (Figs. SC.10). After reading a preprint of this manuscript, a different research group replicated the main result in a different domain: they found that a connection cost causes modularity to evolve when optimizing computer chip architectures (Chung et al. 2012). Confirming the generality of the finding that connection costs improve adaptation rates and that high-performing, low-cost networks are modular is an interesting area for future research.

COMBINATION WITH A DEVELOPMENTAL ENCODING

While modular organization is a key to understand evolvability, its benefits are improved when modules can be reused several times in the same organism, that is, when modularity is combined with *regularity* (Lipson 2007). In this case, modularity does not only facilitates re-wiring, it also improves the ability of an evolutionary system to re-use the module for a different purpose (exaptation) (Gould and Vrba 1982) and to scale up to more complex designs (Lipson 2007).

As described in the previous sections, the evolution of modularity can at least be partially explained by the selective

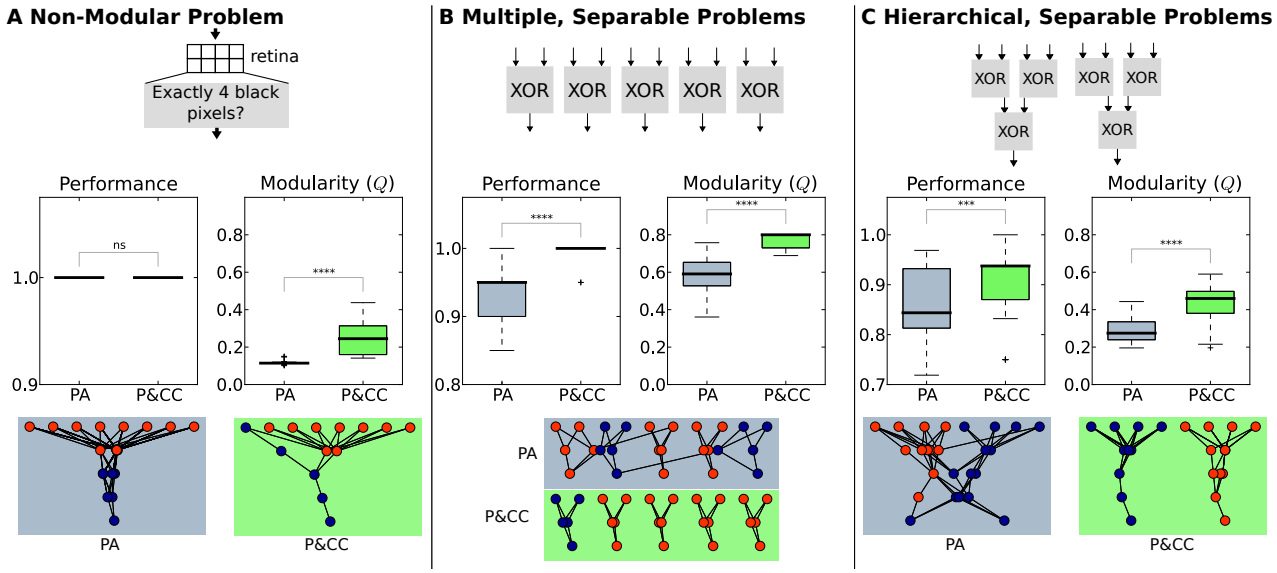


Figure 3.5. Results from tests with different environmental problems. (A) Even on a non-modular problem, modularity is higher with P&CC, though it is lower than for modular problems. **(B, C)** P&CC performs better, is more modular, and has better functional decomposition than PA when evolving networks to solve five separate XOR functions and hierarchically nested XOR functions. The examples are the final, highest-performing networks per treatment. Supplementary Figs. C.2, C.3, and C.4 show networks from all trials. Three and four asterisks indicate p values less than 0.001 and 0.0001, respectively, and *ns* indicates no significant difference.

pressures that give rise to modular networks, may this pressure be the need to minimize connection cost (our hypothesis), to rapidly adapt to changing environments (Kashtan and Alon 2005), or to facilitate specialization in gene activity (Espinosa-Soto and Wagner 2010). Repeating a structure, however, is not possible without a genotype-phenotype map in which the same genes can be re-used several time.

In evolutionary computation, such a genotype-phenotype map are commonly called a *generative encodings* (also called indirect or developmental encodings, see section 1) (Stanley and Miikkulainen 2003; Clune, Stanley, et al. 2011). They are often inspired by natural developmental systems, such as gene regulatory networks, cell division, or chemical gradients, making them more biologically plausible than direct encodings (Wagner and Altenberg 1996; Müller 2007; Stanley 2007). In these generative encodings, compact genomes describe large phenotypes via the reuse of genomic information, giving rise to regular structures. In fact, if we consider the genotype as a compression of the phenotype, large phenotypes encoded by small genotypes are regular by definition.

In this work, we employ the HyperNEAT (section 1) (Stanley, D'Ambrosio, et al. 2009) encoding, which encodes neural networks with a generative encoding called Compositional Pattern Producing Networks (CPPNs) (Stanley 2007). CPPNs produce spatial patterns that exhibit regularity with variation. These spatial patterns define the connectivity across the geometric layout of nodes, enabling HyperNEAT to produce networks that exhibit structural regularity (Clune, Stanley, et al. 2011). We compare three encodings: (1) a direct encoding, (2) HyperNEAT with a Gaussian seed, which is a variant of HyperNEAT that improves the modularity of neural networks by starting the search with networks that are likely to mostly have local connections (Verbancsics and Stanley 2011), and (3) HyperNEAT with a connection cost, computed exactly as in the previous section.

In the three treatment, we added an objective of behavioral diversity (Mouret and Doncieux 2012a; Doncieux and Mouret 2014) wich replaces the speciation mechanism present in the original NEAT/HyperNEAT implementation. Encouraging behavioral diversity in this successfully prevents premature convergence in many experiments (Mouret and Doncieux 2012a) and is easier to set up than the speciation used in NEAT. Technically, behavioral diversity of an individual is calculated by storing the output for every possible input in a binary vector (< 0 is false, > 0 is true) and then taking the average Hamming distance to the binary vector of all other individuals in the population.

We have tested all treatments on three modular and regular problems from the previously described experiments: the Retina Problem, the 5-XOR problem, and the Hierarchical XOR problem. We here focus on the retina problem but results are qualitatively similar with the other problems.

Results

In the retina experiment, the performance of HyperNEAT-CCT is significantly higher at nearly every generation than both HyperNEAT and HyperNEAT-GS (Fig. 3.6A); even after the medians of all treatments have reached perfect performance, lower-performing runs in the HyperNEAT and HyperNEAT-GS treatments make those treatments perform significantly worse than HyperNEAT-CCT. In terms of modularity, the level for HyperNEAT hardly changes over time, while the modularity of HyperNEAT-CCT progressively in- creases; the difference becomes significant after 12000 generations (Fig. 3.6B). The modularity of HyperNEAT-GS, on the other hand, spikes during the first few generations, but then it decreases over time to a significantly lower level than HyperNEAT-CCT (Fig. 3.6B). This behavior is evidence for our hypothesis that the Gaussian seed may not be an effective way to promote modularity in cases where there is no immediate fitness benefit.

To examine functional modularity we look at the best

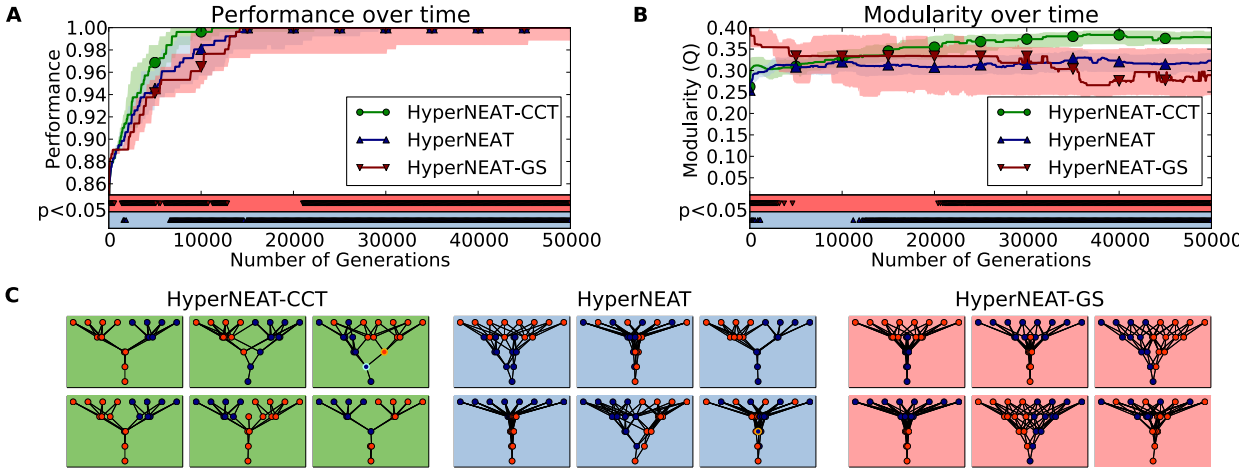


Figure 3.6. **Combining the minimization of connection costs with HyperNEAT (a developmental encoding) (A-B)**. HyperNEAT with connection Cost (HyperNEAT-CCT) significantly outperforms and has higher modularity than both HyperNEAT and HyperNEAT-GS (HyperNEAT with Gaussian seed). **(C)HyperNEAT-CCT networks are visually more modular, exhibit left-right modularity more often, and solve significantly more sub-problems than HyperNEAT or HyperNEAT-GS networks.**

networks produced after 50000 generations. When considering the number of sub-problems solved, HyperNEAT-CCT networks solve an average of 0.67 (out of 2) sub-problems, which is significantly ($p = 0.024$) higher than HyperNEAT networks, which solve an average of 0.41 sub-problems. The differences in modularity are also visually apparent (Fig. 3.6C). The networks of HyperNEAT-CCT look more modular, demonstrate left-right modularity more often, and have more nodes that solve sub-problems than the Hyper- NEAT and HyperNEAT-GS networks.

Comparing HyperNEAT-CCT with a direct encoding (P&CC in the previous section, except that P&CC here uses a behavioral diversity objective), the direct encoding is significantly higher performing in early generations and significantly more modular throughout evolution. The indirect encoding of HyperNEAT seems to struggle more with the irregularities of this problem and is not as good at pruning connections. That is expected, since removing connections in the direct encoding is easy compared to doing so in HyperNEAT (Clune, Stanley, et al. 2011), which has to adapt the patterns produced by the LEO node such that it cuts off the redundant parts while keeping the rest of the network intact.

A main advantage of HyperNEAT is its ability to produce regular patterns (Stanley, D'Ambrosio, et al. 2009; Clune, Stanley, et al. 2011). Compression tests (using gzip to compress the weight matrix, see (Clune, Stanley, et al. 2011)) reveal that HyperNEAT-CCT networks are significantly more regular than the direct encoding (P&CC): the direct encoding with CCT becomes 38% smaller upon compression, but HyperNEAT-CCT compresses further down by 43%, making it significantly more compressible ($p < 0.00001$). We did not count the number of automorphisms (as we did it in the previous chapter) because the two studies we mostly performed concurrently, but this alternative method to evaluate regularity would have been relevant here. The regularity of HyperNEAT-CCT is also visually apparent (Fig. 3.6C). In many of its networks, the left side is mirrored on the right side, even though the signs of the connections are sometimes switched. Other networks feature alternating or mirrored patterns in the biases or connections. While HyperNEAT-CCT networks also exhibit some clear variations in each of its patterns, on balance they are much more regular than the DirectEncoding-CCT networks (Fig. 3.6b), which do not show any discernible patterns.

Overall, this follow-up study shows that the connection cost technique can be used in combination with a generative encoding. However, the developmental encoding did not significantly increase the modularity nor the performance in the tasks we tested. The differences are likely to be more visible in tasks with more inputs/outputs in which direct encodings are usually easily outperformed by developmental encodings (Stanley, D'Ambrosio, et al. 2009; Gauci and Stanley 2010; Clune, Stanley, et al. 2011).

COMBINATION WITH NEUROMODULATED HEBBIAN LEARNING

In a parallel study, we investigated whether a connection cost could lead to improvements in the learning abilities of evolved *plastic* neural networks (see chapter 2). Modularity intuitively should improve learning by having a reinforcement learning module separate from sensory processing modules, allowing learning to happen only in response to a positive or negative reward. In addition, in a scenario in which the agent has to learn several tasks during its lifetime, modularity could also reduce learning interference between tasks by separating functionality into physically distinct modules in which learning can be selectively turned on or off.

We tested these ideas in an abstract world in which an organism performs a daily routine of trying to eat nutritious food while avoiding eating poisonous food. Every day the organism observes every food item one time: half of the food items are nutritious and half are poisonous. To achieve maximum fitness, the individual needs to eat all the nutritious items and avoid eating the poisonous ones. After a number of days, the season changes abruptly from a summer season to a winter season. In the new season, there is a new set of food sources, half of them nutritious and half poisonous, and the organism has to learn which is

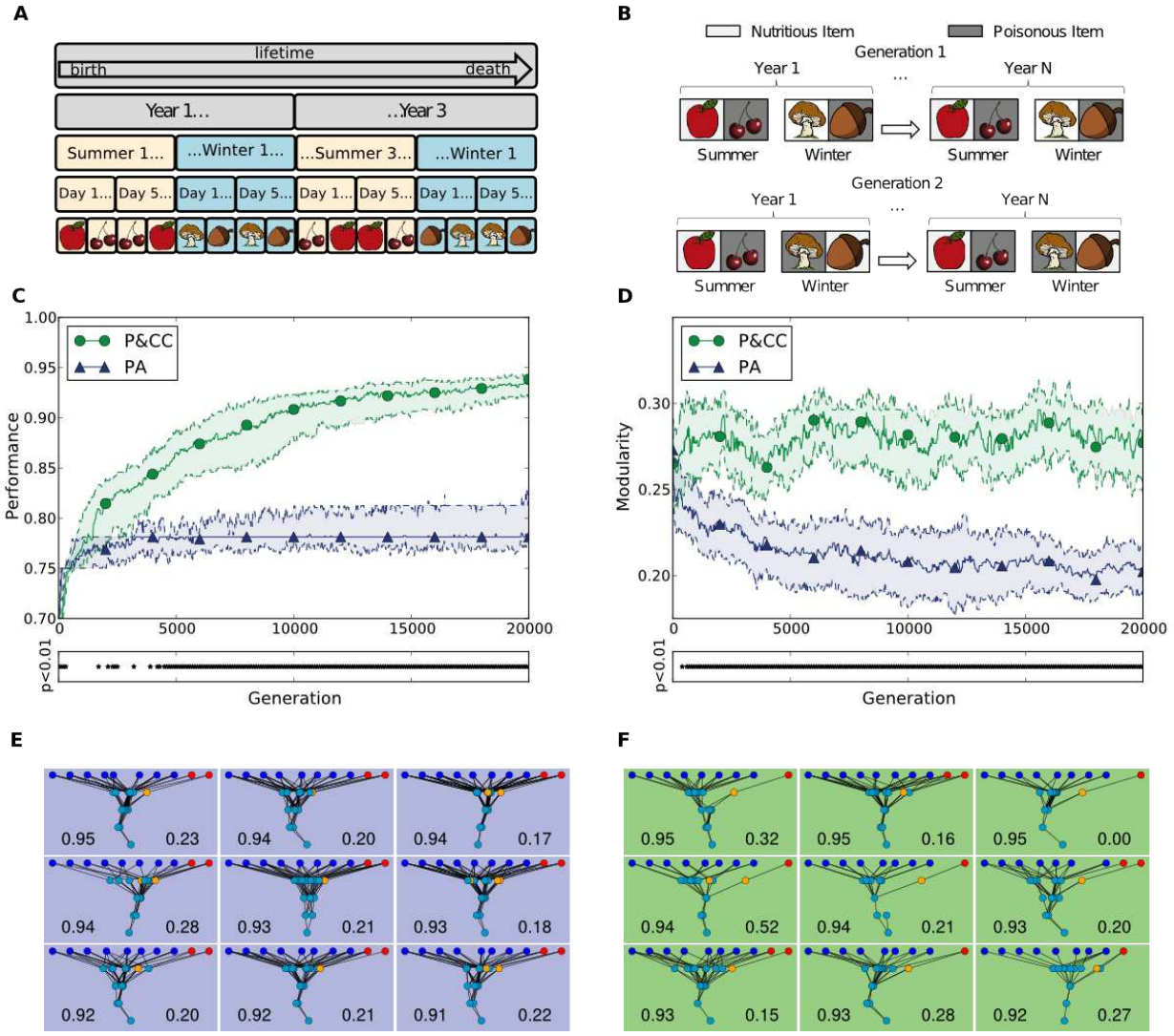


Figure 3.7. **Modularity, connection cost, and plastic neural networks. (A)** The environment for one individual's lifetime. A lifetime lasts 3 years. Each year has 2 seasons: winter and summer. Each season consists of 5 days. In each day, each individual sees all food items available in that season (only two are shown) in a random order. **(B)** To ensure that agents learn associations within their lifetimes instead of genetically hardcoding associations, whether each food item is nutritious or poisonous is randomized each generation. There are four food items per season (two are depicted). **(C-D)**. The addition of a cost for network connections, which is present only in the P&CC treatment, significantly increases performance **(C)** and modularity **(D)**. For each treatment, the median from 100 independent evolution experiments is shown +/- 95% bootstrapped confidence intervals of the median. Asterisks below each plot indicate statistically significant differences at $p < 0.01$ according to the Mann-Whitney U test. **(E)** PA networks are visually non-modular whereas P&CC networks **(F)** tend to create a separate module for learning (orange neurons). Dark blue nodes are inputs that encode which type of food has been encountered. Light blue nodes indicate internal, non-modulatory neurons. Dark orange nodes are reward or punishment inputs that indicate if a nutritious or poisonous item has been eaten. Light orange neurons are neuromodulatory neurons that regulate learning. P&CC networks tend to separate the reward/punishment inputs and neuromodulatory neurons into a separate module that applies learning to downstream neurons that determine which actions to take. For each treatment, the highest-performing network from each of the nine highest-performing evolution experiments are shown. In each panel, the left number reports performance and the right number reports modularity.

which. After this winter season, the environment changes back to the summer season and the food items and their nutritious/poisonous statuses are the same as in the previous summer. Globally, this environment rewards organisms that can learn quickly and avoid forgetting between seasons.

Like in chapter 2, we used neuro-modulated Hebbian plasticity. To mitigate premature convergence, we also used a behavioral diversity objective, like in the previous section.

Results

The addition of a cost for connections (the P&CC treatment) leads to a rapid, sustained, and statistically significant fitness advantage versus not having a connection cost (the PA treatment) (Fig 3.7A). In addition, P&CC networks learn associations faster in their first summer and winter, and maintain higher performance over multiple years (pairs of seasons). As expected, the presence of a connection cost also significantly increases network modularity (Fig. 3.7B). Networks evolved in the P&CC treatment tend to create a separate reinforcement learning module that contains the reward and punishment inputs and most or all neuromodulatory neurons (Fig. 3.7 E-F).

To quantify whether learning is separated into its own module, we tested the frequency with which the reinforcement inputs (reward/punishment signals) were placed into a different module from the remaining food-item inputs. This measure reveals that P&CC networks have a separate module for learning in 31% of evolutionary trials, whereas

only 4% of the PA trials do, which is a significant difference ($p = 2.71 \times 10^{-}-7$). Analyses also reveal that the networks from both treatments that have a separate module for learning perform significantly better than networks without this decomposition (median performance of modular networks in 80 randomly generated environments: $0.87[95\%CI : 0.83, 0.88]$ vs. non-modular networks: $0.80[0.71, 0.84], p = 0.02$). Even though only 31% of the P&CC networks are deemed modular in this particular way, the remaining P&CC networks are still significantly more modular on average than PA networks (median Q scores are $0.25[0.23, 0.28]$ and $0.2[0.19, 0.22]$respectively, $p = 4.37 \times 10^{-6}$).

CONCLUSION

The reported experiments support the hypothesis that selection to reduce connection costs causes modularity, even in unchanging environments. The results also open new areas of research into identifying connection costs in networks without physical connections (e.g. genetic regulatory networks) and investigating whether pressures to minimize connection costs may explain modularity in human-created networks (e.g. communication and social networks).

It is tempting to consider any component of modularity that arises due to minimizing connection costs as a "spandrel", in that it emerges as a byproduct of selection for another trait (Gould and Lewontin 1979; Solé and Valverde 2006). However, because the resultant modularity produces evolvability, minimizing connection costs may serve as a bootstrapping process that creates initial modularity that can then be further elevated by selection for evolvability. Such hypotheses for how modularity initially arises are needed, because selection for evolvability cannot act until enough modularity exists to increase the speed of adaptation (Wagner, Pavlicev, et al. 2007).

DETAILED METHODS

Supplementary figures are available in appendix C.

The main methods are described in: Clune* J, Mouret* J-B, Lipson H. (2013) *The evolutionary origins of modularity.*Proceedings of the Royal Society: B. 280: 20122863. http://dx.doi.org/10.1098/rspb.2012.2863 (* equal contribution).

Additional methods are described in

- Huizinga, J. and Mouret, J.-B. and Clune, J. (2014). *Evolving Neural Networks That Are Both Modular and Regular: HyperNeat Plus the Connection Cost Technique.* Proceedings of GECCO. Pages 697-704.
- Ellefsen, K. O., Mouret, J.-B., Clune C. (2015). *Neural modularity helps organisms evolve to learn new skills without forgetting old skills.* Plos Computational Biology.

Diagnosis-free adaptation to damage

Scientific context. As robots leave the controlled environments of factories to autonomously function in more complex, natural environments (Bellingham and Rajan 2007; Yoerger 2008; Broadbent et al. 2009), they will have to respond to the inevitable fact that they will become damaged (Carlson and Murphy 2005; Sanderson 2010). However, while animals can quickly adapt to a wide variety of injuries, current robots cannot "think outside the box" to find a compensatory behavior when damaged: they are limited to their pre-specified self-sensing abilities, can diagnose only anticipated failure modes (Blanke and Schröder 2006), and require a pre-programmed contingency plan for every type of potential damage, an impracticality for complex robots (Carlson and Murphy 2005; Sanderson 2010).

We proposed two novel algorithms to address these issues. Both of them rely on the same concepts: (1) trial-and-error learning and evolutionary algorithms can find compensatory behaviors without requiring self-diagnosis or pre-specified contingency plans, and (2) a simulator of the *intact* robot can be used to guide the search for a compensatory behavior even if the physical robot is damaged. The underlying intuition is that behaviors that do not use the damaged part will perform similarly in simulation and in reality. In both cases, we mainly evaluate our algorithm with a 6-legged robot injured in five different ways, including damaged, broken, and missing legs.

Our first algorithm, called T-resilience, is based on the transferability approach (Koos, Mouret, et al. 2013), whose original purpose was to cross the reality gap that separates performance of controllers evolved in simulation with their performance on a real robot. In this algorithm, the robot runs an evolutionary algorithm online and periodically transfers a few individuals to learn a predictor of the transferability. Adaptation times are between 20 minutes and 1 hour, depending on the available computational power.

The second algorithm, called "intelligent trial and error", allows robots to adapt to damage in less than two minutes. Before deployment, the robot exploits a novel algorithm to create a detailed map of the space of high-performing behaviors: This map represents the robot's intuitions about what behaviors it can perform and their value. The algorithm used to create the map is a follow-up of the MOLE algorithm that we designed to study the evolution of modularity in networks. If the robot is damaged, it uses the intuitions provided by the map to guide a trial-and-error learning algorithm based on Bayesian optimization. Thanks to our approach, the robot rapidly discovers a compensatory behavior that works in spite of the damage. Experiments reveal successful adaptations for our hexapod robot and for a robotic arm with joints broken in 14 different ways.

Since "intelligent trial and error" clearly outperforms the "t-resilience", this chapter is mainly focused on "intelligent trial and error".

Human context. The first algorithm (T-Resilience) was designed with Sylvain Koos (post-doc/ATER) as a follow-up of his PhD on the transferability approach. I was the main supervisor of Sylvain Koos during this post-doc and the co-supervisor of his PhD (Stéphane Doncieux: 50%, JBM: 50%). Antoine Cully worked on the experiments with Sylvain Koos during his master thesis. Antoine Cully subsequently started a PhD, that I co-supervise (Stéphane Doncieux: 10%, JBM: 90%). The second algorithm (Intelligent trial and error) is one of the main contributions of Antoine's Cully PhD. As the Intelligent Trial and Error relies on MAP-Elites, an algorithm that I co-authored with Jeff Clune while working on the evolutionary origins of modularity, Jeff Clune agreed to help us write the paper, discuss new experiments, and improve our analyses. Danesh Tarapore (post-doc, whom I supervised) joined the team later to help us finish the experiments and the paper.

Main articles:

- Cully, A., Clune, J., Tarapore, D., and Mouret, J.-B. (2015). *Robots that can adapt like Natural Animals*. Nature. (to appear).
- Koos, S. and Cully, A. and Mouret, J.-B. (2013). *Fast Damage Recovery in Robotics with the T-Resilience Algorithm*. International Journal of Robotics Research. Vol. 32:14. pp 1700-1723

Related articles:

- Cully, A. and Mouret, J.-B. (2015). Evolving a behavioral repertoire for a walking robot. *Evolutionary Computation*. Pages to appear.

Other contributors:

- Antoine Cully, Pierre and Marie Curie University (PhD student)
- Danesh Tarapore, Pierre and Marie Curie University (Post-doc)
- Sylvain Koos, Pierre and Marie Curie University (Post-doc)
- Jeff Clune, University of Wyoming (Assistant Prof.)

Author contributions:

- for the paper in Nature: A.C. and J.-B.M. designed the study. A.C. and D.T. performed the experiments. A.C., J.-B.M., D.T. and J.C. analyzed the results and wrote the paper.
- for the paper in IJRR: S.K. and J.-B.M. designed the study. A.C. and S.K. performed the experiments. S.K., A.C., and J.-B.M. analyzed the results and wrote the paper.

Robots have transformed the economics of many industries, most notably manufacturing (Siciliano and Khatib 2008), and have the power to deliver tremendous benefits to society, such as in search and rescue (Murphy 2004), disaster response (Nagatani et al. 2013), health care (Broadbent et al. 2009), and transportation (Thrun et al. 2006). They are also invaluable tools for scientific exploration, whether of distant planets (Bellingham and Rajan 2007; Sanderson 2010) or deep oceans (Yoerger 2008). A major obstacle to their widespread adoption in more complex environments outside of factories is their fragility (Carlson and Murphy 2005; Sanderson 2010): Robots presently pale in comparison to natural animals in their ability to invent compensatory behaviors after an injury (Fig. 4.1A).

Current damage recovery in robots typically involves two phases: self-diagnosis, and then selecting or planning the best contingency plan (Fenton et al. 2001; Verma et al. 2004; Bongard et al. 2006; Kluger and Lovell 2006). Such self-diagnosing robots are expensive, because self-monitoring sensors are expensive, and are difficult to design, because robot engineers cannot foresee every possible situation: this approach often fails either because the diagnosis is incorrect (Fenton et al. 2001; Bongard et al. 2006) or because an appropriate contingency plan is not provided (Kluger and Lovell 2006).

Injured animals respond differently: they learn by trial and error how to compensate for damage (e.g. learning which limp minimizes pain) (Jarvis et al. 2013; Fuchs et al. 2014). Similarly, trial-and-error learning algorithms could allow robots to creatively discover compensatory behaviors without being limited to their designers' assumptions about how damage may occur and how to compensate for each damage type. However, state-of-the-art learning algorithms are impractical because of the "curse of dimensionality" (Kober and Peters 2013): the fastest algorithms constrain the search to a few behaviors (e.g. tuning only 2 parameters, requiring 5-10 minutes) or require human demonstrations (Kober and Peters 2013). Algorithms without these limitations take several hours (Kober and Peters 2013). Damage recovery would be much more practical and effective if robots adapted as creatively and quickly as animals (e.g. in minutes) and without expensive self-diagnosing sensors.

A FIRST APPROACH: T-RESILIENCE

When evolving controllers for robots, Evolutionary Algorithms (EAs) are reported to require many hundreds of trials on the robot and to last from two to tens of hours (e.g. (Hornby, Takamura, et al. 2005; Yosinski et al. 2011)). These EAs spend most of their running time in evaluating the quality of controllers by testing them on the target robot. Since, contrary to simulation, reality cannot be sped up, their running time can only be improved by finding strategies to evaluate fewer candidate solutions on the robot.

By first learning learning a self-model for the robot, then evolving a controller with this simulation, Bongard et al. (Bongard et al. 2006) designed an algorithm for resilience that makes an important step in this direction. Nevertheless, this algorithm has a few important shortcomings. First, actions and models are undirected: the algorithm can "waste" a lot of time to improve parts of the self-model that are irrelevant for the task. Second, it is computationally expensive because it includes a full learning algorithm (the second stage, in simulation) and an expensive process to select each action that is tested on the robot. Third, there is often a "reality gap" between a behavior learned in simulation and the same behavior on the target robot (Jakobi et al. 1995), but nothing is included in Bongard's algorithm to prevent such gap to happen: the controller learned in the simulation stage may not work well on the real robot, even if the self-model is accurate.

Our algorithm is inspired by the "transferability approach" (Koos, Mouret, et al. 2013), which we originally developed to cross the "reality gap" that separates behaviors optimized in simulation to those observed on the target robot (Jakobi et al. 1995). The main proposition of this approach is to make the optimization algorithm aware of the limits of the simulation. To this end, a few controllers are transferred during the optimization and a regression algorithm (here a SVM) is used to approximate the function that maps behaviors in simulation to the difference of performance between simulation and reality. To use this approximated *transferability function*, the single-objective optimization problem is transformed into a multi-objective optimization in which both performance in simulation and transferability are maximized. This optimization is performed with a a multi-objective evolutionary algorithm (NSGA-II, (Deb et al. 2002)).

The same concepts can be applied to design a fast adaptation algorithm for resilient robotics, leading to a new algorithm that we called "T-Resilience" (for Transferability-based resilience). If a damaged robot embeds a simulation of itself, then behaviors that rely on damaged parts will not be transferable: they will perform very differently in the self-model and in reality. During the adaptation process, the robot will thus create an approximated transferability function that classifies behaviors as "working as expected" and "not working as expected". Hence the robot will possess an "intuition" of the damage but it will not explicitly represent or identify them. By optimizing both the transferability and the performance, the algorithm will look for the most efficient behaviors among those that only use the reliable parts of the robots. The robot will thus be able to sustain a functioning behavior when damage occurs by learning to avoid behaviors that it is unable to achieve in the real world.

Experiments

We evaluate the T-Resilience algorithm on an 18-DOFs hexapod robot that needs to adapt to motor failures and broken legs (figure 4.2); we compare it to stochastic local search (Hoos and Stützle 2005), policy gradient (Kohl and Stone 2004) and Bongard's algorithm (Bongard et al. 2006). The algorithms are implemented in the Sferes$_{v2}$ framework (Mouret and Doncieux 2010). The behavior on the real robot is assessed on-board thanks to a RGB-D sensor coupled with a state-of-the-art SLAM algorithm (Endres et al. 2012). For each experiment, a population of 100 controllers is optimized for 1000 generations. Every 40 generations, a controller is randomly selected in the population and transferred on the robot.

Using only 25 tests on the robot and an overall running time of less than one hour on a recent laptop, T-Resilience

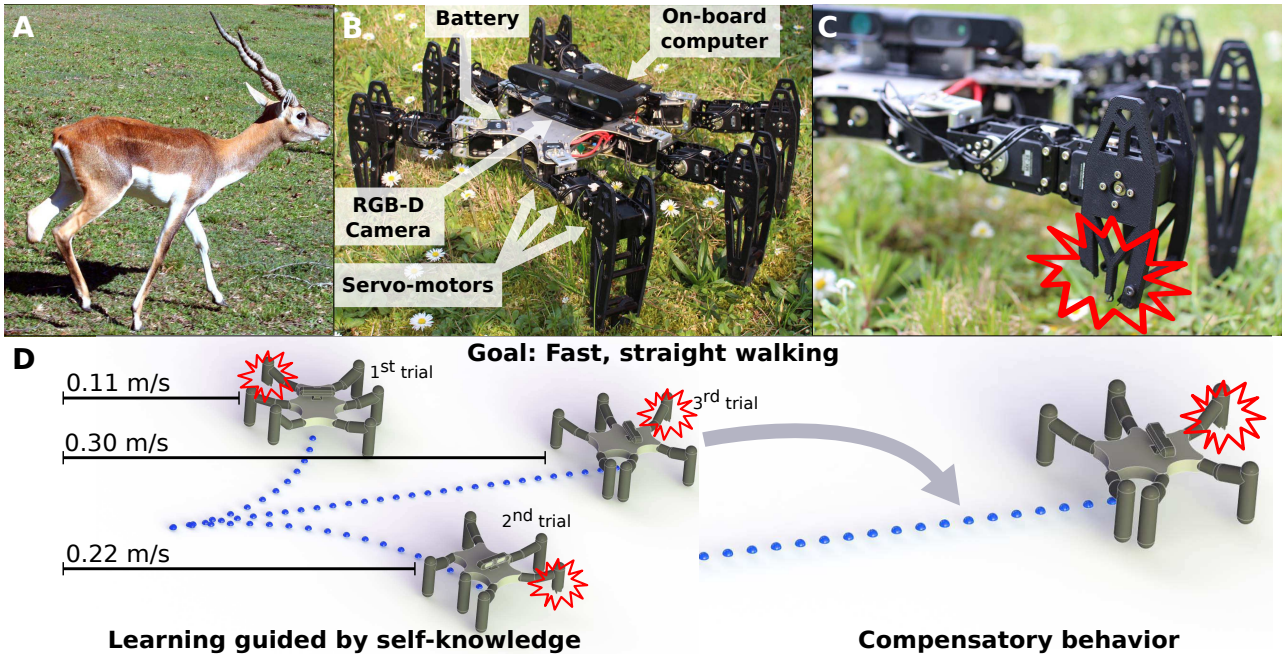


Figure 4.1. With learning and evolutionary algorithms, robots, like animals, can quickly adapt to recover from damage. (A) Most animals can find a compensatory behavior after an injury. Without relying on *predefined* compensatory behaviors, they learn how to avoid behaviors that are painful or no longer effective. **(B)** An undamaged, hexapod robot. **(C)** One type of damage the hexapod may have to cope with (broken leg). **(D)** After damage occurs, in this case making the robot unable to walk straight, damage recovery begins. The robot tests different types of behaviors and has to find a compensatory behaviors with only a few dozen tests.

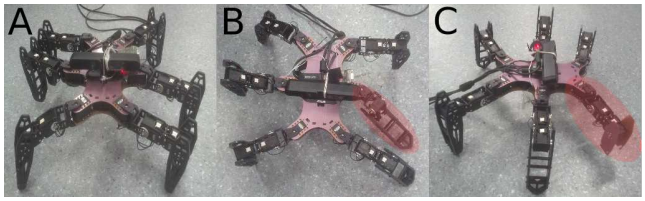


Figure 4.2. (A) The hexapod robot is not damaged. (B) The left middle leg is no longer powered. (C) The terminal part of the front right leg is shortened by half.

consistently leads to substantially better results than the other approaches (figure 4.3).

A FASTER APPROACH: INTELLIGENT TRIAL AND ERROR

T-Resilience is significantly faster than other algorithms because it leverages a dynamic simulator and does not attempt to diagnose the damage. However, evolution happens online: the algorithm requires a lot of computational power and, to obtain fast adaptation times, we had to limit the fine-tuning of solutions as well as the exploration of novel behaviors. As this section will show it, this issue can be addressed by decoupling the exploration in simulation from the tests conducted on the physical robot.

More precisely, we show that faster and better adaptation can be achieved by guiding an intelligent trial-and-error learning algorithm with an automatically generated, pre-computed, behavior-performance map that predicts the performance of thousands of different behaviors. The key insight is that, whereas current learning algorithms either start with no knowledge of the search space (Kober and Peters 2013) or with minimal knowledge from a few human demonstrations (Argall et al. 2009; Kober and Peters 2013), animals better understand the space of possible behaviors and their value from previous experience (Thelen 1995), enabling injured animals to intelligently select tests that validate or invalidate whole families of promising compensatory behaviors.

We have robots store knowledge from previous experience in the form of a map of the behavior-performance space. Guided by this map, a damaged robot tries different types of behaviors that are predicted to perform well and, as tests are conducted, updates its estimates of the performance of those types of behaviors. The process ends when the robot predicts that the most effective behavior has already been discovered. The result is a robot that quickly discovers a way to compensate for damage (e.g. Fig. 4.1C) without a detailed mechanistic understanding of its cause, as occurs with animals. We call this approach "Intelligent Trial and Error" (Fig. 4.1D).

The behavior-performance map is created with a novel algorithm and a simulation of the robot, which either can be a standard physics simulator or can be automatically discovered (Bongard et al. 2006). The robot's designers only have to describe the dimensions of the space of possible behaviors and a performance measure. For instance, walking gaits could be described by how much each leg is involved in a gait (a behavioral measure) and speed (a performance measure). For grasping, performance could be the amount of surface contact, and it has been demonstrated that 90% of effective poses for the 21-degree-of-freedom human hand can be captured by a 3-dimensional behavioral space (Santello 1998). To fill in the behavior-performance map, an optimization algorithm simultaneously searches for a high-performing solution at each point in the behavioral

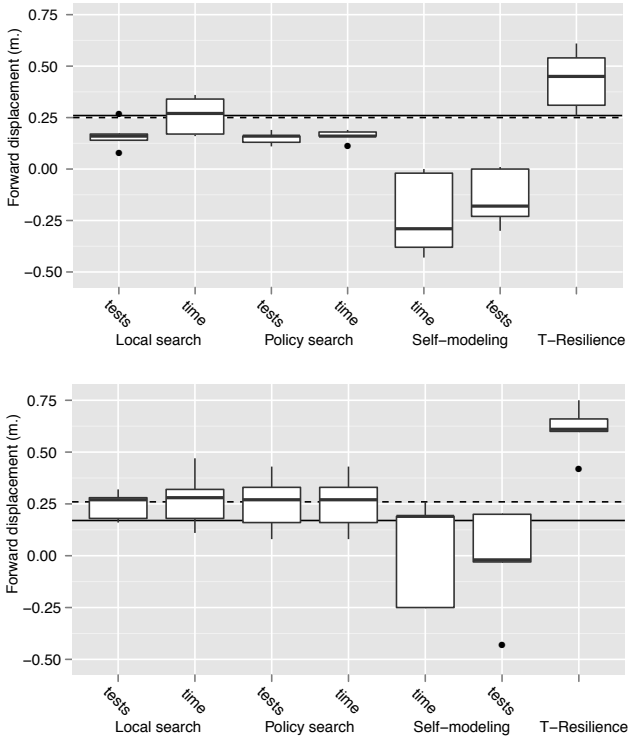


Figure 4.3. Performances (distance covered in 3 seconds) obtained in case B (top) and C (bottom). On each box, the central mark is the median, the edges of the box are the 25th percentile and the 75th percentile. The whiskers extend to the most extreme data point which is no more than 1.5 times the length of the box away from the box. Each algorithm has been run 5 times and distances are measured using the external motion capture system. Except for the T-Resilience, the performance of the controllers found after about 25 transfers ($tests$) and after about 20 minutes ($time$) are depicted (all T-Resilience experiments last about 20 minutes and use 25 transfers). The horizontal lines denote the performances of the reference gait, according to the CODA motion capture system (dashed line) and according to the SLAM algorithm (solid line).

space (Fig. 4.4A,B and Supplementary Fig. D.1). This step requires simulating millions of behaviors, but needs to be performed only once per robot design before deployment (Methods).

A low confidence is assigned to the predicted performance of behaviors stored in this behavior-performance map because they have not been tried in reality (Fig. 4.4B and Supplementary Fig. D.1). During the robot's mission, if it senses a performance drop, it selects the most promising behavior from the behavior-performance map, tests it, and measures its performance. The robot subsequently updates its prediction for that behavior and nearby behaviors, assigns high confidence to these predictions (Fig. 4.4C and Supplementary Fig. D.1), and continues the selection/test/update process until it finds a satisfactory compensatory behavior (Fig. 4.4D and Supplementary Fig. D.1).

All of these ideas are technically captured via a Gaussian process model (Rasmussen and Williams 2006), which approximates the performance function with already acquired data, and a Bayesian optimization procedure (Borji and Itti 2013; Mockus 2013), which exploits this model to search for the maximum of the performance function (Methods). The robot selects which behaviors to test by maximizing an information acquisition function that balances exploration (selecting points whose performance is uncertain) and exploitation (selecting points whose performance is expected to be high) (Methods). The selected behavior is tested on the physical robot and the actual performance is recorded. The algorithm updates the expected performance of the tested behavior and lowers the uncertainty about it. These updates are propagated to neighboring solutions in the behavioral space by updating the Gaussian process (Methods). These updated performance and confidence distributions affect which behavior is tested next. This select-test-update loop repeats until the robot finds a behavior whose measured performance is greater than 90% of the best performance predicted for any behavior in the behavior-performance map (Methods).

We first test our algorithm on a hexapod robot that needs to walk as fast as possible (Fig. 4.1B, D). The robot has 18 motors, an onboard computer, and a depth camera that allows the robot to estimate its walking speed (Supplementary Methods). The gait is parametrized by 36 real-valued parameters that describe the amplitude of oscillation, phase shift, and duty cycle for each joint (Supplementary Methods). The behavior space is 6-dimensional, where each dimension is the proportion of time the i^{th} leg spends in contact with the ground (i.e. the duty factor) (Siciliano and Khatib 2008) (Supplementary Methods).

The created behavior-performance map contains approximately 13,000 different gaits (Supplementary Video S2 shows examples). We tested our robot in six different conditions: undamaged (Fig. 4.5A:C1), four different structural failures (Fig. 4.5A:C2-C5), and a temporary leg repair (Fig. 4.5A:C6). We compare the walking speed of resultant gaits with a widely-used, classic, hand-designed tripod gait (Siciliano and Khatib 2008) (Supplementary Methods). For each of the 6 damage conditions, we ran our adaptation step 5 times for each of 8 independently generated behavior-performance maps (with the default "duty factor" behavioral description), leading to $6 \times 5 \times 8 = 240$ experiments in total. We also ran our adaptation step 5 times on 8 independently generated behavior-performance maps defined by an alternate behavioral description ("body orientation", see Supplementary Methods) on two damage conditions (Fig. 4.5B-C), leading to $2 \times 5 \times 8 = 80$ additional experiments.

When the robot is undamaged (Fig. 4.5A:C1), our approach yields dynamic gaits that are 30% faster than the classic reference gait (Fig. 4.5B, median 0.32 m/s, 5^{th} and 95^{th} percentiles [0.26; 0.36] vs. 0.24m/s), suggesting that Intelligent Trial and Error is a good search algorithm for automatically producing successful robot behaviors, putting aside damage recovery. In all the damage scenarios, the reference gait is no longer effective ($\sim$0.04 m/s for the four damage conditions, Fig. 4.5B:C2-C5). After Intelligent Trial and Error, the compensatory gaits achieve a reasonably fast speed ($> 0.15m/s$) and are between 3 and 7 times more efficient than the reference gait for that damage condition (in m/s, C2: 0.24 [0.18; 0.31] vs. 0.04; C3: 0.22 [0.18; 0.26] vs. 0.03; C4: 0.21 [0.17; 0.26] vs. 0.04; C5: 0.17 [0.12; 0.24] vs. 0.05; C6: 0.3 [0.21; 0.33] vs 0.12).

These experiments demonstrate that Intelligent Trial and Error allows the robot to both initially learn fast gaits and to reliably recover after physical damage. Additional experiments reveal that these capabilities are substantially faster than state-of-the-art algorithms (Supplementary Fig. D.2), and that Intelligent Trial and Error can help with another major challenge in robotics: adapting to new environments (Supplementary Fig. D.3). On the undamaged or repaired

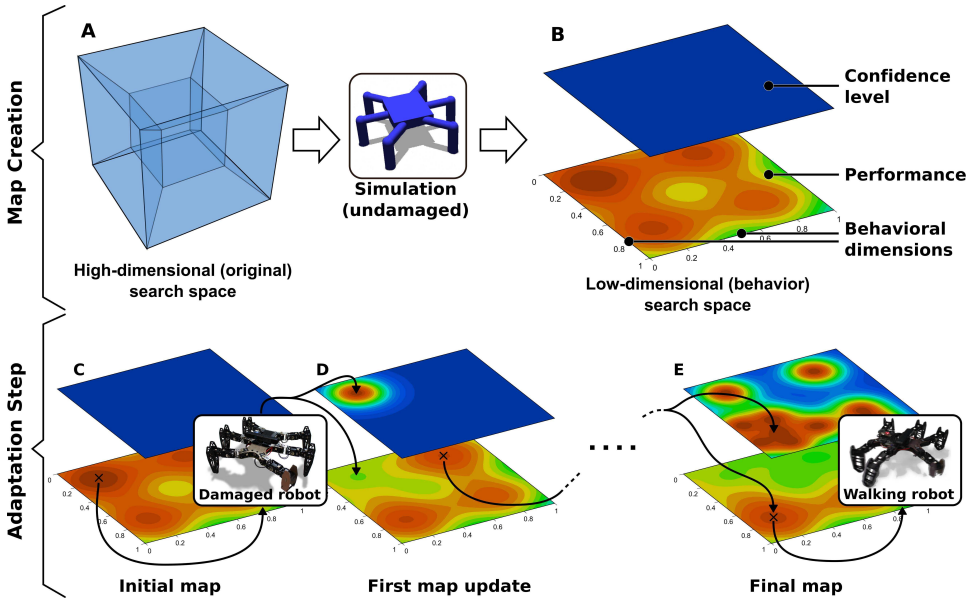


Figure 4.4. **(A & B). Creating the behavior-performance map:** A user reduces a high-dimensional search space to a low-dimensional behavior space by defining dimensions along which behaviors vary. In simulation, the high-dimensional space is then automatically searched to find a high-performing behavior at each point in the low-dimensional behavior space, creating a "behavior-performance" map of the performance potential of each location in the low-dimensional space. In our hexapod robot experiments, the behavior space is six-dimensional: the portion of time that each leg is in contact with the ground. The confidence regarding the accuracy of the predicted performance for each behavior in the behavior-performance map is initially low because no tests on the physical robot have been conducted. **(C & D) Adaptation Step:** After damage, the robot selects a promising behavior, tests it, updates the predicted performance of that behavior in the behavior-performance map, and sets a high confidence on this performance prediction. The predicted performances of nearby behaviors–and confidence in those predictions–are likely to be similar to the tested behavior and are thus updated accordingly. This select/test/update loop is repeated until a tested behavior on the physical robot performs better than 90% of the best predicted performance in the behavior-performance map, a value that can decrease with each test (Supplementary Fig. D.1). The algorithm that selects which behavior to test next balances between choosing the behavior with the highest predicted performance and behaviors that are different from those tested so far. Overall, the Intelligent Trial and Error approach presented here rapidly locates which types of behaviors are least affected by the damage to find an effective, compensatory behavior.

robot (Fig. 4.5: C6), Intelligent Trial and Error learns a walking gait in less than 30 seconds (Fig. 4.5C, undamaged: 24 [16; 41] seconds, 3 [2; 5] physical trials, repaired: 29 [16; 82] seconds, 3.5 [2; 10] trials). For the four damage scenarios, the robot adapts in approximately one minute (66 [24; 134] seconds, 8 [3; 16] trials). Our results are qualitatively unchanged when using different behavioral characterizations, including randomly choosing 6 descriptors among 63 possibilities (Fig. 4.5B-C and Supplementary Fig. D.4). Additional experiments show that reducing the high-dimensional parameter space to a low-dimensional behavior space via the behavior-performance map is the key component for intelligent trial and error: standard Bayesian optimization in the original parameter space does not find working controllers (Supplementary Fig. D.2).

We investigated how the behavior-performance map is updated when the robot loses a leg (Fig. 4.5A:C4). Initially the map predicts large areas of high performance. During adaptation, these areas disappear because the behaviors do not work well on the damaged robot. Intelligent Trial and Error quickly identifies one of the few, remaining, high-performance behaviors (Fig. 4.6 and Supplementary Fig. D.5 and D.6).

The same damage recovery approach can be applied to any robot, such as a robotic arm. We tested 14 different damage conditions with a planar, 8-joint robotic arm (Fig. 4.5D-F and Supplementary Fig. D.7). The behavior-performance map's behavioral dimensions are the x, y position of the end-effector and the performance measure is minimizing the variance of the 8 specified motor angles (Supplementary Methods). During adaptation, performance is measured as distance to the target. Like with the hexapod robot, our approach discovers a compensatory behavior in less than 2 minutes, usually in less than 30 seconds, and with fewer than 10 trials (Fig. 4.5F and Supplementary Fig. D.7).

While natural animals do not use the specific algorithm we present, there are parallels between Intelligent Trial and Error and animal learning. Like animals, our robot does not have a predefined strategy for how to cope with every possible damage condition: in the face of a new injury, it exploits its intuitions about how its body works to experiment with different behaviors to find what works best. Also like animals (Benson-Amram and Holekamp 2012), Intelligent Trial and Error allows the quick identification of working behaviors with a few, diverse tests instead of trying behaviors at random or trying small modifications to the best behavior found so far. Additionally, the Bayesian optimization procedure followed by our robot appears similar to the technique employed by humans when they optimize an unknown function (Borji and Itti 2013), and there is strong evidence that animal brains learn probability distributions, combine them with prior knowledge, and act as Bayesian optimizers (Körding and Wolpert 2004; Pouget et al. 2013).

An additional parallel is that Intelligent Trial and Error primes the robot for creativity during a motionless period, after which the generated ideas are tested. This process is reminiscent of the finding that some animals start the day

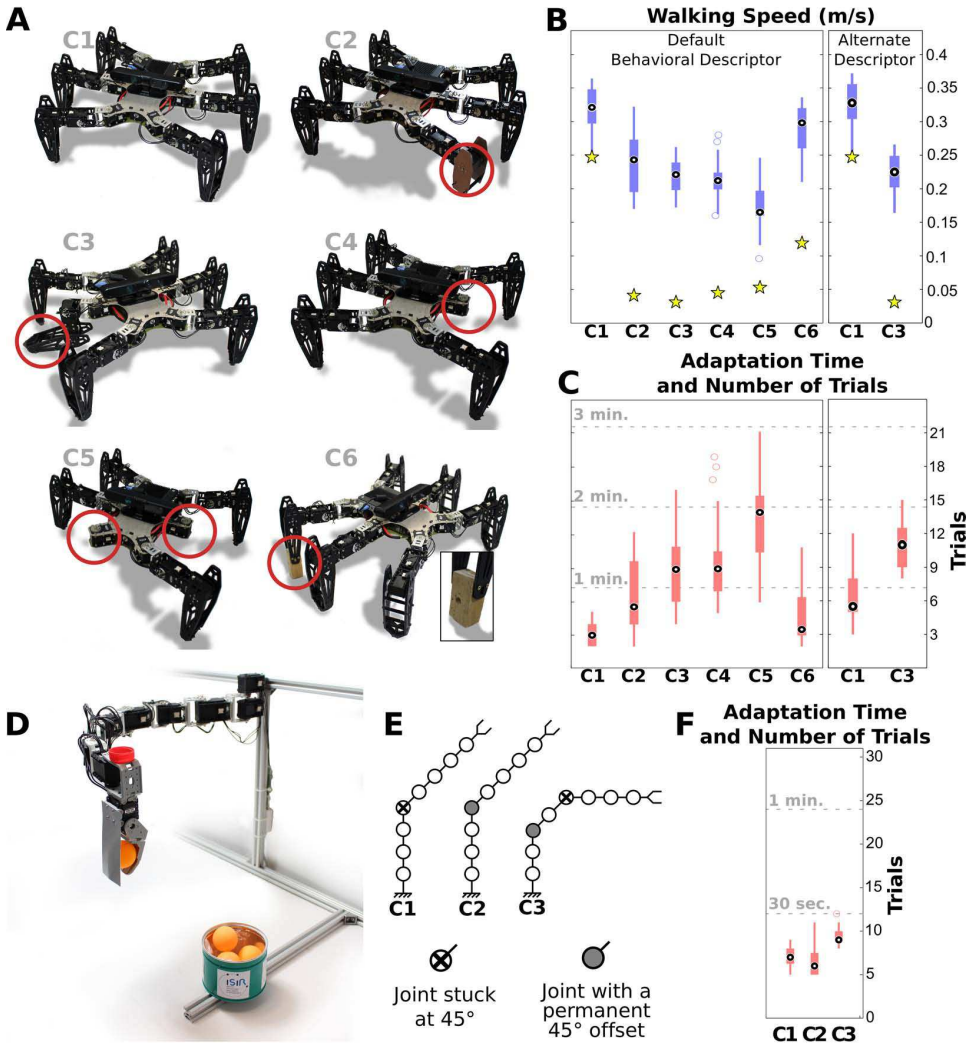


Figure 4.5. **(A) Conditions tested on the physical hexapod robot.** C1: The undamaged robot. C2: One leg is shortened by half. C3: One leg is unpowered. C4: One leg is missing. C5: Two legs are missing. C6: A temporary, makeshift repair to the tip of one leg. **(B) Performance after adaptation.** Box plots represent Intelligent Trial and Error. The central mark is the median, the edges of the box are the 25th and 75th percentiles, the whiskers extend to the most extreme data points not considered outliers, and outliers are plotted individually. Yellow stars represent the performance of the handmade reference tripod gait (Supplementary Methods). Conditions C1-C6 are tested 5 times each for 8 independently created behavior-performance maps with the "duty factor" behavior description (i.e. 40 experiments per damage condition, Supplementary Methods). Damage conditions C1 and C3 are also tested 5 times each for 8 independently created behavior-performance maps with the "body orientation" behavior description (Supplementary Methods). **(C) Time and number of trials required to adapt.** Box plots represent Intelligent Trial and Error. **(D) Robotic arm experiment.** The 8-joint, planar robot has to drop a ball into a bin. **(E) Example conditions tested on the physical robotic arm.** C1: One joint is stuck at 45 degrees. C2: One joint has a permanent 45-degree offset. C3: One broken and one offset joint. A total of 14 conditions were tested (Supplementary Fig. D.7). **(F) Time and number of trials required to reach within 5 cm of the bin center.** Each condition is tested with 15 independently created behavior-performance maps.

with new ideas that they may quickly disregard after experimenting with them (Derégnaucourt et al. 2005), and more generally, that sleep improves creativity on cognitive tasks (Wagner, Gais, et al. 2004). A final parallel is that the simulator and Gaussian process components of Intelligent Trial and Error are two forms of predictive models, which are known to exist in animals (Bongard et al. 2006; Ito 2008). All told, we have shown that combining pieces of nature's algorithm, even if differently assembled, moves robots more towards animals by endowing them with the ability to rapidly adapt to unforeseen circumstances.

MAIN METHODS (Intelligent Trial and Error)

Notations

- $\mathbf{c}$: Parameters of a controller (vector)
- $\mathbf{x}$: A location in a discrete behavioral space (i.e. a type of behavior) (vector)
- χ: A location in a discrete behavioral space that has been tested on the physical robot (vector)
- $\mathcal{P}$: Behavior-performance map (stores performance) (associative table)
- $\mathcal{C}$: Behavior-performance map (stores controllers) (associative table)
- $\mathcal{P}(\mathbf{x})$: Max performance yet encountered at $\mathbf{x}$ (scalar)

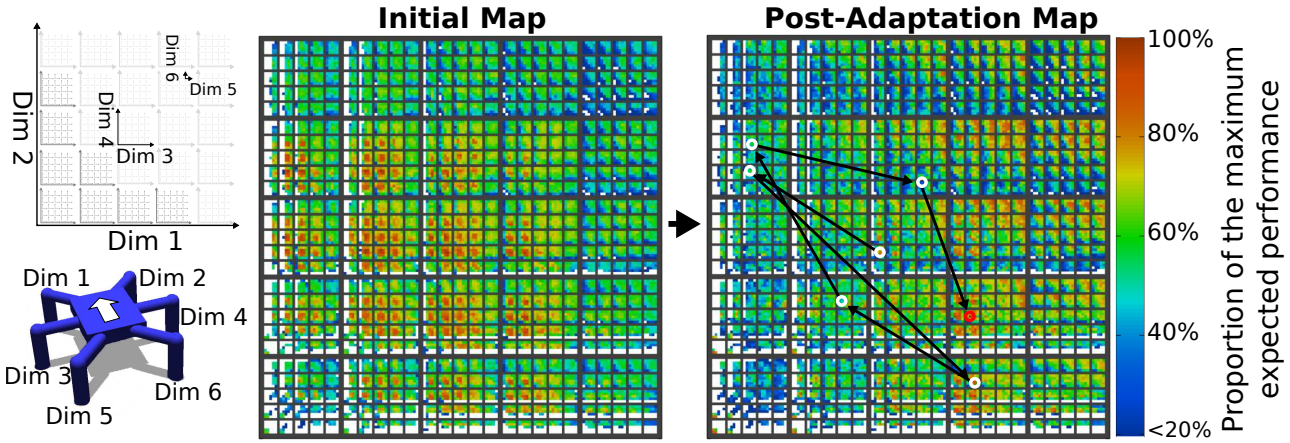


Figure 4.6. **An example behavior-performance map.** This map stores high-performing behaviors at each point in a six-dimensional behavior space. Each dimension is the portion of time that each leg is in contact with the ground. The behavioral space is discretized at five values for each dimension (0; 0.25; 0.5; 0.75; 1). Each colored pixel represents the highest-performing behavior discovered during map creation at that point in the behavior space. The matrices visualize the six-dimensional behavioral space in two dimensions according to the legend in the top-left. The behavior-performance map is created with a simulated robot (bottom left) in the Open Dynamics Engine physics simulator (http://www.ode.org). The left matrix is a pre-adaptation map produced by the map creation algorithm. During adaptation, the map is updated as tests are conducted (in this case, in the damage condition where the robot is missing one leg: Fig. 4.5A:C4). The right matrix shows the state of the map after a compensatory behavior is discovered. The arrows and white circles represent the order in which behaviors were tested on the physical robot. The red circle is the final, discovered, compensatory behavior. Amongst other areas, high-performing behaviors can be found for the damaged robot in the first two columns of the third dimension. These columns represent behaviors that least use the central-left leg, which is the leg that is missing.

- $\mathcal{C}(\mathbf{x})$: Controller currently stored in $\mathbf{x}$ (vector)
- $\chi_{1:t}$: All previously tested behavioral descriptors at time t (vector of vectors)
- $\mathbf{P}_{1:t}$: Performance in reality of all the candidate solutions tested on the robot up to time t (vector)
- $\mathcal{P}(\chi_{1:t})$: Performance in the behavior-performance map for all the candidate solutions tested on the robot up to time t (vector)
- $f()$: Performance function (unknown by the algorithm) (function)
- σ^2_{noise}: Observation noise (a user-specified parameter) (scalar)
- $k(\mathbf{x}, \mathbf{x})$: Kernel function (see section "kernel function") (function)
- $\mathbf{K}$: Kernel matrix (matrix)
- $\mathbf{k}$: Kernel vector $[k(\mathbf{x}, \chi_1), k(\mathbf{x}, \chi_2), ..., k(\mathbf{x}, \chi_t)]$ (vector)
- $\mu_t(\mathbf{x})$: Predicted performance for $\mathbf{x}$ (i.e. the mean of the Gaussian process) (function)
- $\sigma^2_t(\mathbf{x})$: Standard deviation for $\mathbf{x}$ in the Gaussian process (function)

Intelligent Trial and Error algorithm

The Intelligent Trial and Error Algorithm consists of two major steps (Supplementary Fig. D.1): the behavior-performance map creation step and the adaptation step (while here we focus on damage recovery, Intelligent Trial and Error can search for any type of required adaptation, such as learning an initial gait for an undamaged robot, adapting to new environments, etc.). The behavior-performance map creation step is accomplished via a new algorithm introduced in this paper called multi-dimensional archive of phenotypic elites (MAP-Elites), which is explained in the next section. The adaptation step is accomplished via a second new algorithm introduced in this paper called the map-based Bayesian optimization algorithm (M-BOA), which is explained in the "Adaptation Step" section below.

Behavior-performance map creation (via the MAP-Elites algorithm)

The behavior-performance map is created by a new algorithm we introduce in this paper called the multi-dimensional archive of phenotypic elites (MAP-Elites) algorithm. MAP-Elites searches for the highest-performing solution for each point in a user-defined space: the user chooses the dimensions of the space that they are interested in seeing variation in. For example, when designing robots, the user may be interested in seeing the highest-performing solution at each point in a two-dimensional space where one axis is the weight of the robot and the other axis is the height of the robot. Alternatively, a user may wish to see weight vs. cost, or see solutions throughout a 3D space of weight vs. cost vs. height. Any dimension that can vary could be chosen by the user. There is no limit on the number of dimensions that can be chosen, although it becomes computationally more expensive to fill the behavior-performance map and store it as the number of dimensions increases. It also becomes more difficult to visualize the results. We refer to this user-defined space as the "behavior space", because usually the dimensions of variation measure behavioral characteristics. Note that the behavioral space can refer to other aspects of the solution (as in this example, where the dimensions of variation are physical properties of a robot such as its height and weight).

If the behavior descriptors and the parameters of the controller are the same (i.e. if there is only one possible solution/genome/parameter set/policy/description for each location in the behavioral space), then creating the behavior-performance map is straightforward: one simply needs to simulate the solution at each location in the be-

procedure INTELLIGENT TRIAL AND ERROR ALGORITHM
Before the mission:
CREATE BEHAVIOR-PERFORMANCE MAP (VIA THE MAP-ELITES ALGORITHM IN SIMULATION)
while In mission **do**
if Significant performance fall **then**
ADAPTATION STEP (VIA M-BOA ALGORITHM)

procedure MAP-ELITES ALGORITHM
$(\mathcal{P} \leftarrow \emptyset, \mathcal{C} \leftarrow \emptyset)$ ▷ *Creation of an empty behavior-performance map (empty N-dimensional grid).*
for iter $= 1 \rightarrow I$ **do** ▷ *Repeat during I iterations (here we choose I = 40 million iterations).*
if iter < 400 **then**
$\mathbf{c}' \leftarrow$ random_controller() ▷ *The first 400 controllers are generated randomly.*
else ▷ The next controllers are generated using the map.
$\mathbf{c} \leftarrow$ random_selection($\mathcal{C}$) ▷ *Randomly select a controller c in the map.*
$\mathbf{c}' \leftarrow$ random_variation($\mathbf{c}$) ▷ *Create a randomly modified copy of c.*
$\mathbf{x}' \leftarrow$behavioral_descriptor(simu($\mathbf{c}'$)) ▷ *Simulate the controller and record its behavioral descriptor.*
$p' \leftarrow$performance(simu($\mathbf{c}'$)) ▷ *Record its performance.*
if $\mathcal{P}(\mathbf{x}') = \emptyset$ or $\mathcal{P}(\mathbf{x}') < p'$ **then** ▷ *If the cell is empty or if p' is better than the current stored performance.*
$\mathcal{P}(\mathbf{x}') \leftarrow p'$ ▷ *Store the performance of $\mathbf{c}'$ in the behavior-performance map according to its behavioral descriptor $\mathbf{x}'$.*
$\mathcal{C}(\mathbf{x}') \leftarrow \mathbf{c}'$ ▷ *Associate the controller with its behavioral descriptor:*
return behavior-performance map ($\mathcal{P}$ and $\mathcal{C}$)

procedure M-BOA (MAP-BASED BAYESIAN OPTIMIZATION ALGORITHM)
$\forall \mathbf{x} \in$ map: ▷ *Initialisation:*
$P(f(\mathbf{x})|\mathbf{x}) = \mathcal{N}(\mu_0(\mathbf{x}), \sigma_0^2(\mathbf{x}))$ ▷ *Definition of the Gaussian Process.*
where
$\mu_0(\mathbf{x}) = \mathcal{P}(\mathbf{x})$ ▷ *Initialize the mean prior from the map.*
$\sigma_0^2(\mathbf{x}) = k(\mathbf{x},\mathbf{x}) + \sigma_{noise}^2$ ▷ *Initialize the variance prior (in the common case, $k(\mathbf{x},\mathbf{x}) = 1$)*
while $\max(\mathbf{P_{1:t}}) < \alpha \max(\mu_t(\mathbf{x}))$ **do** ▷ *Iteration loop.*
$\chi_{t+1} \leftarrow \arg\max_{\mathbf{x}}(\mu_t(\mathbf{x}) + \kappa\sigma_t(\mathbf{x}))$ ▷ *Select next test (argmax of acquisition function).*
$P_{t+1} \leftarrow$ performance(physical_robot($\mathcal{C}(\chi_{t+1})$)). ▷ *Evaluation of $\mathbf{x_{t+1}}$ on the physical robot.*
$P(f(\mathbf{x})|\mathbf{P}_{1:t+1}, \mathbf{x}) = \mathcal{N}(\mu_{t+1}(\mathbf{x}), \sigma_{t+1}^2(\mathbf{x}))$ ▷ *Update the Gaussian Process.*
where
$\mu_{t+1}(\mathbf{x}) = \mathcal{P}(\mathbf{x}) + \mathbf{k}^\intercal \mathbf{K}^{-1}(\mathbf{P}_{1:t+1} - \mathcal{P}(\chi_{1:t+1}))$ ▷ *Update the mean.*
$\sigma_{t+1}^2(\mathbf{x}) = k(\mathbf{x},\mathbf{x}) + \sigma_{noise}^2 - \mathbf{k}^\intercal \mathbf{K}^{-1}\mathbf{k}$ ▷ *Update the variance.*
$\mathbf{K} = \begin{bmatrix} k(\chi_1,\chi_1) & \cdots & k(\chi_1,\chi_{t+1}) \\ \vdots & \ddots & \vdots \\ k(\chi_{t+1},\chi_1) & \cdots & k(\chi_{t+1},\chi_{t+1}) \end{bmatrix} + \sigma_{noise}^2 I$ ▷ *Compute the observations' correlation matrix.*
$\mathbf{k} = \begin{bmatrix} k(\mathbf{x},\chi_1) & k(\mathbf{x},\chi_2) & \cdots & k(\mathbf{x},\chi_{t+1}) \end{bmatrix}$ ▷ *Compute the $\mathbf{x}$ vs. observation correlation vector.*

Figure 4.7. Pseudo-code for the Intelligent Trial and Error Algorithm, the MAP-Elites algorithm, and the Map-based Bayesian Optimization Algorithm (M-BOA). Notations are described in the methods section.

havior space and record the performance. However, if it is not known a priori how to produce a controller/parameter set/description that will end up in a specific location in the behavior space (i.e. if the parameter space is of higher dimension than the behavioral space: e.g., in our example, if there are many different robot designs of a specific weight, height, and cost, or if it is unknown how to make a description that will produce a robot with a specific weight, height, and cost), then MAP-Elites is beneficial. It will efficiently search for the highest-performing solution at each point of the low-dimensional behavioral space. It is more efficient than a random sampling of the search space because high-performing solutions are often similar in many ways, such that randomly altering a high-performing solution of one type can produce a high-performing solution of a different type (see Supplementary Fig. D.9 and Supplementary Experiment S4). For this reason, searching for high-performing solutions of all types simultaneously is much quicker than separately searching for each type. For example, to generate a lightweight, high-performing robot design, it tends to be more effective and efficient to modify an existing design of a light robot rather than randomly generate new designs from scratch or launch a separate search process for each new type of design.

MAP-Elites begins by generating a set of random candidate solutions. It then evaluates the performance of each solution and records where that solution is located in the behavior space (e.g. if the dimensions of the behavior space are the height and weight, it records the height and weight of each robot in addition to its performance). For each solution, if its performance is better than the current solution at that location in the behavior-performance map, then it is added to the behavior-performance map, replacing the solution in that location. In other words, it is only kept if it

is the best of that type of solution, where "type" is defined as a location in the behavior space. There is thus only one solution kept at each location in the behavior space (keeping more could be beneficial, but for computational reasons we only keep one). If no solution is present in the behavior-performance map at that location, then the newly generated candidate solution is added at that location.

Once this initialization step is finished, Map-Elites enters a loop that is similar to stochastic, population-based, optimization algorithms, such as evolutionary algorithms (Eiben and Smith 2003): the solutions that are in the behavior-performance map form a population that is improved by random variation and selection. In each generation, the algorithm picks a solution at random via a uniform distribution, meaning that each solution has an equal chance of being chosen. A copy of the selected solution is then randomly mutated to change it in some way, its performance is evaluated, its location in the behavioral space is determined, and it is kept if it outperforms the current occupant at that point in the behavior space (note that mutated solutions may end up in different behavior space locations than their "parents"). This process is repeated until a stopping criterion is met (e.g. after a fixed amount of time has expired). In our experiments, we stopped each MAP-Elites run after 40 million iterations. Because MAP-Elites is a stochastic search process, each resultant behavior-performance map can be different, both in terms of the number of locations in the behavioral space for which a candidate is found, and in terms of the performance of the candidate in each location.

The pseudo-code of the algorithm is available in Figure 4.7.

Adaptation step (via M-BOA: the map-based Bayesian optimization algorithm)

The adaptation step is accomplished via a Bayesian optimization algorithm seeded with a behavior-performance map. We call this approach a map-based Bayesian optimization algorithm, or M-BOA.

Bayesian optimization is a model-based, black-box optimization algorithm that is tailored for very expensive objective functions (a.k.a. cost functions) (Lizotte et al. 2007; Griffiths et al. 2009; Brochu et al. 2010; Snoek et al. 2012; Borji and Itti 2013; Mockus 2013). As a black-box optimization algorithm, Bayesian optimization searches for the maximum of an unknown objective function from which samples can be obtained (e.g., by measuring the performance of a robot). Like all model-based optimization algorithms (e.g. surrogate-based algorithms (Booker et al. 1999; Forrester and Keane 2009; Jin 2011), kriging (Simpson et al. 1998), or DACE (Sacks et al. 1989; Jones et al. 1998)), Bayesian optimization creates a model of the objective function with a regression method, uses this model to select the next point to acquire, then updates the model, etc. It is called *Bayesian* because, in its general formulation (Mockus 2013), this algorithm chooses the next point by computing a posterior distribution of the objective function using the likelihood of the data already acquired and a prior on the type of function.

Here we use Gaussian process regression to find a model (Rasmussen and Williams 2006), which is a common choice for Bayesian optimization (Lizotte et al. 2007; Griffiths et al. 2009; Brochu et al. 2010; Calandra et al. 2014). Gaussian processes are particularly interesting for regression because they not only model the cost function, but also the uncertainty associated with each prediction. For a cost function f, usually unknown, a Gaussian process defines the probability distribution of the possible values $f(\mathbf{x})$ for each point $\mathbf{x}$. These probability distributions are Gaussian, and are therefore defined by a mean (μ) and a standard deviation (σ). However, μ and σ can be different for each $\mathbf{x}$; we therefore define a probability distribution *over functions*:

$$P(f(\mathbf{x})|\mathbf{x}) = \mathcal{N}(\mu(\mathbf{x}), \sigma^2(\mathbf{x})) \tag{4.1}$$

where $\mathcal{N}$ denotes the standard normal distribution.

To estimate $\mu(\mathbf{x})$ and $\sigma(\mathbf{x})$, we need to fit the Gaussian process to the data. To do so, we assume that each observation $f(\chi)$ is a sample from a normal distribution. If we have a data set made of several observations, that is, $f(\chi_1), f(\chi_2), ..., f(\chi_t)$, then the vector $[f(\chi_1), f(\chi_2), ..., f(\chi_t)]$ is a sample from a *multivariate* normal distribution, which is defined by a mean vector and a covariance matrix. A Gaussian process is therefore a generalization of a n-variate normal distribution, where n is the number of observations. The covariance matrix is what relates one observation to another: two observations that correspond to nearby values of χ_1 and χ_2 are likely to be correlated (this is a prior assumption based on the fact that functions tend to be smooth, and is injected into the algorithm via a prior on the likelihood of functions), two observations that correspond to distant values of χ_1 and χ_2 should not influence each other (i.e. their distributions are not correlated). Put differently, the covariance matrix represents that distant samples are almost uncorrelated and nearby samples are strongly correlated. This covariance matrix is defined via a *kernel function*, called $k(\chi_1, \chi_2)$, which is usually based on the Euclidean distance between χ_1 and χ_2 (see the "kernel function" sub-section below).

Given a set of observations $\mathbf{P}_{1:t} = f(\chi_{1:t})$ and a sampling noise σ^2_{noise}(which is a user-specified parameter), the Gaussian process is computed as follows (Rasmussen and Williams 2006; Brochu et al. 2010):

$$P(f(\mathbf{x})|\mathbf{P}_{1:t}, \mathbf{x}) = \mathcal{N}(\mu_t(\mathbf{x}), \sigma^2_t(\mathbf{x}))$$

where :

$$\mu_t(\mathbf{x}) = \mathbf{k}^\intercal \mathbf{K}^{-1} \mathbf{P}_{1:t}$$

$$\sigma^2_t(\mathbf{x}) = k(\mathbf{x}, \mathbf{x}) + \sigma^2_{noise} - \mathbf{k}^\intercal \mathbf{K}^{-1} \mathbf{k}$$

$$\mathbf{K} = \begin{bmatrix} k(\chi_1, \chi_1) & \cdots & k(\chi_1, \chi_t) \\ \vdots & \ddots & \vdots \\ k(\chi_t, \chi_1) & \cdots & k(\chi_t, \chi_t) \end{bmatrix} + \sigma^2_{noise} I$$

$$\mathbf{k} = \begin{bmatrix} k(\mathbf{x}, \chi_1) & k(\mathbf{x}, \chi_2) & \cdots & k(\mathbf{x}, \chi_t) \end{bmatrix} \tag{4.2}$$

Our implementation of Bayesian optimization uses this Gaussian process model to search for the maximum of the objective function $f(\mathbf{x})$, $f(\mathbf{x})$ being unknown. It selects the next χ to test by selecting the maximum of the *acquisition function*, which balances exploration – improving the model in the less explored parts of the search space – and exploitation – favoring parts that the models predicts as promising. Here, we use the "Upper Confidence Bound" acquisition function (see the "information acquisition function" section

below). Once the observation is made, the algorithm updates the Gaussian process to take the new data into account. In classic Bayesian optimization, the Gaussian process is initialized with a constant mean because it is assumed that all the points of the search space are equally likely to be good. The model is then progressively refined after each observation.

The key concept of the map-based Bayesian optimization algorithm (M-BOA) is to use the output of MAP-Elites as a prior for the Bayesian optimization algorithm: thanks to the simulations, we expect some behaviors to perform better than others on the robot. To incorporate this idea into the Bayesian optimization, M-BOA models the *difference* between the prediction of the behavior-performance map and the actual performance on the real robot, instead of directly modeling the objective function. This idea is incorporated into the Gaussian process by modifying the update equation for the mean function ($\mu_t(\mathbf{x})$, equation 4.2):

$$\mu_t(\mathbf{x}) = \mathcal{P}(\mathbf{x}) + \mathbf{k}^\intercal \mathbf{K}^{-1}(\mathbf{P}_{1:t} - \mathcal{P}(\chi_{1:t})) \tag{4.3}$$

where $\mathcal{P}(\mathbf{x})$ is the performance of $\mathbf{x}$ according to the simulation and $\mathcal{P}(\chi_{1:t})$ is the performance of all the previous observations, also according to the simulation. Replacing $\mathbf{P}_{1:t}$ (eq. 4.2) by $\mathbf{P}_{1:t} - \mathcal{P}(\chi_{1:t})$ (eq. 4.3) means that the Gaussian process models the difference between the actual performance $\mathbf{P}_{1:t}$ and the performance from the behavior-performance map $\mathcal{P}(\chi_{1:t})$. The term $\mathcal{P}(\mathbf{x})$ is the prediction of the behavior-performance map. M-BOA therefore starts with the prediction from the behavior-performance map and corrects it with the Gaussian process.

The pseudo-code of the algorithm is available in Figure 4.7.

Kernel function The kernel function is the covariance function of the Gaussian process. It defines the influence of a controller's performance (on the physical robot) on the performance and confidence estimations of not-yet-tested controllers in the behavior-performance map that are nearby in behavior space to the tested controller (Supplementary Fig. D.8A).

The Squared Exponential covariance function and the Matérn kernel are the most common kernels for Gaussian processes (Rasmussen and Williams 2006; Brochu et al. 2010; Snoek et al. 2012). Both kernels are variants of the "bell curve". Here we chose the Matérn kernel because it is more general (it includes the Squared Exponential function as a special case) and because it allows us to control not only the distance at which effects become nearly zero (as a function of parameter ρ, Supplementary Fig. D.8A), but also the rate at which distance effects decrease (as a function of parameter ν).

The Matérn kernel function is computed as follows (Matérn et al. 1960; Stein 1999) (with $\nu = 5/2$):

$$k(\mathbf{x}_1, \mathbf{x}_2) = \left(1 + \frac{\sqrt{5}d(\mathbf{x}_1,\mathbf{x}_2)}{\rho} + \frac{5d(\mathbf{x}_1,\mathbf{x}_2)^2}{3\rho^2}\right) \exp\left(-\frac{\sqrt{5}d(\mathbf{x}_1,\mathbf{x}_2)}{\rho}\right)$$
where $d(\mathbf{x}_1, \mathbf{x}_2)$ is the Euclidean distance in behavior space. (4.4)

Because the model update step directly depends on ρ, it is one of the most critical parameters of the Intelligent Trial and Error Algorithm. We selected its value after extensive experiments in simulation (Supplementary Fig. D.8 and Supplementary Method).

Information acquisition function The information acquisition function selects the next solution that will be evaluated on the physical robot. The selection is made by finding the solution that maximizes the acquisition function. This step is another optimization problem, but does not require testing the controller in simulation or reality. In general, for this optimization problem we can derive the exact equation and find a solution with gradient-based optimization (Fiacco and McCormick 1990). For the specific behavior space in the example problem in this paper, though, the discretized search space of the behavior-performance map is small enough that we can exhaustively compute the acquisition value of each solution of the behavior-performance map and then choose the maximum value.

Several different acquisition functions exist, such as the probability of improvement, the expected improvement, or the Upper Confidence Bound (UCB) (Brochu et al. 2010; Calandra et al. 2014). We chose UCB because it provided the best results in several previous studies (Brochu et al. 2010; Calandra et al. 2014). The equation for UCB is:

$$\mathbf{x}_{t+1} = \arg\max_{\mathbf{x}} (\mu_t(\mathbf{x}) + \kappa\sigma_t(\mathbf{x})) \tag{4.5}$$

where κ is a user-defined parameter that tunes the tradeoff between exploration and exploitation.

The acquisition function handles the exploitation/exploration trade-off of the adaptation (M-BOA) step. In the UCB function (Eq. 4.5), the emphasis on exploitation vs. exploration is explicit and easy to adjust. The UCB function can be seen as the maximum value (argmax) across all solutions of the weighted sum of the expected performance (mean of the Gaussian, $\mu_t(\mathbf{x})$) and of the uncertainty (standard deviation of the Gaussian, $\sigma_t(\mathbf{x})$) of each solution. This sum is weighted by the κ factor. With a low κ, the algorithm will choose solutions that are expected to be high-performing. Conversely, with a high κ, the algorithm will focus its search on unexplored areas of the search space that may have high-performing solutions. The κ factor enables fine adjustments to the exploitation/exploration trade-off of the M-BOA algorithm (the adaptation step). We describe how we chose the κ value in supplementary methods.

Supplementary Figures are available in appendix D. Supplementary methods are available in the published paper.

Evolvability signatures

Scientific context. The two algorithms introduced in the previous chapter allow walking robots to recover from damages with a few trials. However, they were purposely tested with simplistic walking controllers (sine waves) so that our contribution can be focused on the adaptation algorithm, and not the control strategy. There are two issues with these controllers: (1) they are open-loop, meaning that they cannot deal with small variations in the environment, and (2) they constrain the gaits to sine-like gaits, meaning that some damages may not be perfectly compensated if the optimal behavior requires more complex trajectories. The two algorithms can be used with any kind of controllers, and therefore any controller that can be evolved with evolutionary algorithms can replace the sine-wave walking controllers. Nevertheless, there are many ways to implement walking controllers, and even more ways to evolve them with evolutionary algorithms. How to choose the most promising approach?

Traditionally, methods to evolve robot controllers are assessed solely by the fitness score they can reach for a specific task (e.g. the maximum walking speed). However, the fitness score provides only limited insights into the encoding, in particular because it is also linked to the evolutionary algorithm that has been used. For instance, some encodings (and parameters) might bias the search towards exploration and others might be more conservative. Different evolutionary algorithms are likely to require different characteristics from the encoding.

To compare encodings in a more independent way, we introduce the concept of "evolvability signatures", which picture the statistical distribution of behavior diversity and fitness after mutations. We tested the relevance of this concept by evolving controllers for hexapod robot locomotion using five different genotype-to-phenotype mappings (direct encoding, generative encoding of open-loop and closed-loop central pattern generators, generative encoding of neural networks, and single-unit pattern generators (SUPG)). We observed a predictive relationship between the evolvability signature of each encoding and the number of generations required by hexapods to adapt from incurred damages. Our study also reveals that, across the five investigated encodings, the SUPG scheme achieved the best evolvability signature, and was always foremost in recovering an effective gait following robot damages. Overall, our evolvability signatures neatly complement existing task-performance benchmarks, and pave the way for stronger foundations for research in evolutionary computation.

Human context. This chapter is one of the main contribution Danesh Tarapore's 1-year post-doc, which I supervised. The work was performed concurrently to the work described in the previous chapter, which explains why the result of this chapter are not yet integrated with those of the previous one.

Main articles:

- Tarapore, D., and J.-B. Mouret. *Evolvability signatures of generative encodings: beyond standard performance benchmarks.* Information Sciences (2015).
- Tarapore, D. and Mouret, J.-B. (2014). Comparing the evolvability of generative encoding schemes. Artificial Life 14: Proceedings of the Fourteenth International Conference on the Synthesis and Simulation of Living Systems, MIT Press, publisher. Pages 55-62.

Related articles:

- Mouret, J.-B. and Doncieux, S. (2012). Encouraging Behavioral Diversity in Evolutionary Robotics: an Empirical Study. Evolutionary Computation. Vol 20 No 1 Pages 91-133.

Other contributors:

- Danesh Tarapore, Pierre and Marie Curie University (Post-doc)

Author contributions: designed the experiments: DT and JBM. Performed the experiments: DT; analyzed the results: DT and JBM.

In most evolutionary computation studies, fitness comparison is the main instrument used to compare different evolutionary systems and assess their progress. Such a benchmark-based comparative approach has led to incremental improvements in the robot's performance in specific tasks (e.g., for multilegged robot locomotion, the inclusion of evolved gaits on the commercial release of Sony's AIBO (Hornby, Takamura, et al. 2005; Valsalam and Miikkulainen 2008), and the progressive improvements in walking speed of the QuadraBot (Yosinski et al. 2011; Lee et al. 2013)), and is sufficient if excelling at the given function is the ultimate goal for the robot.

Nonetheless, if the evaluated task is treated as a tool to compare different evolutionary systems, and as a stepping stone to harder problems, then a mere comparison of performance does not suffice: such a methodology of comparison only provides a very limited amount of information about the behavior of the system. In particular, it does not provide any insights on, (i) how efficiently does the evolutionary process explore the search space (e.g., can it also lead to solutions for other similar tasks, or is it biased to the type of solutions useful only for a very specific task?), and (ii) what capabilities are provided to the evolved population to respond to novel situations (e.g., an unexpected breakage of the multilegged robot's limbs, or changes to its weight distribution). Furthermore, while adaptive evolutionary systems utilize a variety of population-diversity maintenance methods to operate with dynamic fitness (Jin and Branke 2005), they are mostly concerned with numerical optimization problems (e.g., (Morrison and De Jong 1999)), and

constrained to fitness-based indices to evaluate available approaches (Weicker 2002). In summary, there is a need for additional metrics when comparing evolutionary systems, especially if one is interested in the adaptive abilities provided by evolution.

In benchmark-based comparative approaches, the fitness value in an evolutionary system is often used as a proxy for the *evolvability* provided by the system (Gruau 1994; Komosiński and Rotaru-Varga 2001; Hornby, Lipson, et al. 2003; Clune, Beckmann, Ofria, et al. 2009) — the capacity of the evolved population to rapidly adapt to novel environments (Hu and Banzhaf 2010). Unfortunately, such a fitness-based proxy provides little information on the potential of the evolutionary system to generate novel phenotypes, and consequently rapidly adapt to new, untested environments. To counter the limitations of the fitness measure, we introduce a new evolvability metric that features both the quality and quantity of phenotypic variation following genetic change. With this new metric, we can *visualize* evolvability in the behavior-diversity/performance space and *predict* the performance of the population in previously untested environments. Such predictive insights on the adaptive characteristics of evolved individuals is particularly important, since it is difficult if not impossible to consider and evaluate apriori every possible scenario the robot may encounter during its operation.

We employ our new approach to "signaturize" evolvability to compare many different encodings of controllers extracted from the literature. Numerous encodings have been proposed in EC, taking inspiration from natural developmental processes, in particular, to evolve control systems for robots (e.g., (Lewis et al. 1992; Gruau 1994; Kodjabachian and Meyer 1998; Clune, Beckmann, Ofria, et al. 2009; Cheney, MacCurdy, et al. 2013; Lee et al. 2013; Morse et al. 2013)). Given the multitude of available encodings, it is crucial to compare them and understand their differences, so that the EC community can focus on the most promising ones. In the selection of encodings investigated in our study, both direct and generative schemes are considered. Direct encodings encompass a one-to-one mapping between genes and phenotypic traits, and are the simplest form of encoding thus serving as a reference for comparison (e.g., (Koos, Cully, et al. 2013)). We also evaluate the more complex generative encodings characterized by a one-to-many mapping between genes and phenotypic traits (Stanley and Miikkulainen 2002; Stanley 2007). These state of the art encodings are expected to exploit geometric information of the robot morphology to generate regular and modular phenotypic patterns (e.g., (Stanley, D'Ambrosio, et al. 2009; Clune, Stanley, et al. 2011; Morse et al. 2013)).

Overall, we investigate five encodings for the classical EC problem of legged robot locomotion (Lewis et al. 1992; Gruau 1994; Hornby, Takamura, et al. 2005; Bongard et al. 2006; Clune, Beckmann, Ofria, et al. 2009; Clune, Ofria, et al. 2009; Clune, Stanley, et al. 2011; Yosinski et al. 2011; Koos, Cully, et al. 2013; Lee et al. 2013): (1) open-loop central pattern generator (CPG) evolved with a direct encoding, (2) open-loop CPG based on nonlinear oscillators (Crespi et al. 2013), evolved with a Compositional Pattern Generator (CPPN) (Stanley 2007), (3) closed-loop CPG evolved with a CPPN, (4) artificial neural network (ANN) evolved with CPPN, inspired by HyperNEAT (Stanley, D'Ambrosio, et al. 2009; Clune, Stanley, et al. 2011), and (5) the recently introduced single-unit pattern generator (SUPG) (Morse et al. 2013).

For all these encodings, the questions are the same: are these encodings facilitating evolvability, and are the encoded individuals capable of adapting rapidly to novel situations? Furthermore, does the inclusion of a sensory feedback mechanism improve the evolvability provided, and the adaptive capabilities of the individual? To both answer these questions and evaluate the relevance of our measure of evolvability, our experiments are divided into two phases: first, we compare the evolvability signature obtained with each encoding, and consequently predict their adaptability to novel scenarios, then we evaluate the accuracy of our predictions by analyzing the ability of each encoding to effectively deal with the new scenarios (here, when some of the robot's legs are damaged).

MEASURING EVOLVABILITY

Background

The process of evolution in natural systems comes from the cooperation of, (i) exploratory genotypic variation, (ii) the corresponding phenotypic variation, and (iii) selection operators that preserves the improvements in heritable phenotypic traits over previous generations. The crucial coordination between these three forces yields the evolvability of an evolutionary system (Alberch 1991; Hu and Banzhaf 2010; Pavlicev and Wagner 2012).

The first formal definition of evolvability stems from research in computer science. In experiments with optimization algorithms using genetic programming, Lee Altenberg defined evolvability as "the ability of a population to produce variants fitter than any yet existing" (Altenberg 1994). In natural evolutionary systems, Kirschner and Gerhart (Kirschner and Gerhart 1998) describe evolvability, also called evolutionary adaptability, as "the capacity to generate heritable, selectable phenotypic variation". Marrow (Marrow et al. 1999) considers evolvability as a characteristic relevant to both artificial and natural evolutionary systems, and viewed as the capability of a population to evolve. In a summary of results from both evolutionary biology and evolutionary computer science, Wagner and Altenberg (Wagner and Altenberg 1996) view evolvability as "the ability of random variations to sometimes produce improvements". These incremental improvements are critically dependent on the Genotype-to-Phenotype encoding. Mappings facilitating evolvability, confer on the individual a robustness to lethal mutations, and exhibit a modular architecture wherein genes preferably only affect traits with the same function (Pavlicev and Wagner 2012).

Although the concept of evolvability is still very much under discussion, for our study we adopt the definition pertaining to adaptability, and the generation of major phenotypic breakthroughs (Pigliucci 2008; Clune* et al. 2013): *Evolvability is the capability of a population to rapidly adapt to novel and challenging environments.*

The measurement of evolvability conferred by an encoding is a complex and difficult problem. Phenotypic fitness or task performance is a directly observable measure, and a criteria for selection. However, the *potential* to generate a better fitness, evolvability, is a less tangible type of ob-

servable and is more difficult to measure (Hu and Banzhaf 2010). While a formal method to quantify evolvability has not yet been agreed upon in the literature, some empirical methods have been proposed notwithstanding.

In Gerhart and Kirschner's theory of facilitated variation (Gerhart and Kirschner 2007), which unifies most earlier findings of cellular and developmental processes with characteristics of evolvability, the capacity of an individual to evolve is considered to have two functional components: (i) to curtail the proportion of lethal mutations; and (ii) to decrease the number of mutations necessary to evolve diverse or novel phenotypes. Nonetheless, most studies measuring evolvability focus mainly on one only of these two aspects. Most comparisons estimate evolvability solely as the proportion of mutations that are beneficial to an individual (e.g., (Hornby and Pollack 2002; Hornby, Lipson, et al. 2003; Reisinger and Miikkulainen 2007; Clune, Beckmann, Ofria, et al. 2009; Clune, Stanley, et al. 2011)), and irrespective of the phenotypic novelty of the resultant offspring. By contrast, (Reisinger, Stanley, et al. 2005; Lehman and Stanley 2011c, 2013) quantify evolvability on the basis of the phenotypic diversity resulting from genetic change, usually without considering the quality of the change. However, both factors are essential to quantify evolvability, to discount for, (i) mutations that generate very diverse phenotypes, but prove lethal to the organism, and (ii) mutations resulting in small increments in performance that improve on a trait, but may not be able to generate novel phenotypes. Therefore, for our comparison between encodings, evolvability is visualized by characterizing both the nature of the genetic mutation, and the quantity of generated phenotypic variation.

Evolvability signatures

In this chapter, the evolvability provided by the direct and generative encodings is characterized by computing the effect of genetic mutations on, (i) the viability of the mutated individual, and (ii) the diversity of phenotypes generated. The two resultant effects are treated separately instead of being combined into a single quantitative measure of evolvability, to consider the trade-offs between them in their individual influence on evolvability (Deb 2001).

Feature 1: Deleteriousness of mutations. The first feature in our signature of evolvability is computed as the proportion decrease in the fitness of a mutated individual.

For an individual i and the mutant i', we have,

$$f_1 = \frac{F_i' - F_i}{F_i} \tag{5.1}$$

where F_i and F_i', are the fitness values before and after the application of a random genetic mutation, respectively.

The feature f_1 reflects the behavior quality following beneficial ($f_1 > 0$), neutral ($f_1 \approx 0$), and deleterious ($f_1 < 0$) genetic change. Additionally, mutations that prove lethal are associated with f_1 values less than -1, reflecting a 100% or larger decrease in individual fitness.

Feature 2: Diversity of behaviors. Following the theory of facilitated variation (Gerhart and Kirschner 2007), the second feature in our signature of evolvability evaluates the diversity of phenotypes than can be reached from a given individual. The phenotype can here be understood in two ways: in an evolutionary biology perspective, the phenotype can describe both morphological traits and behaviors (Arnold 1992; Dawkins 1999), whereas in a evolutionary robotics perspective, only morphological traits are considered to be parts of the phenotype (e.g., the parameters and the topology of an evolved networks form the phenotype) (Stanley and Miikkulainen 2003). The distinction between phenotype and behaviors avoids potential confusions when working on developmental encodings (genotype-phenotype maps), which focus on morphological traits, or when working on selective pressures, which often focus more on the behavior than on the representation (Doncieux and Mouret 2014).

In the present chapter, we focus on the diversity of *behaviors*, as done in evolutionary biology, because it best distinguishes promising individuals from the poor performers when evolving robot controllers (Lehman and Stanley 2011a; Mouret and Doncieux 2012a). For instance, all the neural-networks that are not connected to the robot's actuators lead to the same stopped-robot behavior, whereas the morphological traits (synaptic parameters and topology of the neural-network) can be widely different. A second advantage of looking at behaviors instead of morphological traits is that the behavior representation can be independent of the implementation of the controller, thus allowing us to compare the evolvability of very different controllers like CPGs, neural networks, and SUPG controllers.

Measuring behavioral differences recently received a lot of attention in evolutionary robotics because several experiments showed that explicitly encouraging the diversity of evolved behaviors helps to mitigate the issue of premature convergence (Mouret 2011b; Mouret and Doncieux 2012a; Doncieux and Mouret 2013; Doncieux and Mouret 2014). It is also the main driving force in the Novelty Search algorithm, which leads to high-performing individuals in deceptive domains by only searching for novel behaviors and disregarding task-fitness values (Lehman and Stanley 2011a). Following this interest in measuring behavioral differences, many behavioral diversity metrics have been proposed, ranging from task-specific metrics (e.g., difference between end points of a robot's trajectory), to more task-agnostic measures (e.g., differences in the robot sensory-motor flow), and various information theoretic measures (detailed review in (Doncieux and Mouret 2014) and in (Mouret and Doncieux 2012a)).

Among the investigated measures, the mutual information diversity metric provides a general approach to compute a non-linear, non-monotonic relationship between behaviors, that is applicable to numerical and symbolic behavioral representations, both in the continuous and discrete domains (Cover and Thomas 1991; Kraskov et al. 2004). For our signature of evolvability, we compute the behavioral diversity as the normalized mutual information between behaviors of an individual, before and after its genome is mutated. Such a diversity

Assuming that the behavior of an individual i can be represented as a discrete vector B_i (details in (Mouret and Doncieux 2012a)), for the behaviors B_i and B_i', of individ-

ual i and mutant i', we have:

$$H(B_i) = -\sum_{b_i \in B_i} p(b_i) \log p(b_i) \tag{5.2a}$$

$$H(B_i, B_i') = -\sum_{b_i \in B_i} \sum_{b_i' \in B_i'} p(b_i, b_i') \log p(b_i, b_i') \tag{5.2b}$$

$$f_2 = 1 - \frac{H(B_i) + H(B_i') - H(B_i, B_i')}{max(H(B_i), H(B_i')} \tag{5.2c}$$

where $H(B_i)$ is the entropy of the behavior B_i comprising the individual states b_i with probability $p(b_i)$, $H(B_i, B_i')$ is the joint entropy between behaviors B_i and B_i' with joint probability density function $p(b_i, b_i')$, and f_2 denotes the inverse of the normalized mutual information between the two behaviors.

The entropy and joint entropy are computed by first approximating $p(b_i)$ and $p(b_i, b_i')$, by counting the number of instances of each behavior state. Systematic errors in the probability estimates, consequent to the limited number of available data samples, is compensated for by adding a corrective term E to the computed entropy: $E = (S_i - 1)/2T$ (where T is the size of the temporal window over which the entropy is computed, and S_i is the number of states for which $p(b_i) \neq 0$), and $E = (S_i + S_{i'} - S_{i,i'} - 1)/2T$ to the joint entropy (where S_i, $S_{i'}$, $S_{i,i'}$, and T have an analogous meaning to the previous case) (Roulston 1999). Integrating the corrective term to the equations for entropy and joint entropy, we have:

$$H(B_i) = -\sum_{b_i \in B_i} p(b_i) \log p(b_i) + \frac{S_i - 1}{2T} \tag{5.3a}$$

$$H(B_i, B_i') = -\sum_{b_i \in B_i} \sum_{b_i' \in B_i'} p(b_i, b_i') \log p(b_i, b_i') + \frac{S_i + S_{i'} - S_{i,i'} - 1}{2T} \tag{5.3b}$$

Estimates of the corrected entropy (eq. 5.3a) and joint entropy (eq. 5.3b) are then used to update the mutual information distance between behaviors. The resulting feature f_2 represents the quantity of behavioral variation following genetic change, and is indicative of the ability of the evolutionary system to produce novel behaviors.

METHODS

Hexapod robot locomotion problem

The evolution of locomotion gaits for multilegged robots is a classical problem in evolutionary robotics, addressed in many studies utilizing both direct and generative encodings, on bipedal (e.g., (Liu and Iba 2004)), quadrupedal (e.g., (Hornby, Takamura, et al. 2005; Téllez et al. 2006; Valsalam and Miikkulainen 2008; Clune, Stanley, et al. 2011; Risi and Stanley 2013)), and hexapedal robots (e.g., (Zykov et al. 2004; Barfoot et al. 2006; Valsalam and Miikkulainen 2008)) – employed here for the comparison of different encodings. In most existing studies on evolved locomotion gaits, the performance of an individual is analyzed solely by its walking speed and the required number of generations of evolution. The rate of evolution and evolved performance has also been linked to evolvability provided by the encoding scheme, wherein controllers achieving a higher task fitness and requiring fewer generations to evolve are considered more evolvable (e.g., see (Gruau 1994; Komosiński and Rotaru-Varga 2001; Hornby, Lipson, et al. 2003; Clune, Beckmann, Ofria, et al. 2009)). While these approaches provide interesting insights on the performance of the Genotype-to-Phenotype mapping, they largely ignore its capabilities to generate viable phenotypic variations (diverse gaits in case of legged robots). However, the diversity of evolved walking gaits is important for the legged robot to recover rapidly from faults such as, the loss of one or more limbs, or motor malfunctions (Koos, Cully, et al. 2013), and for the robot to adapt to previously unencountered environmental changes. Furthermore, an efficient recovery is particular relevant for hexapedal legged robots, wherein the probability of component failure is high, consequent to the large number of moving parts.

Gait representation. The behavioral diversity in our signature of evolvability corresponds to the inter-gait diversity in the hexapod robot locomotion problem. For this diversity, a hexapod gait is represented using a gait diagram ([ )p. 379]Siciliano2008, comprising a binary matrix C of leg-surface contacts:

$$C_{tl} = \begin{cases} 1 & \text{if leg } i \text{ makes surface contact at time-step } t, \\ 0 & \text{otherwise.} \end{cases}$$

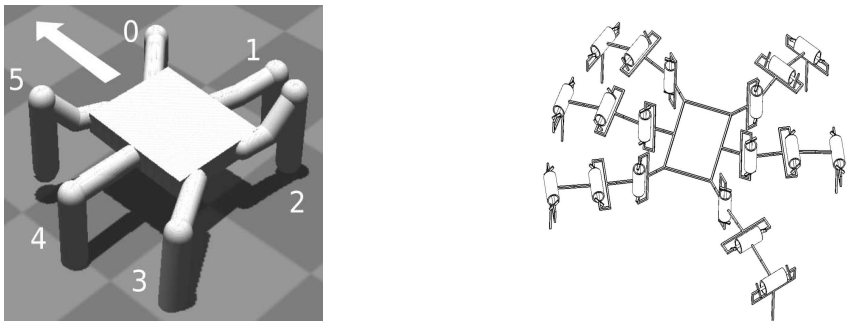
where $t \in \{0 \dots T\}$, the gait is evaluated for T time-steps, and the hexapod legs $l \in \{0 \dots 5\}$.

The hexapod gait for an individual i is represented by binary vector B_i, comprising the contacts in C concatenated in row-major order, $B_i = [C_{00}, C_{10} \dots C_{T5}]$. Diversity between two gaits is measured as the normalized mutual information between the corresponding gait vectors (eq. 5.2c).

Encoding schemes analyzed

Generative encodings for evolving our hexapod locomotion controllers are based on CPPNs (Stanley 2007). The CPPN abstracts the processes of embryonic development by determining the attributes of phenotypic components as a function of their geometric location in the individual, instead of simulating complex inter-cellular interactions and chemical morphogen gradients to determine component location (Carroll 2005). In nature, cells differentiate themselves into different lineages, influenced by their immediate environment (the epigenetic landscape, (Goldberg, Allis, et al. 2007)). Analogously, the CPPN genome outputs the fate of an organismal component as a function of its geometric coordinates in the individual.

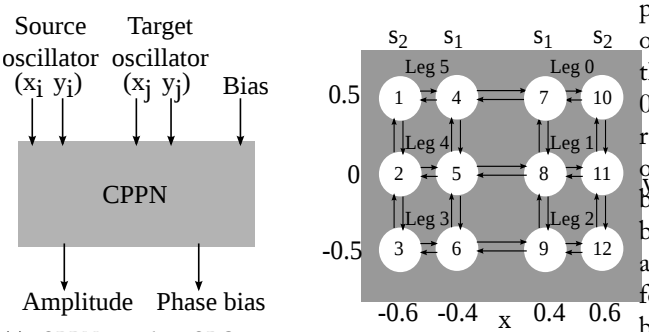
The CPPN genome is represented as a directed graph, comprising a set of *Sine*, *Gaussian*, *Sigmoid*, and *Linear* type of nodes, connected by weighted links. The node type indicates the activation function applied to the sum of its weighted inputs, to compute the node output. Selected activation functions can succinctly encode a wide variety of phenotypic patterns, such as symmetry (e.g., a Gaussian function) and repetition (e.g., a Sine function), that evolution can exploit. Mutations to the CPPN genome can change the connection weights and node type, and add or remove nodes from the graph. Consequently, the topology of the CPPN is unconstrained, open-ended, and can represent any



(a). Hexapod robot

(b). Kinematic scheme

Figure 5.1. (a) Snapshot of an 18-DOF simulated hexapod robot walking on a horizontal surface, with contacts simulated. (b) Kinematic scheme of the robot, with cylinders representing actuated pivot joints. The three servos on each leg, s_1, s_2 and s_3, are labeled in increasing order of distance to robot torso. In order to maintain the last subsegment of each leg vertical (for enhanced stability), the control signal for the third servo (s_3) is always in antiphase to that of the second servo (s_2). Consequently, the robot is reduced to a 12 DOF system, despite being actuated by 18 motors



(a). CPPN encoding CPG parameters.

(b). CPG substrate.

Figure 5.2. Encoding CPGs with CPPNs. The intrinsic amplitude A_i of each oscillator is encoded by the CPPN as a function of its position in the substrate. Phase biases $\phi_{i,j}$ of inter-oscillator couplings are determined by querying the CPPN with the coordinates of every pair of adjacent source (x_i, y_i) and target (x_j, y_j) oscillators. For every pair of adjacent oscillators, the query is made only once as $\phi_{i,j} = -\phi_{j,i}$.

possible relationship between the input coordinates of the phenotypic component and its output attributes (see details in (Stanley 2007)).

In this chapter, the CPPN genotype is mapped to four very different phenotypes to control hexapod locomotion, open-loop CPGs (Central Pattern Generators), closed-loop CPGs, ANNs (minimal HyperNEAT), and SUPGs. The SUPG is a new type of macro-neuron introduced by Morse et al. (Morse et al. 2013) to genetically encode spatio-temporal oscillatory patterns.

Open-loop Central Pattern Generator (CPG). The open-loop generatively encoded CPG system is comprised of 12 coupled amplitude-controlled phase oscillators (Ijspeert, Crespi, et al. 2007), governing the actuation of the 12 servos (s_1 and s_2, on each of 6 robot legs). Each oscillator is modeled by a set of ordinary differential equations such that the output of the oscillator γ_i exhibits a limit cycle behavior, producing a stable periodic output.

In this first generative encoding scheme evaluated, a CPPN encodes the intrinsic amplitudes A_i and the inter-oscillator phase biases $\phi_{i,j}$ of the 12 oscillators of the CPG (Fig. 5.2a). The oscillators are placed in a 2-D Cartesian grid termed the *substrate*, so that each oscillator has a distinct (x, y) coordinate, and so as to reflect the hexapod robot morphology (Fig. 5.2b). The intrinsic amplitude of each oscillator i is obtained by inputting to the CPPN the coordinates (x_i, y_i), and setting the inputs (x_j, y_j) to 0. Amplitudes output are scaled to the allowable angular range of the corresponding motors. In the CPG, adjacent oscillators are coupled together (see Fig. 5.2b). The phase bias for every pair of adjacent oscillators i and j is obtained by querying the CPPN with inputs (x_i, y_i) and (x_j, y_j), and scaling the output to range $[0, 2\pi]$. Furthermore, the following two constraints are introduced: (i) couplings are bilaterally symmetrical, i.e., $\phi_{i,j} = -\phi_{j,i}$. Therefore, for every pair of adjacent oscillators, the phase bias is queried only once; (ii) phase biases $\phi_{2,1}$, $\phi_{2,3}$, $\phi_{7,4}$, $\phi_{9,6}$, $\phi_{10,11}$ and $\phi_{12,11}$ are not queried, but computed such that the sum of phase biases in every closed loop of the CPG is a multiple of 2π (oscillators numbered in Fig. 5.2b). Therefore, the total number of CPG parameters generatively encoded by the CPPN is 23 (12 intrinsic amplitude and 11 phase bias parameters).

Closed-loop Central Pattern Generator (CPG). The second generative encoding scheme evaluated is an extension of the open-loop scheme. While the generatively encoded CPG parameters and the CPPN encoding remains the same as in the open-loop model (Fig. 5.2), the modification introduced is a sensory feedback mechanism that modulates the oscillations produced by the CPG. In this closed-loop encoding scheme, feedback signals from the touch sensors attached to each of the six legs of the hexapod trigger a phase-resetting mechanism (Aoi and Tsuchiya 2005), that adapts the oscillation period depending on the gait and the terrain. For the phase-resetting mechanism, two extreme positions of the horizontal orientation of the robot leg are introduced with respect to the robot trunk, (i) the anterior extreme position (AEP), where the swing phase transitions to the stance phase, and (ii) the posterior extreme position (PEP), where the stance phase transitions to the swing phase.

Artificial Neural Network (ANN). The third generative encoding scheme evaluated is a simplified version of HyperNEAT indirect encoding[1] In previous work, the CPPN has been used successfully to evolve modular and regular patterns in the connection space of the ANN, resulting in symmet-

[1]The CPPN is evolved with a simple multiobjective evolutionary algorithm, instead of the NEAT method (details in (Tonelli and Mouret 2013)).

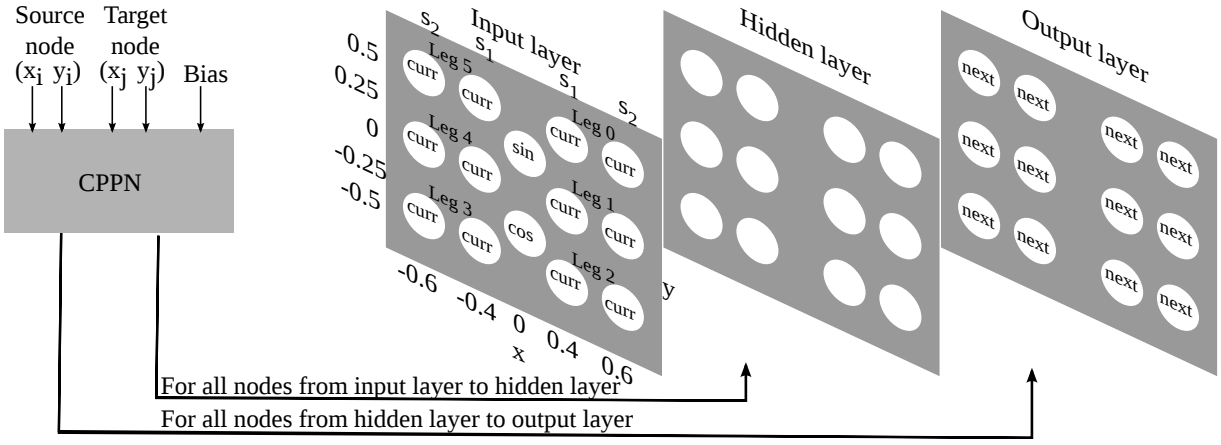


Figure 5.3. Encoding ANNs with CPPNs (inspired by (Clune, Stanley, et al. 2011; Yosinski et al. 2011; Lee et al. 2013)). Inter-neuron connection weights are encoded as function of the position of source and target neurons of each neural connection. The CPPN outputs the weights of input-hidden and hidden-output neuron connections, for each source (x_i, y_i) and target (x_j, y_j) neuron in proximal layers. The ANN is input the requested angles of the previous time-step for the first two servos ($\mathtt{s}_1$ and $\mathtt{s}_2$) on each leg, and a sine and cosine wave. The output neurons specify the new joint angles for the current time-step.

ric and coordinated gaits for both simulated and physical quadruped robots (Stanley, D'Ambrosio, et al. 2009; Clune, Stanley, et al. 2011; Yosinski et al. 2011; Lee et al. 2013). The results encourage us to include the HyperNEAT encoding in our comparative study.

The CPPNs encode the weights of a fixed topology, single-layer feedforward ANN, featuring 2-D Cartesian grids of inputs, hidden and output neurons (Fig. 5.3). Neurons of the ANN are positioned in the substrate, in accordance with the hexapod robot morphology. Using the encoding, the CPPN is iteratively queried the positions of all source (x_1, y_1) and target (x_2, y_2) neurons in proximal layers, along with a constant bias, and it outputs the corresponding weights of the input-hidden and hidden-output neuron connections.

The ANN receives as input the previously requested angles (actual angles unknown) for each of the 12 pivot joints of the hexapod robot ($\mathtt{s}_1$ and $\mathtt{s}_2$, for 6 legs). In addition, sine and cosine waves of frequency 1 Hz are also input to the ANN, to facilitate periodic oscillations at the output neurons. The output from the ANN at each time-step are 12 numbers (one for each of $\mathtt{s}_1$ and $\mathtt{s}_2$, on each of 6 legs) in interval $[-1, 1]$, that are scaled to the allowable angular range of the corresponding motors, and indicate the next position of each motor.

SUPGs oscillators In the fourth generative encoding scheme evaluated, the CPPN encodes the attributes of a SUPG. The SUPG is a macro-neuron (Fig. 5.4) that upon being triggered, produces a single cycle of a CPPN encoded activation pattern. Consequently, the repeated triggering of the SUPG results in temporal oscillations. In previous work, the SUPG outperformed HyperNEAT encodings in evolving locomotion gaits for a simulated quadruped robot (Morse et al. 2013). The resultant SUPG gaits appeared faster and steadier in extended evaluations, encouraging us to study the encoding both in terms of performance, and the evolvability provided.

In this encoding, the CPPN is input the position (x, y) of the SUPG in the substrate, and the elapsed time (in interval $[0, 1]$) since the SUPG was last triggered (Fig. 5.4a). During the period of the SUPG, its internal, individual timer ramps upwards with each simulation time-step, from an initial value of 0 to a maximum value of 1 (Fig. 5.4b). Therefore, the resultant activation pattern output by the SUPG is a function of both, its position in the substrate, and the time since the last cycle was initiated. Applying the SUPGs for hexapod locomotion, the substrate comprises 12 SUPGs positioned to reflect the robot morphology (Fig. 5.4c). The outputs of the SUPGs at each time-step specify the desired angles for the first and second servos ($\mathtt{s}_1$ and $\mathtt{s}_2$), on each leg of the robot.

In an individual SUPG, the period of the internal timer can be restarted, following the occurrence of an external trigger event. Consequently, the SUPG cycle does not need to match the length of an optimal step. Rather, the oscillation period can be adjusted depending on the gait and the terrain, by restarting the SUPG whenever its associated foot touches the ground. Therefore, the two SUPGs on each leg of our hexapod robot are triggered by the corresponding foot touching the ground, producing a closed-loop control.

At the start of the simulation, all six legs of the robot are touching the ground, resulting in all the SUPGs being triggered simultaneously. To avoid the resulting hopping gaits, the first trigger to each SUPG is delayed by an offset. The offset output of the CPPN is determined for the $\mathtt{s}_1$ SUPG on each leg by supplying its coordinates as input, and setting the time input to 0. The same offset value is also applied to the $\mathtt{s}_2$ SUPG on the leg, allowing both the oscillators on each leg to start simultaneously.

Direct encoding Locomotion controllers evolved with direct encoding are designed to be simple, wherein the actuation along each DOF of the robot is governed by the periodic signal of an open-loop amplitude controlled phase oscillator. With this encoding, hexapod leg actuation is governed by the differential equation model. There are 12 evolved parameters for the intrinsic amplitudes of oscillators, governing the actuation of the two servos $\mathtt{s}_1$ and $\mathtt{s}_2$, on each of six legs of the hexapod. In addition, 11 inter-oscillator phase bias parameters are also evolved (see Fig. 5.2 for details on constraints on phase biases). Consequently, a directly encoded controller for the hexapod robot is fully represented by 23 parameters.

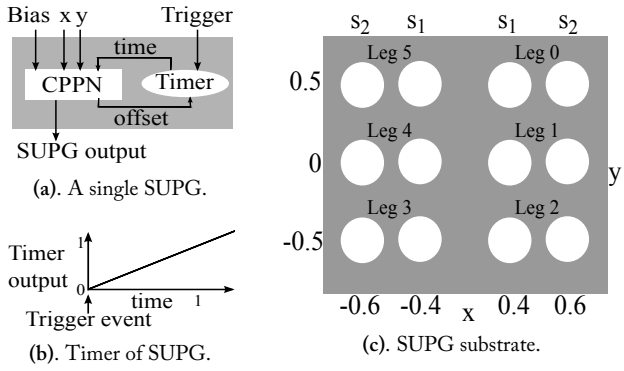


Figure 5.4. Encoding SUPGs with CPPNs (inspired by (Morse et al. 2013)). (a) The SUPG output is a function of its coordinates (x, y) in the substrate, and the elapsed time since last trigger (output of Timer). The time of first trigger is determined by an offset. (b) Once triggered, the SUPG timer ramps upward linearly from 0 to 1 and stays there, until it is re-triggered. (c) Positions of the 12 SUPGs in the substrate, outputting the desired angles for the first two servos (s_1 and s_2), on each leg of the hexapod.

EXPERIMENTS

We conducted $8,000$ generations of artificial selection in populations consisting of 100 individuals. Our aim was to evolve controllers for the hexapod robot to walk forward, evaluated for a period of 5 s (334 time-steps). The Non-dominated sorting genetic algorithm II (Deb et al. 2002) was used to simultaneously optimize the following three objectives:

$$\text{Maximize} \begin{cases} -F_i \\ -|\Theta_i| \\ \frac{1}{N}\sum_{j=0}^{j=N} D(B_i, B_j) \end{cases} \quad (5.4)$$

where for individual i in the population, F_i is the fitness computed as the distance between the final position of i and a goal located 25 m directly in front of the robot's initial position, Θ_i denotes the angle of the robot's trajectory with respect to the normal forward walking direction, $D(B_i, B_j)$ is the hamming distance between the binary gait vectors of individual i and j, and N is the size of the population.

In eq. 5.4, the first and second objectives reward individuals to walk forward large distances towards a goal, unattainable by the robot within the experiment evaluation time. The third objective is introduced to facilitate the exploration of diverse solutions and avoid premature convergence to sub-optimal solutions at local minima (Mouret and Doncieux 2012a).

Artificial selection was conducted in 20 independent replicates, for the **Direct** encoding, and the four generative encodings, (i) **CPG** (open-loop controller), (ii) **CPG-f/b** (closed-loop controller), (iii) **ANN** (minimal HyperNEAT), and (iv) **SUPG**[2]. Performance and evolvability analysis are reported for the best individual of each replicate, selected to have the highest fitness in the population, and with an angle of trajectory in the range of $\pm 1°$ (simulation source code can be downloaded from http://pages.isir.upmc.fr/evorob_db.)

Performance

In all five encodings, the performance of the best individuals rapidly increased with a quasi-stable equilibrium being reached with less than $5,000$ generations of selection (Fig. 5.5a). Additionally, individuals with evolved CPGs (Direct, CPG and CPG-f/b) converged more rapidly as compared to those encoded with the ANN and SUPG schemes (Fig. S4, generations 0 to $8,000$). After $8,000$ generations, the performance in forward displacement of the Direct, CPG, CPG-f/b, ANN and SUPG encodings was 1.92 ± 0.19, 1.79 ± 0.08, 1.68 ± 0.13, 2.93 ± 1.60 and 2.78 ± 1.43 m, respectively (Median$\pm$IQR, see Fig. 5.5b). The ANN and SUPG schemes achieved the highest performance values across all five encodings (Mann-Whitney test, $d.f. = 38$, all $p < 0.001$), but with no significant difference in performance between them. Furthermore, amongst the Direct, CPG, and CPG-f/b encodings, no significant improvement in performance was detected with a generative encoding, or with the inclusion of a feedback mechanism (Mann-Whitney test, $d.f. = 38$).

Importantly, intrinsic inter-encoding differences existed in the frequencies of oscillation governing leg actuation. The frequency of the CPG oscillations was prefixed at 1 Hz, irrespective of the sensory feedback provided, and the direct or generative nature of the encoding. By contrast, the individuals evolved with ANN and SUPG schemes were capable of expressing higher frequency oscillations (2.44 ± 1.95 Hz for ANN, and 3.81 ± 0.73 Hz for SUPG), and the frequency of the gait may itself be under selection. *Consequently, an assessment of the encodings solely on the basis of the performance is biased, and other measures are needed to compare encodings.*

Evolvability analysis

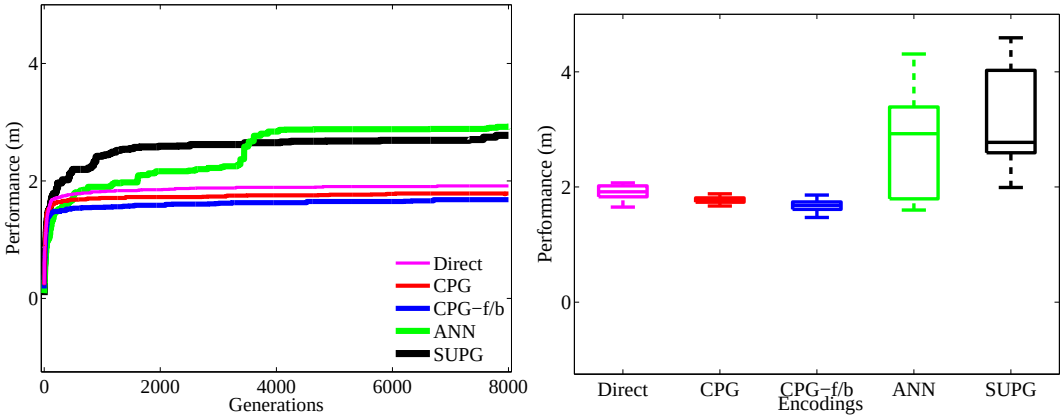
The evolvability provided by the encoding schemes is analyzed by mutating the best individual of each replicate at generation $8,000$, and reporting the following: (i) The proportion decrease in performance consequent to the mutation (eq. 5.1); and (ii) The gait diversity, computed as the mutual information between gait vectors of the original and mutated individual (eq. 5.2c). The individual is mutated at a predetermined mutation rate as used during selection, in $1,000$ separate and independent instances. Finally, a kernel density estimation (Scott 2009) is used to visualize the resultant landscape of $20,000$ data points ($1,000$ mutations $\times$ 20 replicates), pooled together from all replicates.[4]

Distinct evolvability signatures were exhibited by the Direct, CPG, CPG-f/b, ANN, and SUPG encoding schemes, after $8,000$ generations of selection (see Fig. 5.6). In evolvability analysis with a direct encoding, a conservative exploration of the phenotype, limited to solutions close to the

[2]A single evolution replicate required about 24h of computational time on a 8-cores Intel Xeon E5520 at 2.27 Ghz.

[3]On each box, the mid-line marks the median, and the box extends from the lower to upper quartile below and above the median. Whisker outside the box generally indicate the maximum and minimum values, except in case of outliers, which are shown as crosses. Outliers are data points outside of 1.5 times the interquartile range from the border of the box.

[4]Bivariate density estimation, with Gaussian type kernels over a grid of 100×100 equidistant points.



(a). Performance time-line. **(b)**. Final performance.

Figure 5.5. Performance in forward displacement for the Direct, CPG, CPG-f/b, ANN and SUPG encoding schemes: (a) Median performance for $8,000$ generations of selection; and (b) Performance of encodings at generation $8,000$.[3]

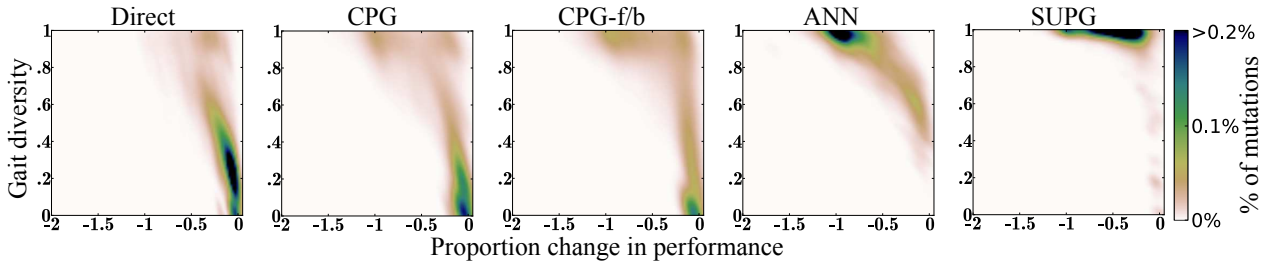


Figure 5.6. Evolvability signatures for different encodings: Gait diversity and the proportion decrease in performance, following $20,000$ independent mutations of the best individuals, for the Direct, CPG, CPG-f/b, ANN, and SUPG encoding schemes after $8,000$ generations of selection, pooled from all 20 replicates. In good evolvability signatures, mutations are located in the upper-right corner of the signature space, indicating high gait diversity, and a robustness to deleterious and lethal mutations.

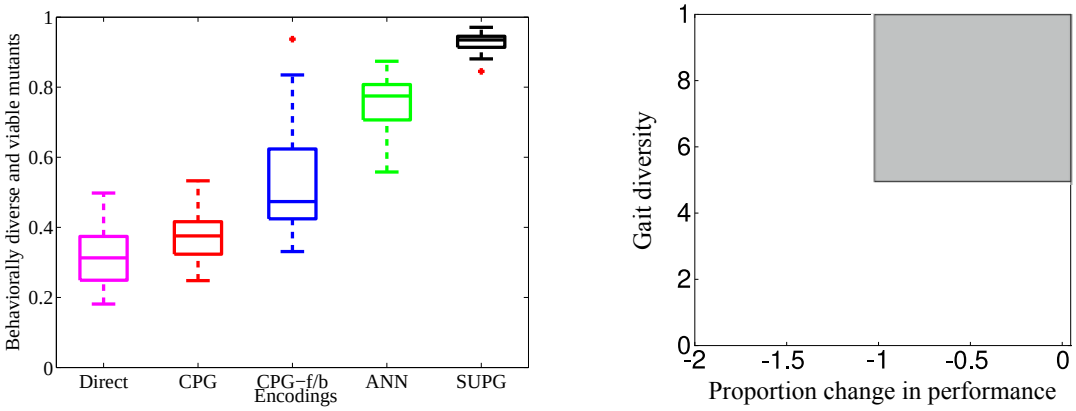
unmutated individuals was found (11.9% and 0.33, median decrement in performance and gait diversity, respectively). A generative encoding of the CPG model had only a minor effect on the evolvability provided (performance decrement of 12.7% and gait diversity of 0.36). The inclusion of a feedback mechanism in the generatively encoded CPG model resulted in more diverse gaits (0.74), but with not much change in the performance loss (16.6%) following mutations. By contrast, the generative encoded ANN and SUPG schemes were much more aggressive in the exploration of the phenotypic landscape, with the gait diversity of mutated individuals at 0.95 for ANN, and 0.99 for the SUPG encodings. However, differences existed in the severity of negative effects of mutations amongst the two encoding schemes. The ANN encoded individuals were sensitive to the effects of deleterious mutations, resulting in a 78.9% drop in performance. In comparison, individuals evolved with the SUPG encoding were much more resilient to the negative effects of mutations, with a smaller decrement of 43.1% in performance following mutation.

The evolvability provided by the encodings is further analyzed by computing the number of mutations in the evolvability signature (Fig. 5.6), that are both non-lethal and result in diverse locomotion gaits. Mutations are classified as lethal if they result in performance decrement in excess of 100% ($f_1 < -1$, see eq. 5.1), corresponding to the failure of any forward movement by the robot. Similarly, a mutation is considered to generate a diverse gait, if the inter-gait diversity exceeds 0.5 ($f_2 > 0.5$, see eq. 5.2c).

The proportion of viable and diverse-gait generating mutations was affected by the encoding scheme (see Fig. 5.7, Kruskal-Wallis test: $p < 0.001$). Across the five encodings, the SUPG scheme was most efficient at generating such mutations (Mann-Whitney test, $d.f. = 38$, all $p < 0.001$). Both the ANN and the CPG-f/b encodings led to an intermediate number of viable and diversity generating mutations (ANN significantly higher than CPG-f/b, and both different from all other encodings, all $p < 0.001$). The lowest mutation count was achieved by the Direct and CPG encoding schemes (not significantly different from each other $p = 0.03$, but different from the three other encodings, all three $p < 0.001$). Thus, across the five encoding schemes, the SUPG approach provided the highest evolvability, with the capability to explore very different but viable gaits. Additionally, with a more strict definition of viable mutants resulting in no more than 50% drop in performance, the SUPG still achieved the highest beneficial mutation count (all $p < 0.001$), with no difference between the other four encodings (see Fig. S1).

Evolvability under varying mutation intensities

In this section, we study the sensitivity our signature of the evolvability provided by the Direct, CPG, CPG-f/b, ANN and SUPG encodings with respect to the parameters of the variation operator used to generate mutants. The main questions are, if and how differences in the mutation operator affect our conceived signature of evolvability? We ran a series



(a). Beneficial mutations generated by different encodings.

(b). Marked region of beneficial mutations in evolvability signature.

Figure 5.7. Proportion of viable mutants with gait diversity in excess of 0.5, from $1,000$ independent mutations of the best individuals at generation $8,000$ in each of 20 replicates, for the Direct, CPG, CPG-f/b, ANN and SUPG encoding schemes. These beneficial mutations are counted from the shaded region of each encoding's evolvability signature.

of experiments to access evolvability, with genetic mutants generated at different intensities. Mutations were considered at the standard mutation rate and mutation step-size as used during selection (medium intensity), and at a four-fold decrease (low intensity) and a four-fold increase (high intensity) of the standard mutation operator parameters of both rate and step-size.

In order to analyze the effect of the variation operator on our signature of evolvability, in Fig. 5.8 we outline the perimeter of the evolvability signatures generated from low, medium and high intensity mutations (see Fig. S2 for interior of signature). In all five encodings, the distribution of mutants shift towards more diverse gaits and is accompanied by larger loss in performance, following increments in the mutation intensity. A 16 fold increment in the mutation rate and step-size (low to high mutation intensity) resulted in a 0.94, 0.96 and 0.98 difference in mutant gait diversity for the Direct, CPG and CPG-f/b encodings respectively (see Fig. S3a, b and c). By contrast, the ANN and SUPG schemes achieved highly diverse gaits at the lowest mutation intensity (0.94 for ANN and 0.99 for SUPG), and an increment from low to high mutation intensity only resulted in a 6.0% for ANN, and 0.4% for SUPG further increase in the mutated gaits diversity (Fig. S3d and e). The increment from low to high mutation intensity also resulted in a drop in mutant performance of 63.7%, 90.5%, 93.5%, 98.6% and 54.9%, for the Direct, CPG, CPG-f/b, ANN, and SUPG encodings respectively (Fig. S3f-j). In summary, the SUPG encoding facilitates the exploration of diverse gaits even when mutants are generated at a low mutation intensity. Furthermore, across all five encodings, the SUPG scheme provides the most resilience against the deleterious nature of high intensity mutations.

Damage recovery

The significance of our evolvability signatures of the Direct, CPG, CPG-f/b, ANN and SUPG encodings was investigated by analyzing the adaptation of the evolved robot's gait, following the removal of one or more of its legs. We expect that for the encodings registering a better evolvability signature, the corresponding evolved individuals would require fewer generations to recover an effective walking gait.

In these experiments, the new (damage recovery) populations were comprised of 100 mutated individuals of the best individual of each replicate at generation $8,000$ of selection. In separate preliminary experiments, the use of the entire population at generation $8,000$ (instead of the best individuals) did not change our results of the adaptability provided by different encodings. Individuals in the damage recovery population were mutated at the standard mutation rate and step size used during selection. A further $10,000$ generations of artificial selection was conducted on the populations of amputee hexapods for each of the following three damage scenarios: (i) an asymmetrical damage, following the removal of one leg of the robot (leg 1, Fig. 5.9a); (ii) a symmetrical damage occurs, wherein the two middle legs on either side of the robot are removed (legs 1 and 4, Fig. 5.9b); and (iii) a highly asymmetrical damage occurs consequent to the removal of the middle leg on one side and the rear leg on the opposing side of the hexapod (legs 1 and 3, Fig. 5.9c). The number of generations required to regain an effective gait and the proportion of the original performance (undamaged robot's performance at generation $8,000$) recovered for each of the three damage scenarios is analyzed.

In the $10,000$ generations of selection, all the five encodings were capable of recovering a majority of their original performance in forward displacement, irrespective of the damage to the hexapod robot (see Fig. S4). After $10,000$ generations post robot damage, the Direct, CPG, CPG-f/b, ANN and SUPG schemes all recovered the highest proportion of their original performance in the first damage scenario (0.89 ± 0.07, Median$\pm$IQR across all encodings), followed by an intermediate recovery in the second (0.78 ± 0.07), and third (0.72 ± 0.05) scenarios (Fig. 5.10a, b and c). Across all five encodings, the SUPG was most efficient in recovering its original performance in the first (1.02 ± 0.14) and second (0.99 ± 0.23) scenarios (for both, all $p < 0.001$), while in both scenarios no significant difference in recovery was registered between the remaining four encodings (Fig. 5.10a and b). In the third scenario which was the hardest, the SUPG again achieved the highest performance recovery (0.88 ± 0.3), although it was no longer significantly different between encodings (Fig. 5.10c,

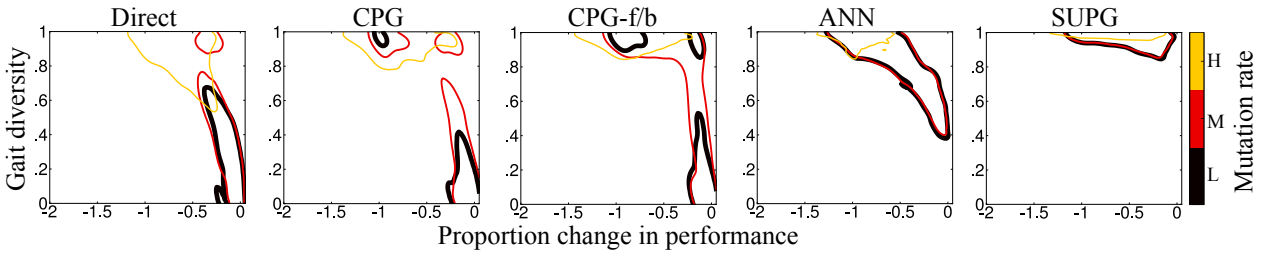
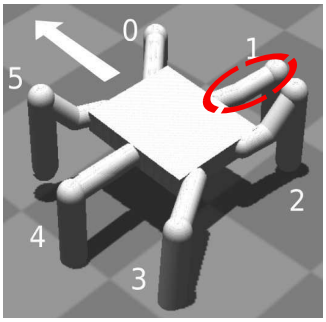
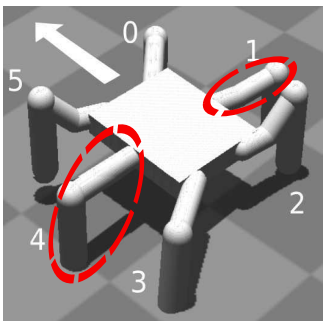


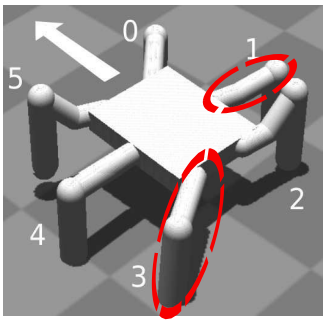
Figure 5.8. Contours of the evolvability signature for the Direct, CPG, CPG-f/b, ANN, and SUPG encoding schemes with mutations generated at low (L), medium (M), and high (H) intensity, of the best individuals at generation $8,000$ of selection. The low, medium, and high mutation intensities were quantitatively 0.25, 1, and 4 times respectively, the mutation rate and step-size used during selection. Individual contours encompass a 0.025% or higher density of mutants.



(a). Scenario 1: Removal of right-middle leg.



(b). Scenario 2: Removal of right-middle and left-middle legs.



(c). Scenario 3: Removal of right-middle and left-rear legs.

Figure 5.9. The three damage scenarios imposed on the hexapod robot (undamaged robot in Fig. 5.1).

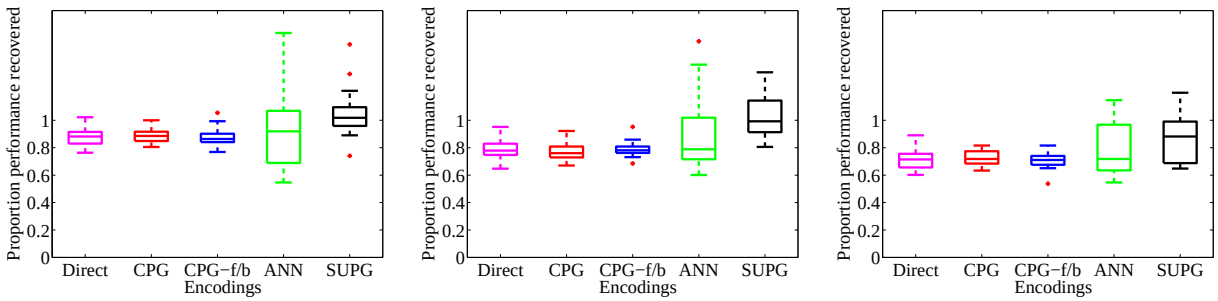
Kruskal-Wallis test: $d.f. = 4$, $p = 0.018$).

In order to analyze the time required by damaged hexapods to recover an effective gait, in Fig. 5.11 we have plotted the number of generations required to restore 85% of the original performance in forward displacement. Across the three robot-damage scenarios, amputee hexapods in the first scenario achieved the highest recovery rate (13 ± 3.25 of 20 replicates restored performance in $10,000$ generations, across all encodings), followed by an intermediate recovery rates in the second (4 ± 7.75 replicates), and third scenarios (1 ± 8.25 replicates). In the first two scenarios, the SUPG encoded individuals recovered at least an order of magnitude faster (373 and 957 generations in scenarios 1 and 2, respectively) than individuals with the Direct, CPG, CPG-f/b and ANN encodings (see Fig. 5.11a and b, all $p < 0.001$). In both scenarios, no significant difference in recovery was registered between these four encoding schemes. The SUPG encoded amputee hexapods also exhibited the fastest recovery in scenario 3 (8466.5 generations), although it was no longer significantly different from ANN encoded hexapods (see Fig. 5.11c, $p = 0.49$). Furthermore, in this scenario, all three CPG based encodings (Direct, CPG, and CPG-f/b) performed poorly, with only one replicate making the 85% mark in the $10,000$ generations of selection. In summary, across all five encoding schemes, the SUPG encoded individuals had the fastest recovery, despite being in increasingly difficult robot-damage scenarios wherein most of the individuals encoded by the other encodings failed to recover an effective walking gait.

CONCLUSION & DISCUSSION

Our results revealed a direct relationship between the estimated evolvability provided by the encodings, and the capability of the evolved individuals to adapt to severe changes in morphology, simulated by the amputation of one or more of the hexapod legs. Amongst the five encodings evaluated, the SUPGs (Morse et al. 2013) had the best evolvability signature, and their encoded individuals were also foremost to recover following sustained damages. In both the easy and the intermediate robot-damage scenarios (scenarios 1 and 2), the SUPG encoded individuals were capable of recovering 85% of their performance on the undamaged robot in all but two replicates, and did so more than an order of magnitude faster than the other four encodings. Furthermore, even in the most difficult robot-damage scenario (scenario 3), the SUPG scheme achieved the fastest recovery in the majority of the evaluated replicates.

The ANN encoding scheme (minimal HyperNEAT) was capable of producing highly diverse hexapod gaits, following genetic mutations. The high behavioral diversity generated is consistent with the earlier use of this encoding to evolve gaits for the QuadraBot robot (Clune, Beckmann,

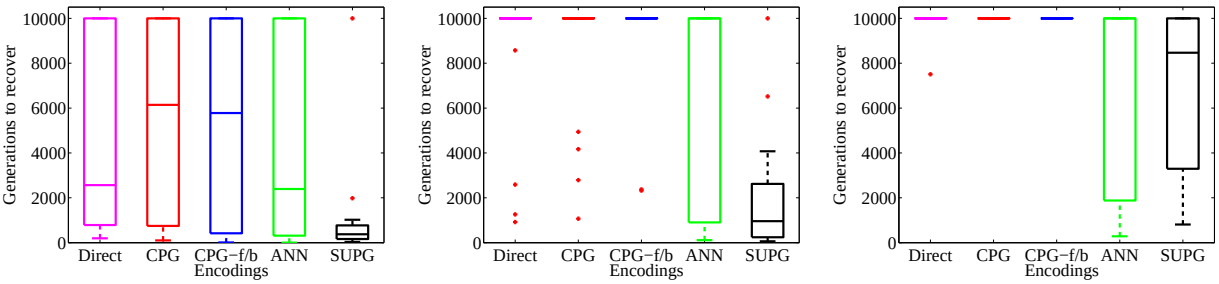


(a). Scenario 1: Removal of right-middle leg.

(b). Scenario 2: Removal of right-middle and left-middle legs.

(c). Scenario 3: Removal of right-middle and left-rear legs.

Figure 5.10. Proportion of the original performance in forward displacement restored $10,000$ generations after the three damage scenarios, across 20 replicates, for the Direct, CPG, CPG-f/b, ANN and SUPG encoding schemes.



(a). Scenario 1: Removal of right-middle leg.

(b). Scenario 2: Removal of right-middle and left-middle legs.

(c). Scenario 3: Removal of right-middle and left-rear legs.

Figure 5.11. The number of generations of selection required to restore 85% of the original performance of the undamaged hexapod in forward displacement across 20 replicates, for the Direct, CPG, CPG-f/b, ANN, and SUPG encoding schemes. In replicates unable to attain the 85% mark, the recovery time was set to the upper limit of $10,000$ generations.

Ofria, et al. 2009; Clune, Stanley, et al. 2011). However, in our ANN implementation, most of genetic mutations producing diverse gaits were highly deleterious, and resulted in little forward hexapod movement. Consequently, an estimate of evolvability solely on the basis of the generated behavioral diversity (Reisinger, Stanley, et al. 2005; Lehman and Stanley 2011c, 2013) is not reliable, and both the quality (individual viability) and quantity of phenotypic variation consequent to genetic change is required to characterize evolvability. Furthermore, the poor evolvability signature for the ANN encoding scheme is reflected in its poor recovery from sustained robot damages.

In studies on evolution of multilegged robot locomotion, the generative encodings exploit the symmetry of the robot morphology to generate regular and coordinated gait patterns that often outperform gaits evolved with direct encodings (e.g., (Clune, Stanley, et al. 2011)). Generative encodings also facilitate scalability, wherein evolution in the low-dimensional genetic search space is capable of evolving complex phenotypes comprising of many more dimensions (Stanley and Miikkulainen 2003). However, no difference in performance was registered between the directly and generatively encoded CPGs for our hexapod locomotion problem, perhaps consequent to the already low-dimensional search space for the directly encoded locomotion controllers. For example, our directly encoded CPGs for hexapod locomotion comprise 23 amplitude and phase bias parameters, in contrast to the 800 *Fixed-Topology NEAT* (FT-NEAT) encoded neural weight parameters for quadruped locomotion controllers (Clune, Beckmann, Ofria, et al. 2009). Thus, the potential benefits of phenotypic scalability in utilizing generative encoding schemes are reduced in our study.

For our signature of evolvability, we mutated the individuals with a predetermined mutation rate, tuned to allow a speedy convergence of the evolved solutions. This is a critical consideration as variations to the mutation rate can affect the viability and gait diversity of generated mutants. A comparison of the evolvability provided by encodings at low and high mutation intensities suggests that with an increase in mutation rate, the peak of the distribution of mutants shifts towards more diverse gaits with a larger decrease in task performance. However, the overall shape of the distribution, highlighting desirable regions of the evolvability landscape, remained the same for all five encodings. Importantly, across the five encodings, the SUPG scheme continued to provide the highest resilience to deleterious genetic change, despite a 16 fold increase in mutation intensity.

The high evolvability and rapid recovery provided by the SUPGs may be consequent to the closed-loop control ingrained in the encoding scheme. Such a feedback mechanism provides an adaptive period of the SUPG oscillations, that can be adjusted to the new step size of a gait more appropriate for a four or five legged robot, after the hexapod has suffered damages. Alternatively, the better performance of SUPGs may be consequent to the open-ended encoding of control signals by these oscillators. In contrast to the simple sinusoidal waves of the CPG-based schemes wherein only the signal amplitude and phase difference is encoded, no constrains are imposed on the CPPNs encod-

ing the SUPG output signals. The resulting unconstrained encoding may help for example in adjusting the duty ratio for each oscillator to match the new swing and stance phase durations of the remaining undamaged hexapod legs. Furthermore, since the inclusion of feedback in the CPG encoding (CPG-f/b) registered no improvement in task-performance, evolvability or damage recovery, a combination of the SUPG closed-loop system and the open-ended encoding of its oscillatory signal may be responsible for its high performance and adaptive capabilities.

In our evolvability signatures, the phenotypic variation from genetic change was associated with the mutual information between hexapod gaits. The diversity may also be computed for the gaits of bipedal and quadrupedal type of robots. Similarly, behavioral diversity may be computed for other benchmark problems such as, the final position of a robot in a maze navigation task (Lehman and Stanley 2011c; Mouret and Doncieux 2012a), the final positions of balls in an arena for the robot ball-collecting task (Doncieux and Mouret 2009, 2010), and a vector of board piece moves in game playing tasks (Reisinger and Miikkulainen 2007; Gauci and Stanley 2010). Consequently, our approach to estimate evolvability is easily applicable to a wide range of tasks, commonly used in evolutionary computation experiments.

The systematic building and organizing of knowledge, a requirement in any scientific discipline, can not be achieved without a wide assortment of quantifiable measures to compare and contrast concepts, hypotheses, testable explanations and predictions. In the field of evolutionary computation, task-performance has been prominently and often solely used as such a quantifiable measure to form links between the different available evolutionary systems. However, the evaluated fitness by its very nature, is limited to the specific problem for which the individual solutions are tested. By contrast, an estimate of evolvability facilitated by an evolutionary process, may be applicable to a much broader scope of scenarios, allowing the formation of a more generic relationship between existing evolutionary system implementations. An evolvability-based approach of comparison between evolutionary processes may help to extrapolate to unevaluated problem regions in-between existing evaluations, and consequently better lend itself to building a strong theoretical foundation for future research in the field.

Discussion

Bio-inspiration and common tools

We started this manuscript with a contribution about bio-inspired on-line learning (chapter 2) because we want our evolved robots to be able to adapt their behavior online. For instance, online learning is often cited as a way to cross the reality gap when evolving robot controllers (Urzelai and Floreano 2001), and it is a classic justification of evolutionary robotics: building *adaptive* robots is hard, but evolution could design the "adaptive part" for us. Thanks to our experiments, we understood how one of the current main topic of the evolutionary robotics community, namely generative and developmental systems, could open the door for evolved neural networks with good learning abilities. However, by performing these experiments, we realized how far we are from having evolved robots with advanced learning abilities. In our experiments, we only had 4 discrete inputs/outputs, no issues with non-linear classification (each lever was associated with a single stimulus), and no distal reward problem. Even in this simplistic setup, we needed days of computation on our 256-core cluster.

Robots have to solve a much more complicated challenge when they have to learn online. They usually have more input/outputs, although this issue could be mitigated by the use of a generative encoding. More importantly, they need to solve the distal reward problem with their own, evolved, learning system, whereas understanding how a neuron-based, asynchronous network can learn by reinforcement is still an open problem in neurosciences (Soltoggio and Steil 2013; Soltoggio 2015). While it is definitely possible to evolve neuro-controllers with basic learning abilities, one can doubt this approach will be able to rival the results obtained with reinforcement learning methods like Deep-Q-Learning[1] (Mnih et al. 2015).

Unexpectedly, our experiments about synaptic plasticity may open more research avenues to understand the evolution of cognitive abilities than in artificial intelligence. In particular, our results highlight the complex trade-offs between flexibility, that is, the ability to learn in many different situations, trainability, that is the ability to easily learn in some situations, and cost efficiency, that is the tendency to remove connections that bring no short-term advantage. Each part of the brain is likely to be on a different position on this 3-dimensional Pareto front. Similarly, each species is likely to have evolved toward a different trade-off, depending on its niche. Future work should explore these trade-offs and might shed new light on the origins of the structure of nervous systems.

Our second contribution is about evolving modular neural network. We explicitly targeted it to evolutionary biology, that is why we used a task that had been previously used to study the evolutionary origins of modularity (Kashtan and Alon 2005; Kashtan, Noor, et al. 2007) instead of evolving modular controllers for robots.

Nonetheless, this chapter is the main stepping stone to what may be my most important contribution to robotics and artificial intelligence so far. Advances in robotics are sometimes not where we expect them to be! When studying the evolutionary origins of modularity, we wanted to show the fitness landscape projected in a connection cost versus modularity space (figure 3.2). By visualizing this landscape, we can develop intuitions about where the "optimum" of the landscape is and how the the population moves. To solve this problem, we designed a first algorithm, called MOLE (Mouret and Clune 2012), which takes some inspiration form "divergent" search algorithms like novelty search (Lehman and Stanley 2011a; Mouret 2011b; Cully and Mouret 2015), and share some similarities with algorithms in developmental robotics (Baranes and Oudeyer 2013). We later extended it and simplified it, which led us to the MAP-Elites algorithm (see chapter 4). In effect, this algorithm allows the generation of thousands (13,000 in our locomotion experiments in chapter 4) of good ways to achieve a task and their organization in a consistent way.

This algorithm is the basis of the "Intelligent Trial and Error" algorithm (chapter 4), which has many practical implications for robotics. As a result, a tool developed to *analyze* our results for evolutionary biology enabled us to introduce an efficient method for diagnostic-free damage recovery in robotics.

In the traditional approach to bio-inspired robotics, a discovery is made in biology, an abstract model is implemented on a robot, and the robot is improved by the ideas brought by biology (Pfeiffer 2007; Meyer and Guillot 2008; Kovač 2013). By contrast, in our approach, evolutionary biology and evolutionary robotics tackle different questions but the *tools* are similar. In other words, *the inspiration is not in the results obtained in biology, but more in the journey to get them.*

This inspiration is bi-directional, as we used many of the tools and intuitions that we developed in evolutionary robotics to contribute to more biologically-oriented questions. In particular, our contribution about modularity (chapter 3) uses multi-objective evolutionary algorithms, which were developed by engineers and not biologists (Deb 2001) and which have been extensively used in our previous work (Doncieux and Mouret 2014). Most of our experiments also use behavioral diversity (Mouret and Doncieux 2009, 2012a) and novelty-based multi-objectivization (Mouret 2011b) to avoid issues with premature convergence.Overall, this methodological convergence between computational evolutionary biology and evolutionary robotics is illustrated by the fact that we use the same software framework (Mouret and Doncieux 2010) for all the experiments, both for biology, artificial intelligence, and robotics.

Figure 6.1 links our contributions and the main concepts developed in this manuscript. It can be observed that all the concepts are at least present in two contributions, and that many of them are used both in biology-oriented contributions and engineering-oriented contributions:

- all our contributions rely on multi-objective evolution-

[1] Some results have been published about using neuro-evolution (HyperNEAT, in particular) to automatically find players for the the Atari 2600 games (Hausknecht et al. 2012), that is, in the same domain as the demonstrations of the Deep Q-Learning algorithm. Nonetheless, these neuro-evolution experiments did not produce neural networks that can learn online. Instead, they evolved a network for each game.

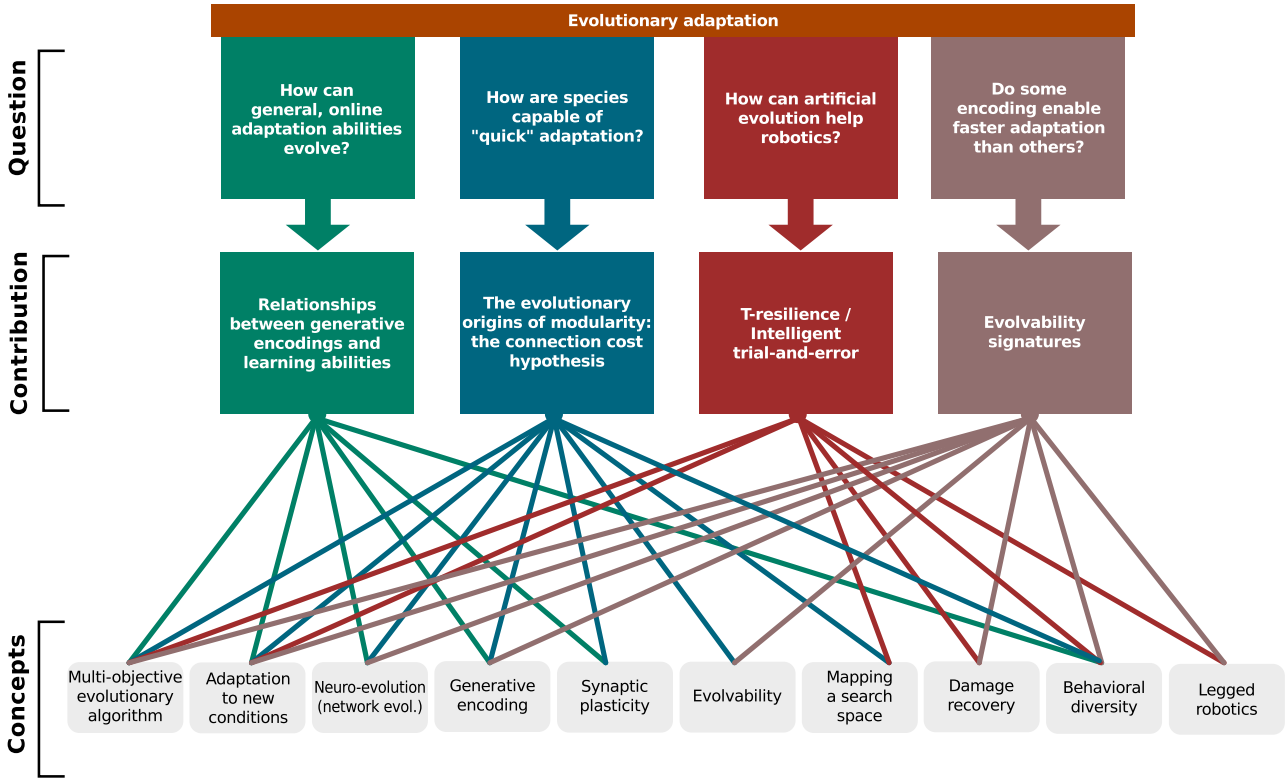


Figure 6.1. Questions, contributions, and concepts of this manuscript. All the main concepts are shared by at least two contributions, and many of them by 3 or 4 out of 4 contributions. Many concepts are shared by a contribution oriented towards biology and another contribution oriented towards robotics.

ary algorithms, in particular to tune the selective pressure (Doncieux and Mouret 2014).

- all our contributions deal with the adaptation to new conditions, sometimes with online learning (chapter 2), sometimes with evolution (chapters 4 and 5), and sometimes with machine learning (chapter 4);
- all our contributions use behavioral diversity to mitigate premature convergence (in some of them, it is only activated in a subset of the experiments);
- 3 out of 4 contributions rely on neuro-evolution (chapters 2, 3 and 5);
- 3 out of 4 contributions deal with generative encodings for neuro-evolution (chapters 2, 3, and 5);
- 2 out of 4 contributions deal with synaptic plasticity for online learning (chapters 2 and 3);
- 2 out of 4 contributions explicitly deal with evolvability as the ability of species to adapt to novel conditions in a few generations[2] (chapters 3 and 5);
- 2 out of 4 contributions use the idea that we can map the elites of a search space to explore it and to visualize it (chapters 3 and 4);
- 2 out of 4 contributions deal with damage recovery in robotics (chapters 5 and 4).

Better tools for better foundations

In spite of their computational roots, evolutionary computation and evolutionary robotics are experimental fields. The most classic approach to know whether a particular new idea is good is to launch a set of experiments with different approaches, measure the best fitness achieved and conclude that one approach is better than the others (see (Doncieux, Bredeche, et al. 2015) for a discussion on this topic). Most of the recent papers replicate each experiments many times and use statistical tests to check that the difference between each approach is actually significant. This approach is very similar to the one used in experimental fields like medecine (Bland 2000), biology (Sokal and Rohlf 1981), and psychology (Edwards 1950).

This methodology can lead to incremental improvements when the goal of the study is actually to solve the problem at hand. In evolutionary robotics, the "benchmarks" are most often potential stepping stones for more ambitious problems. For instance, nobody cares about increasing the number of balls that a robot can collect (Mouret and Doncieux 2012a), but this is a simple example of a sequential task that might require minimal cognitive abilities. In such cases, simply analyzing the fitness values might be short-sighted because the experiments involve complex processes with many different parameters and many different ideas: for the same ideas, it is very likely that the outcome of the comparison would change if the experimenter would use different parameters and different experimental benchmark problems (Karafotias et al. 2014).

As a consequence, there is a clear need for more diverse analysis tools in evolutionary computation. When designing encodings for neural networks, for example, we do not need to only know whether this encoding is "competitive" for some benchmark problem(s). We also need to characterize the networks that are the most likely, that is the biases, and how diverse are the networks that can be expressed. Are the networks modular? hierarchical? sparse? small-world?

[2]According to some definitions, everything that makes evolution faster is a major contributor to evolvability. We here use a stricter definition that focus on the adaptation abilities.

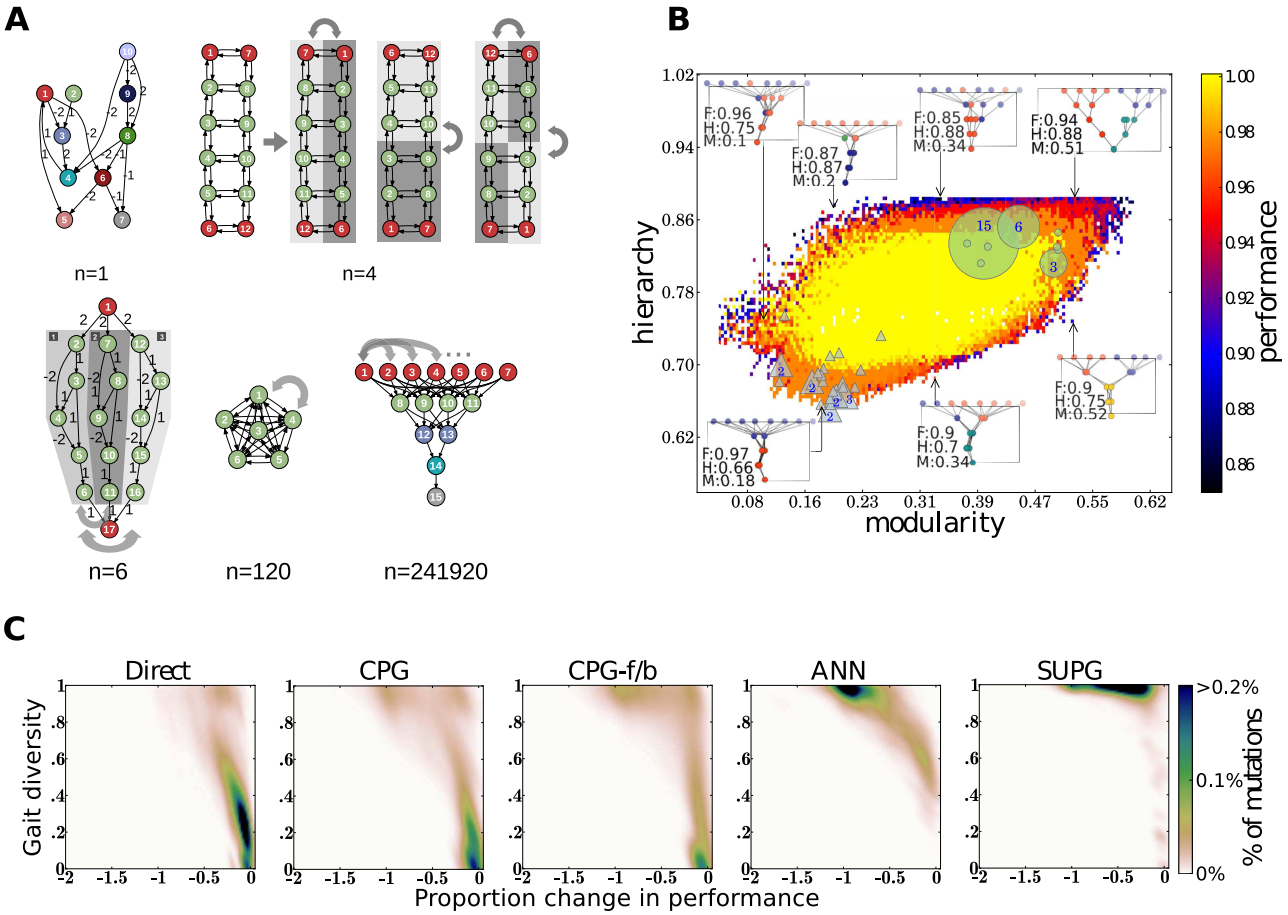


Figure 6.2. **Illustration of some of the analysis tools that we introduced recently in this manuscript. A.** Counting the number of automorphisms to measure the regularity of a network. **B.** Using MAP-Elites to picture a search landscape (here modularity versus hierarchy). **C.** Evolvability signatures of encodings for neural networks.

regular? The value returned by this kind of indicators is not bad or good, like for the fitness values, it is only a characterization. For instance, having no bias might appear to be good, because all the networks would be equally possible, but it could also appear to be bad, because some networks are likely to be more interesting than others, and some biases can make the search much faster. Since we do not know the right amount of bias for each potential feature of a neural network, the only thing we can do now is to characterize it. Similarly, in biology it is important to understand *why* some components of a treatment are critical, and not only to know it.

This is why we spent a large amount of time during the last years to develop new analysis tools: to make the field go far, *we need to understand what we do and how we do it*.

Measuring regularity in networks Most of our work involves evolving networks, most often neural networks. The analysis of networks has drawn a considerable attention in biology to understand gene regulatory networks, protein networks, and neural networks (Guimera and Amaral 2005; Alon 2006; Karlebach and Shamir 2008; Bullmore and Sporns 2009), among other biological networks. Analyzing networks is also a central question in social sciences (Scott 2012) and computer networks (Albert et al. 1999). As a result, there is now a set of widely accepted measures for modularity (Newman 2006; Leicht and Newman 2008; Fortunato 2010), hierarchy (Mones et al. 2012), and for "small-worldness" (Bullmore and Sporns 2009). These measure complement classic network characterizations like clustering coefficient, network diameter, and shortests paths (Bullmore and Sporns 2009).

Nonetheless, quantifying regularity is still an open question. Studies in evolutionary computation often used the compression ratio of the connectivity matrix (Clune, Stanley, et al. 2011; Huizinga et al. 2014), but this would in theory require to compute the compression for all the possible arrangements of nodes, whereas there are $n!$ arrangements, n being the number of nodes. In addition, a connectivity matrix is only a possible representation of a graph: it is not necessarily the easiest to compress. Other studies in evolutionary computation used measure that are specific to the genotype-phenotype map (Hornby 2005). The concept of network motifs is close (Milo et al. 2002), but current algorithms are designed to quantify small topological patterns (a few nodes) and cannot capture more general patterns like the duplication of a large sub-network. There exist a few algorithms to directly compress graph structures, but they are greedy approximations (Peshkin 2007; Hayashida and Akutsu 2010). A last source of inspiration is the literature on graph complexity (Bonchev and Buck 2005) and graph entropy (Dehmer and Mowshowitz 2011), but we did not have the opportunity to evaluate these methods yet.

Our solution to quantify the regularity is conceptually simple, easy to implement, and fast to evaluate (see chapter 2). We started with compression-based measures but we wanted to evaluate the compression ratio for all the possible arrangements. By experimenting with the enumeration of the arrangements, we observed that the number of possi-

ble arrangements depends on the regularity of the network. For instance, we noticed that if a sub-network is duplicated, then two arrangements are equivalent. An interesting way to quantify the regularity is therefore to count the number of equivalent arrangements. According to this concept, a fully connected network with uniform synaptic weights is perfectly regular, and a fully random network is perfectly irregular.

In graph theory, these equivalent arrangements are called "automorphisms", a concept which captures the idea of axes of symmetry in a graph: if two nodes can be swapped without changing the graph, then there is an axis of symmetry. More axes of symmetry means a better compression ratio because two symmetrical groups of nodes need to be described only once (Mowshowitz 1968a,b; Zenil et al. 2013). While finding the autormorphisms of a graph is NP-hard, algorithms to count them are fast in practice and implementations are available (McKay 1981; Junttila and Kaski 2007; Katebi et al. 2012). Figure 6.2A reports the number of automorphisms of a few example networks.

Overall, counting automorphisms is a useful addition to our network toolbox[3] which allows us to characterize evolved networks with fast[4] and general algorithms.

Drawing neural networks In chapter 3, we used Newman's modularity score to measure the modularity of our networks (Leicht and Newman 2008). We also needed to to measure the connection cost, that is, to sum the length of the connections. We chose to fix the positions of the inputs and the outputs so that it reflects the inherent geometry of the inputs (e.g. the shape of a retina, or the morphology of an animal), but how can we place the hidden nodes? Nervous systems minimizes the connection cost (Cherniak et al. 2004; Chklovskii 2004; Ahn et al. 2006; Wen and Chklovskii 2008; Raj and Chen 2011), therefore it makes sense to place them in the position that minimizes the connection cost. This problem can be solved exactly by quadratic programming (Chklovskii 2004)

Incidentally, solving this problem also allows us to visualize neural networks in a novel way. There exists many plotting packages for graph and networks, the most prominent one being GraphViz (Ellson et al. 2002). Most of them use a relaxation algorithms and "virtual springs" that connect nodes (Fruchterman and Reingold 1991). They usually do not work well with constraints and do not allow to fix the coordinates of some nodes. Nonetheless, the algorithm that we used to position the hidden neurons can also be used to draw networks so that the nodes are placed to minimize the total connection cost. A few networks are drawn with this algorithm on figure 6.2B (see also all the networks in chapter 3). This drawing algorithm is included in our network toolbox and we are likely to continue to rely on it for future work.

Visualizing a search space Understanding the relationships between phenotypic characteristics and fitness is central to evolutionary biology and the design of new evolutionary algorithms. Whether in computational models of evolution (Wright 1932; Kauffman 1993; Lenski, Ofria, Pennock, et al. 2003) or in evolutionary algorithms (Mitchell et al. 1992), the common approach is to perform selection based on fitness and study the phenotypes that evolve. Unfortunately, computational evolution tends to be highly convergent, meaning there is little diversity in the population and thus little variation along key phenotypic dimensions. Such a lack of diversity prevents an understanding of how fitness would change along those dimensions 'had evolution searched there'. The problem is compounded by the fact that fitness landscapes often have many local optima that populations get stuck on, which makes it difficult to know if there are higher fitness peaks in other areas of the fitness landscape that evolution failed to discover. Both biologists and engineers often spend a lot of time asking that very question, and would benefit from tools that help them answer it.

While search spaces and fitness landscapes are at the center of many discussions in evolutionary computation and evolutionary biology, we are very rarely able to *see* them because they are often too high-dimensional. The computer science literature offers plenty of options for dimension reduction and visualization of high-dimensional data (Andrews 1972; Haykin 1998; Tenenbaum et al. 2000; Kohonen 2001). Nonetheless, they are "passive" algorithm that start from a data-set and search for the best low-dimensional representation. They do not tackle the issue of generating this data set.

By contrast, to identify the fitness peaks, we must actively search for them. It is not enough to sample millions of solutions and plot them, for the same reason as random sampling is often not a good optimization algorithm: finding by chance a fitness peak is very unlikely for any large search space (in most cases, the probability of finding the best possible fitness will decrease exponentially when the number of dimensions of the search space increases).

This is why the MAP-Elites algorithm (chapters 3 and 4) is a search algorithm and not a classic dimension reduction algorithm: it aims at avoiding to evaluate many irrelevant points by focusing on the most promising ones. It could be seen as a *active* dimension reduction algorithm. Our experiments show that with an equivalent budget of evaluations, our algorithm finds much better solutions and covers the search space much better than random sampling (see Fig. D.9).

Evolvability signatures Evolvability signatures are another tool that we recently introduced; please see chapter 5.6 for a exhaustive description. Evolvability signatures have several features that make them close to the other tools that we already described in this section:

- like MAP-Elites, it is visual tool that draws 2-dimensional pictures;
- like MAP-Elites and our previous work on behavioral diversity (Mouret and Doncieux 2012a), it is based on behavioral distances between individuals;
- like our regularity measure, it reveals some intrinsic biases of encodings without needing to decide whether these biases are good or bad.

Acquiring general knowledge

Evolutionary robotics is a long-term project that needs strong foundations to continue its progress. Besides the lack of analysis tools, evolutionary robotics suffers from a lack of

[3] see https://github.com/jbmouret/network_toolbox

[4] Having fast analysis algorithms is important because we typically want to plot a value for each generation and for a whole population: we need to compute each measure many times.

general knowledge: after dozen of years of research, the field does not know many facts. In a recent review (Doncieux, Bredeche, et al. 2015), we tentatively attempted to list "what we know in evolutionary robotics". Here is what we found:

1. neural networks offer a good controller paradigm;
2. complexification is good;
3. performance-oriented fitness can be misleading;
4. selective pressure is at least as important as the encoding;
5. targeting real robots is challenging.

Many of these items could be summarized as "it is more complicated that it seems". For instance, contrary to a common intuition, performance-based fitness functions might not be the best way to search for a high-performing solution; and, contrary to the hopes of the community, targeting real robots is harder than it seems. The other items reflect more a consensus in the community than real knowledge (e.g., items 1 and 2).

One of the main reason why we know so little is that most of the published papers are proofs-of-concepts that propose a potentially interesting idea and showcase it in a custom-made experiment. Such papers are interesting to identify the promising ideas and foster discussions, but they do not bring much more knowledge than "it is possible to do X with Y". Most of these papers compare the results obtained with the proposed idea with those obtained with comparable ideas from the state-of-the-art. This is obviously good and helpful show the potential of the idea. Nevertheless, as we argued before, when the task "Y" is only a stepping stone for more ambitious challenges, concluding than one approach is better than another one on a single task is only a clue that the first approach might be more promising on the long term.

An intuitive step to improve the generality of results is to test each idea on several tasks. For instance, in our work about the evolution of modularity (chapter 3), we used four different tasks with different features. Similarly, in our work about damage recovery (chapter 4), we used two different robots (a walking hexapod and planar arm) and many different damage conditions (6 for the hexapod, 14 for the arm).

A less intuitive and less common approach is to generalize across the other directions of an evolutionary robotics experiments. In essence, any evolutionary robotics experiments involves 4 main components:

1. a task and;
2. an encoding;
3. a selective pressure (typically, a fitness function and/or some objectives);
4. an evolutionary algorithm.

Many articles test several tasks, but only a handful of them attempt to generalize across the encodings, the selective pressures, and the evolutionary algorithms.

In our work about behavioral diversity (Mouret and Doncieux 2012a), we tested 2 encodings (a direct encoding for neural networks and an Elman network whose parameters are evolved), 3 tasks, and 3 different methods to apply a selective pressure for diversity. Thanks to this approach, we were able to reach a general conclusion: a selective pressure that encourage behavioral diversity has much more impact on the success of an evolutionary robotics experiment than the encoding, regardless of the technique employed to encourage diversity.

In our contribution about synaptic plasticity (chapter 2), we used 3 different encoding, which allowed us to draw conclusions that aim at being valuable *for all the generative encodings* or, more precisely, for all the encodings that have a bias towards regularity. This approach highlights that it is possible to acquire knowledge that goes beyond "technique A works better than technique B". In effect, we expect our conclusions to be useful for any encoding, including those that have not been invented yet.

In these experiments, measuring the regularity with a technique that was independent of the encoding was key to reach a general conclusion. This illustrates that having tools to measure properties of the process or the solutions is critical to disconnect the knowledge from the specific technique used to obtain the aforementioned properties.

In our contribution about evolvability signatures (chapter 5), we studied encodings independently of the evolutionary algorithm. An interesting consequence is that different evolutionary algorithms and parameters might work best with different evolvability signatures. For instance, for damage recovery with NSGA-II, we successfully conjectured that the SUPG encoding would perform very well because a single mutation generates high-performing but very diverse behaviors. However, our preliminary experiments with SUPG and MAP-Elites suggest that SUPGs generate too much diversity for MAP-Elites and we obtained better maps with a simple direct encoding. The exact combination of encoding/algorithm/task is therefore an open question, but these experiments show that it is important to study each of them independently.

Overall, these contributions show that is is possible to reach general conclusions provided that we develop more analysis tools and make the effort of going beyond benchmark-based papers that compare setups as a whole (encodings+selective pressure+evolutionary algorithm+parameters+task).

Research project

For the next five years, I will mainly focus on applications to robotics and more precisely on using evolution-inspired algorithms for damage recovery. I will continue to collaborate with biologists to contribute to evolutionary biology, although it will not be central in my work.

TOWARDS HIGHLY-RESILIENT ROBOTS

My main objective for the next five years is to build on the success of our "Intelligent Trial and Error" algorithm (chapter 4) to propose novel algorithms that will allow autonomous robots to discover new behaviors when they are damaged or, more generally, when they face an unforeseen situation. We will continue to rely on a "divergent" evolutionary algorithm for exploration (MAP-Elites, like in chapter 4, Novelty search (Lehman and Stanley 2011a), or future variants and improvements) combined with a model-based optimization algorithm for online search (like Bayesian Optimization).

In five years, we should be able to put a broken robot in a normal room (e.g. an appartment) and it will autonomously discover a way to continue its task. For instance, a walking robot with a broken leg will have to discover in full autonomy how to walk again, for every direction, and while taking into account the obstacles.

Our ideal algorithm will have the following features:

General: the same algorithm should be able to work with minor modifications on any robot and any task, including walking robots, mobile manipulators, and humanoid-like robots.

Fast: the process should not take more than a few minutes, in particular to avoid damaging further the robot.

Creative: the process has to be as creative as possible so that the robot can find innovative behaviors when needed.

Multi-task: learning to do a simple task is often not enough. For instance, learning to walk in a single direction is useless; the robot needs to learn to walk in every direction.

Deployable: robots do not do their mission in an empty experimental rooms; they have to deal with their environment *while* they learn (e.g. learn to walk while avoiding obstacles).

Multi-objective: most robots have to optimize simultaneously several conflicting objectives, for instance maximizing the walking speed and minimizing the energy consumption. Since an infinity of trade-offs are Pareto-optimal, the learning algorithm has to find the Pareto front of the search space.

This project is funded for 2015-2020 by the European Commission (ERC ResiBots).

Challenge 1: generality

Objective: Test and refine our algorithms until the same algorithm can work on very different platforms and tasks.

Experimental setups. To ensure the generality of our results, we will perform each experiment with three different setups (figure 7.1). The various setups will allow us to evaluate how our methods scales up and ensure that they are not tied to a particular type of robot or task. One of the main technical challenges of this project is to implement our algorithms on these three very different robots and tasks, with different constraints.

- Wheeled robot + arm (7 degrees of freedom). Main task: a mobile robot with a robotic arm has to grasp balls and put them in a basket on top of the robot. *This task corresponds to a vacuum cleaning robot (e.g. a Roomba) that needs to clear the objects from the room before vacuum cleaning.* Grasping will be made easy by using the "jamming gripper" (by Empire Robotics). Controller: dynamic motion primitives (Kober and Peters 2013); damage conditions: block a motor of the arm, break one gear of a motor (i.e., make the degree of uncontrolled); reward: number of balls in the basket, measured by the robot (weight of the balls).
- Wheel-legged hybrid robot (30 degrees of freedom). Main task: locomotion in every direction; Controller: non-linear oscillators (Ijspeert, Crespi, et al. 2007); damage conditions: remove one leg, remove two legs, disconnect a motor, make one leg shorter, make one leg longer; reward: walking speed, measured onboard with a RGBD visual odometry algorithm; *This high-mobility robot is the kind of robot used for search and rescue missions.*
- Crawling iCub (up to 53 degrees of freedom). Main task: crawling in every direction; Controller: Non-linear oscillators (Degallier et al. 2008); damage conditions: loosen several cables, block one motor, disconnect one of the control board; reward: external (measured with a motion capture system). *The iCub robot will allow us to demonstrate that our approach scales to advanced robots like humanoids.*

Challenge 2: fast but creative

Objective: Constrain the search as less as possible while keeping the algorithm fast.

Background and angle of attack. We will explore two ideas: (1) using MAP-Elites with neural networks, to relax the constraints on the structure of the controllers, and (2) allow the "Intelligent Trial and Error Algorithm" to search *outside of the map*.

As an evolutionary algorithm, MAP-Elites can evolve neural networks in the exact same way as vector of parameters. We can therefore rely on evolvability signatures (chapter 5) to choose the best encoding for each task. Using evolved neural-networks will also be an interesting way to discover closed-loop controllers, instead of the naive open-loop based controllers that we used in our experiments with "Intelligent Trial and Error".

We did some preliminary tests with locomotion controllers and MAP-Elites. Our results targeted for classic evolutionary algorithms suggested that the SUPG encod-

Figure 7.1. Overview of the main experimental setups. In each setup, we will test several different damage conditions (removing a leg, loosening a cable, blocking a degree of freedom, etc.) **A.** Room clearing/ball collecting experiment, with a wheeled robot, a robotic arm, and a "jamming gripper". **B.** Locomotion experiment, with a wheel-legged hexapod robot, designed in our group (Jehanno et al. n.d.). **C.** Locomotion experiments, with the iCub robot.

ing (Morse et al. 2013) is the most promising encoding for hexapod locomotion (chapter 5). However, filling the map with SUP-G controllers was harder than expected: it seems that the evolvability provided by SUPG is good to evolve with a dynamic fitness (e.g. when there is a damage, see chapter 5), but not good to fill a map with MAP-Elites. The evolvability signatures reveals that SUPG generates controllers that are diverse and high-performing, which helps it to find controllers if a damage occurs. However, this makes it "jump" all over the map, which is not good for MAP-Elites because this algorithms relies on the assumption that good solutions will generate good solutions in the close neighborhood. This preliminary experiment reveals a strength of the analysis based on evolvability signatures: they allow us to analyze an encoding without having to know what is good and what is bad. In the future, we will continue these experiments to be able to search a map of very diverse controllers.

An important weakness of the "Intelligent Trial and Error" is that the post-damage behavior has to be present in the map. While this situation is not very likely when the map is large, it is still possible and might prevent the algorithm to find a solution for some unforeseen situations. Nonetheless, the main concept of "Intelligent Trial and Error" is to guide an online search algorithm (here Bayesian optimization) with priors that come from the simulation of the undamaged robot. This concept can be instantiated differently than with a search in the map generated by MAP-Elites.

We know that pure random sampling in a high-dimensional space will not be able to find any performance peak, let alone several ones, therefore we need some algorithm that search for these peaks. A simple idea is to put the solutions found by MAP-Elites back to the original search space, then to interpolate between them with some regression method. This would allow to know the performance peaks in the original search space. However, this technique will allow the algorithm to know the peaks, but not the valleys, which are almost as important as the peaks for the online search algorithm. We could take into account all the solutions that have been tested by MAP-Elites during the search, but, in our experiments, we usually evaluate 20 millions candidate solutions. No regression method is likely to perform well with 20 millions samples. As a consequence, the algorithm that we will have to design will need to select the points that best represent the landscape in the high-dimensional space.

A difficult, open question is how to combine these two ideas for the best creativity – using neural networks and move the search back in the original search space because any naive combination would require to be able to "interpolate" between neural networks with different structures, which is hard (if not impossible). Comparing these two ideas will at least give us some insight about *where* the creativity is needed: in the controller's structure or in the parameter space?

Challenge 3: multi-task

Objective: Extend the Intelligent Trial and Error algorithm (chapter 4) so that several variants of the task can be learned in a single learning session; for instance: learning to walk in every direction.

Background and angle of attack. Almost all published algorithms to learn motor patterns are devised to learn a single motor pattern, for instance walking forward with a legged robot. This problem is much simpler than the general problem of low-level controller learning: *learning a general controller that can accept commands issued by a higher-level system (e.g. a planning algorithm)*. This problem is typically addressed by testing controllers (or parametrized policies) with several different inputs and averaging the rewards for all the tested scenarios (e.g. (Kodjabachian and Meyer 1998; Mouret, Doncieux, and Meyer 2006)). For instance, we used an evolutionary algorithm to design a neural network that pilots a simulated flapping robot; we tested the ability of the neural network to drive the robot to 8 different targets and the reward function was essentially the sum of the distances to the targets (Mouret, Doncieux, and Meyer 2006). Unfortunately, this approach is very costly. First, it tests each candidate solution in each scenario, thus increasing the learning time by at least an order of magnitude. Second, learning a general controller is much more challenging than learning a simple, open-loop specialized one.

We will here investigate an alternative approach that is to learn a *repertoire of simple controllers* instead of a single, general controller. This method avoids the challenge of learning a complex controller; however, it typically involves as many learning processes as there are behaviors in the repertoire. When learning a gait controller, this means running the learning evolutionary algorithm for each possible target point, hence slowing down learning by a factor equal to the number of targets. In recent work (Cully and Mouret 2013; Cully and Mouret 2015), we proposed an algorithm, based

on "Novelty search with local competition" (Lehman and Stanley 2011b), that generates a repertoire of controllers in a single run of an evolutionary algorithm. We validated it on a hexapod walking task that have a controller for every direction (300 controllers). BR-Evolution is, however, an evolutionary algorithm and it is not designed to work with Bayesian optimization. In addition, it has not been tested with a damaged robot.

Our goal is to extend our "Intelligent Trial and Error" algorithm so that learning several tasks at once an be quickly achieved. Our main insight is to take advantage of the failed attempts for the task that actually perform a potentially useful variant of the task. For instance, when a robot learn to walk forward, many policies make it turn. Instead of discarding these turning gaits, we will keep and improve them so that the robot knows a *repertoire* of policies.

This kind of multi-task learning is conceptually close to concepts in transfer learning (Taylor and Stone 2009; Doncieux 2013), since it corresponds to transferring the knowledge acquired to learn one task (e.g. going forward) to make it easier to learn another one (e.g. turning). Multi-task learning also shares some similarities with intrinsically motivated learning and questions in developmental robotics because multi-task learning can be seen as the discovery of the capabilities of a body (Baranes and Oudeyer 2013; Benureau and Oudeyer 2013; Moulin-Frier and Oudeyer 2013). We will compare our ideas to these algorithms and possibly take inspiration from them.

Challenge 4: deployability

Objective Demonstrate that robots can recover from damage in full-autonomy, without starting each trials in the same position.

Background and angle of attack. The current learning experiments in robotics are mostly episodic (Kober and Peters 2013): the robot tries a policy, computes the reward using the final state, is reset to its initial state, and starts evaluating the next policies by starting in the exact same initial position. This is, for instance, the case of the experiments in which a robot learns to play tennis table (Kober and Peters 2013), and in most of the experiments that involve learning walking gaits (Kohl and Stone 2004; Hornby, Takamura, et al. 2005; Lizotte et al. 2007; Koos, Cully, et al. 2013).

Ideally, we would like robots to learn in a non-episodic, on-policy way, that is the task of the robot never ends and rewards collected during the task are distributed to each atomic action that led to the reward. However, this kind of non-episodic learning is difficult to achieve with direct policy search algorithm, especially in continuous search spaces.

We will here implement an intermediate between episodic learning and non-episodic learning: semi-episodic learning. The policy will still evaluated by episode, but the best variant will be selected. For instance, in a walking task, the robot could possess a repertoire of variants of the walking gait, each variant going to a different direction. When the robot faces an obstacle, it should select the variant that it expects to make it avoid the obstacle. The main challenge here is to extend Bayesian optimization to not only predict performance, but also the modifications of the state of the robot, like the final direction of the robot. For example, if the legged robot expected to turn left but actually turned right, it needs to update the model to account for both the unexpected performance and the unexpected direction.

Semi-episodic learning will extend the results of multi-task learning. It will be an important key for the final demonstration of this project: fully autonomous damage recovery, without any intervention by the experimenters.

Challenge 5: multi-objective search

Objective Extend the "Intelligent Trial and Error" algorithm to search for the set of Pareto-optimal solutions instead of the highest-performing solution.

Background. Robots often have to optimize several conflicting objectives. Typical examples are energy expenditure versus walking speed for a walking robot, and accuracy versus speed for a manipulator. In many cases, the robot (or the robot's designer) does not have a precise idea of the ideal trade-off, therefore the time spent to optimize a particular combination of objective is easily wasted. In some cases, the ideal trade-off might depend on the context. For instance, a walking robot might want to trade energy for walking speed if it senses a danger.

Multi-objective optimization is classically formalized in term of Pareto dominance: a candidate solution x_1 dominates a candidate solution x_2 if and only if it is not worse for all the objectives and better for at least one objective; but if x_1 is better than x_2 for some objectives and worse than for others, x_1 does not dominate x_2 and x_2 does not dominate x_1; in this case, x_1 and x_2 are equally good trade-offs. The goal of multi-objective optimization is to find the Pareto front, that is, the non-dominated part of the search space. Evolutionary algorithms have proven to be especially well suited for multi-objectivization (Deb 2001; Coello et al. 2007) and we used them for most of our work (Mouret 2011b; Mouret and Doncieux 2012a; Clune* et al. 2013; Koos, Cully, et al. 2013; Koos, Mouret, et al. 2013; Tonelli and Mouret 2013). Nonetheless, as with all evolutionary algorithms, multi-objective evolutionary algorithms need many evaluations before converging to an accurate estimate of the Pareto front.

Multi-objective reinforcement learning is a more recent question but it is starting to attract interest (Vamplew et al. 2011). Only a handful articles investigated some multi-objective reinforcement learning with real robots in mind (Tesch et al. 2011; Ahmadzadeh et al. 2014; R. Ariizumi et al. 2014). The main challenge is that learning with a robot is already difficult because it requires many trials, but finding a Pareto is likely to require even more trials.

We will take inspiration from our "Intelligent Trial and Error" algorithm to explore as much as possible in simulation. To do so, we will need to design a multi-objective variant of MAP-Elites. We will then combine it with multi-objective Bayesian optimization (Tesch et al. 2011, 2013) or surrogate-based evolutionary algorithms like ParEGO (Knowles 2006). We will focus our attention on hypervolume-based methods that are made possible by recent theoretical advances (Couckuyt et al. 2014; Hupkens et al. 2015).

THE EVOLUTIONARY ORIGINS OF BEHAVIORAL GENERALIZATION

This part of my project will be carried out in collaboration with Jean-Baptiste André (Institut des Sciences de l'Evolution - CNRS - Montpellier) and Nicolas Bredeche (ISIR, Université Pierre and Marie Curie).

Understanding adaptive breakthrough (or innovation) is one of the main question of evolutionary biology (Kirschner and Gerhart 2006). The objective of this project is to shed new light on evolutionary innovation *in the case of behavioral innovations*, by focusing on the role of generalization (Parter et al. 2008; Watson et al. 2014) – the ability for individuals of a given species to behave in an adaptive fashion in an environment that has never been met before by their species. To do so, we will leverage our work on synaptic plasticity, which was about the origins of general learning abilities, on Bayesian optimization (chapter 2 and 4) because it is an efficient approach for online learning, and we will combine them with environment-driven artificial evolution (Bredeche et al. 2012), because it is an interesting model of natural evolution that can explicitly take the environment into account.

Most adaptive functions in living species require the coordinated action of several traits. Hence, the adaptive modification of a function generally implies the correlated modifications of several traits together, which is unlikely to occur under the effect of independent random mutations. As a result, the origin of evolutionary novelties is still a challenge to understand (Wagner and Altenberg 1996; Pigliucci 2008). The occurrence of functional novelties is greatly facilitated, however, if the modification of one trait can be accompanied automatically with a correlated modification of other traits, keeping the ensemble functional. For instance, the size of the visual cortex must always remain perfectly fine-tuned to the size of the retina. Hence, innovations in the visual system require a parallel modification of both sides. This is facilitated, however, because the visual cortex develops under the control of the retina itself, such that it becomes automatically larger in response to mutations affecting the retina (West-Eberhard 2003).

Applying this principle in the case of behavior, *innovations are facilitated in one aspect of behavior if other aspects of behavior respond in a novel adaptive manner in the face of the first modification*. In other words, this implies that cognitive mechanisms generalize, i.e. that they still work in a relevant manner in novel situations, beyond the situations for which they have been selected by natural selection in the past. In this project, we aim to show that generalization is a key concept in biological evolution, and that it plays a key role to understand the origin of complex, intelligent, behaviors.

The evolutionary role of generalization is related to the notion of exaptation in biological evolution (Gould and Vrba 1982): When a given cognitive mechanism is generalizable, it can be built by natural selection to do one thing, and eventually perform another. The difference with exaptation is twofold, however. First, the concept of generalization helps make sense in a principled manner of the cases in which exaptation occurs, rather than relying on "just-so" stories (Watson et al. 2014). Second, whereas in exaptation a system changes function, from function A to function B, in generalization a system can be used in many new functions, beyond what it has initially been made for (Fig. 7.2). Hence, generalization can systemically boost behavioral innovations.

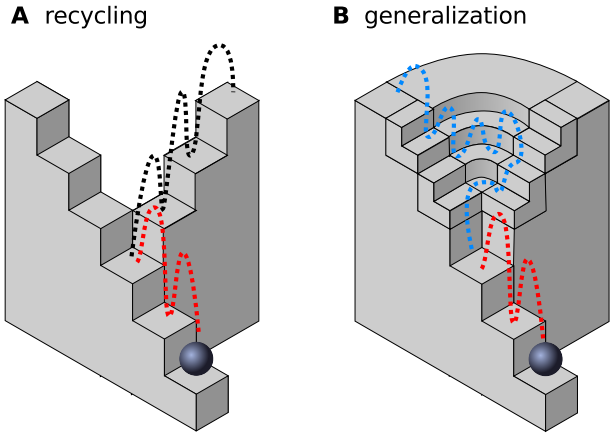


Figure 7.2. illustration of the conceptual difference between exaptation (recycling of functional modules) and generalization.

The evolutionary role of generalization is also related to the concept of robustness (Wagner 2008, 2013). In the evolution of physical traits (as opposed to behavioral traits), evolutionary innovations are considered to be primarily facilitated by the ability of the other parts of the organism to maintain their homeostasis in the face of random mutations, so-called robustness (Wagner 2008). The case of behavioral traits is different because their function is not to remain constant but to respond to the variability of the environment. Hence, whereas physical innovations are facilitated by robustness, behavioral innovations are facilitated by generalization.

We will focus on two main challenges which are stepping stones to the overall objective of understanding the evolutionary origins of generalization in cognitive systems:

- proposing a situated and individual-based model of evolution that can take learning abilities into account;
- testing the effect of environmental factors on the emergence and maintenance of generalization.

Challenge 1: an individual-based model that combines evolution and learning

Learning is an important cognitive mechanism to enable generalization, but studying the combination of learning and evolution is challenging because they both are complex processes that are not fully understood and therefore not easily abstracted. An additional challenge is the computational time because the experimenter needs to give the time for the agents to learn (their simulated life has to be long enough). Testing the survival of each individual is therefore likely to require a lot of time-steps in a simulation. Last, we want to study the influence of the environment (its complexity, its variability, etc.) on learning abilities; therefore, we need to incorporate the environment in an appropriate way.

Our starting point will be the mEDEA algorithm (Bredeche et al. 2012), in which a population of robots can move, exchange genomes, and "die". This evolutionary algorithm is an interesting model because (1) robots are situated in an explicit environment that we can modify (contrary to more abstract models, like

those used by Hinton and Nowlan (Hinton and Nowlan 1987)), and (2) it does not need an explicit fitness (the best genomes are those that are the most often transmitted, may that be because they allow a better survival or more opportunities to mate), which makes it closer to biological evolution. In combination with mEDEA, we will consider two kinds of learning mechanisms, neural networks with neuro-modularity plasticity, and Bayesian optimization.

Synaptic plasticity (chapters 2 and 3) can be easily combined with evolved neural networks and is rooted in neurosciences (Abbott and Nelson 2000; Trappenberg 2010). A strength of this approach is that evolution can evolve the learning system. However, this is also a weakness because it makes the evolutionary challenge more complex, and therefore experiments might need too much time to be done in practice. More importantly, we did not take into account the distal reward problem in our experiments (Soltoggio and Steil 2013), whereas robots that wander in an environment and collect reward, like in mEDEA, will likely need to solve it.

An alternative is to use Bayesian optimization (chapter 4), because this algorithm is one of the fastest learning algorithm for robotics (Lizotte et al. 2007; Calandra et al. 2014), and because it explains well, though at an abstract level, how humans optimize (Borji and Itti 2013). Nevertheless, the best way to combine Bayesian optimization for online learning and evolution is not obvious. Ideally, we would like that evolution gives "instincts" to Bayesian optimization. This idea matches well the concept of priors, which is similar to what we did in our "Intelligent Trial and Error" algorithm: evolution could give prior knowledge in the form a good estimation of the reward associated with each set of parameters of the controllers. Nonetheless, this idea assumes that there is a reward system, whereas the mEDEA algorithm does not include any explicit fitness or objective. A second idea would therefore to let evolution design the reward system, which would constitute the "instincts" (e.g. instinctively, sugar is good). To allow us to select the amount of learning, we will allow learning to act on a subset of the possible parameters of controllers.

Challenge 2: when and why does generalization evolve?

It is well documented that generalization mechanisms like learning can have complex interactions with the evolutionary process (Hinton and Nowlan 1987; Dennett 2003; West-Eberhard 2003; Kirschner and Gerhart 2006). Our main challenge is to understand, thanks to the model sketched in the previous section, how natural selection, acting by definition in the short term, can favor cognitive mechanisms that happen to have the long term advantage of generalization (Johnston 1982). It should be emphasized that generalization can occur without learning, for instance some neural structure might solve a problem in the general case whereas another one only solve it in a few particular cases. As a consequence, learning will only be an option for evolution and we will be attentive to study if other kinds of generalization occur in our experiments.

The intuitive hypotheses is that generalization is encouraged when the environment is very dynamic, but other hypotheses can be investigated. For example, we can expect than generalization is favored when the environment is too complex with regard to the information that can be stored in the genome (Godfrey-Smith 2002): if the environment is too complex, it becomes hopeless to encode in the genome the best behavior for each situation. Another hypothesis is that generalization is required when a neuronal structure cannot be fully specified in the genome because of its size; in this case, the agents will likely need general learning abilities (see chapter 2).

Conclusion

I started my PhD with the desire to make robots more adaptive by taking inspiration from evolution. At the end of my PhD (December 2008), we reached a point at which we were able to evolve neuro-controllers for simple tasks in a reliable way thanks to the optimization of behavioral diversity (Mouret and Doncieux 2009; Mouret 2011a; Mouret and Doncieux 2012a). Since then, our results have been replicated and extended by many teams (Schrum and Miikkulainen 2010; Doncieux and Mouret 2013; Gomes and Christensen 2013; Lehman, Stanley, and Miikkulainen 2013; Li, Storie, et al. 2014). We subsequently introduced an approach to make this technique applicable to real robots, that is, to cross the reality gap (Koos, Mouret, et al. 2013; Mouret, Koos, et al. 2013).

Nonetheless, our techniques were focused on automatically designing controllers for robots. They did not directly make robots more capable of online adaptation: somehow, we drifted.

Most of the work that we conducted after my PhD is a return to this initial motivation: making robots capable of surviving in a dynamic and unpredictable world. Going back and forth between evolutionary biology and robotics, we now have a better idea of:

- how to evolve plastic neural networks with online general learning abilities ⇒ by using generative encoding;
- how can biological networks adapt their behavior to new evolutionary challenges in a few generations ⇒ by being modular, thanks to the selective pressure to minimize the cost of connections;
- what encodings for neural networks combines creativity and viability (minimal loss of fitness) ⇒ thanks to the analysis of the evolvability signatures, we can conclude that SUPG is the best encoding of our set of encodings and our task (locomotion);
- how to harness evolutionary algorithms for fast, creative, and on-line damage recovery in robotics ⇒ by filling many niches in simulation with an evolutionary algorithm, and searching among them with Bayesian optimization.

Each of this small steps makes us closer to having robots that can creatively react to unforeseen situations. In the future, we will do our best to:

- move as far as possible towards realistic setups ⇒ put a damaged robot in a standard apartment, wait for 2 minutes, and the robot should have discovered a way to compensate for the damage;
- make our results are as general as possible ⇒ test everything on several different robots, and if possible for different encodings/algorithms/techniques to draw the most general conclusions that we can;
- continue to investigate both theoretical questions in evolutionary biology and applied questions in robotics;
- continue to develop new analysis tools that go beyond classic performance-based benchmarks.

Bibliography

- L. F. Abbott, S. B. Nelson (Nov. 2000). "Synaptic plasticity: taming the beast". In: *Nature neuroscience* 3, pp. 1178–1183. ISSN: 1097-6256. DOI: 10.1038/81453 (cit. on pp. 5, 9, 11, 59).
- S. R. Ahmadzadeh, P. Kormushev, D. G. Caldwell (2014). "Multi-Objective Reinforcement Learning for AUV Thruster Failure Recovery". In: *IEEE Symposium on Adaptive Dynamic Programming and Reinforcement Learning (ADPRL)* (cit. on p. 57).
- Y.-Y. Ahn, H. Jeong, B. J. Kim (July 2006). "Wiring cost in the organization of a biological neuronal network". In: *Physica A: Statistical Mechanics and its Applications* 367, pp. 531–537. ISSN: 03784371. DOI: 10.1016/j.physa.2005.12.013 (cit. on pp. 18, 52).
- P. Alberch (1991). "From genes to phenotype: dynamical systems and evolvability". English. In: *Genetica* 84.1, pp. 5–11. DOI: 10.1007/BF00123979 (cit. on p. 38).
- R. Albert, H. Jeong, A.-L. Barabási (Sept. 1999). "Internet: Diameter of the World-Wide Web". en. In: *Nature* 401.6749, pp. 130–131. ISSN: 0028-0836. DOI: 10.1038/43601 (cit. on p. 51).
- U. Alon (2006). *An Introduction to Systems Biology: Design Principles of Biological Circuits*. CRC press (cit. on pp. v, 17, 51).
- L. Altenberg (1994). "The evolution of evolvability in genetic programming". In: *Advances in Genetic Programming* 3, pp. 47–74 (cit. on p. 38).
- D. F. Andrews (1972). "Plots of high-dimensional data". In: *Biometrics*, pp. 125–136 (cit. on pp. viii, 52).
- S. Aoi, K. Tsuchiya (2005). "Locomotion control of a biped robot using nonlinear oscillators". In: *Autonomous Robots* 19.3, pp. 219–232 (cit. on p. 41).
- B. Argall, S. Chernova, M. Veloso, B. Browning (2009). "A survey of robot learning from demonstration". In: *Robotics and Autonomous Systems* 57.5, pp. 469–483 (cit. on p. 29).
- S. J. Arnold (1992). "Constraints on phenotypic evolution". In: *American Naturalist*, S85–S107 (cit. on p. 39).
- J. E. Auerbach, J. C. Bongard (2014). "Environmental influence on the evolution of morphological complexity in machines". In: *PLOS Computational Biology* 10.1, e1003399 (cit. on p. 4).
- G. Baldassarre, M. Mirolli (2013). *Intrinsically motivated learning in natural and artificial systems*. Springer (cit. on pp. iv, 2).
- W. Banzhaf, G. Beslon, S. Christensen, J. A. Foster, F. Képès, V. Lefort, J. F. Miller, M. Radman, J. J. Ramsden (2006). "Guidelines: From artificial evolution to computational evolution: a research agenda". In: *Nature Reviews Genetics* 7.9, pp. 729–735 (cit. on pp. iv, 2).
- A. Baranes, P.-Y. Oudeyer (2013). "Active learning of inverse models with intrinsically motivated goal exploration in robots". In: *Robotics and Autonomous Systems* 61.1, pp. 49–73 (cit. on pp. iv, 2, 49, 57).
- T. Barfoot, E. Earon, G. D'Eleuterio (2006). "Experiments in learning distributed control for a hexapod robot". In: *Robotics and Autonomous Systems* 54.10, pp. 864–872 (cit. on pp. 7, 40).
- H. Barlow (1989). "Unsupervised learning". In: *Neural computation* 1.3, pp. 295–311 (cit. on pp. iv, 2).
- J. E. Barrick, D. S. Yu, S. H. Yoon, H. Jeong, T. K. Oh, D. Schneider, R. E. Lenski, J. F. Kim (2009). "Genome evolution and adaptation in a long-term experiment with Escherichia coli". In: *Nature* 461.7268, pp. 1243–1247 (cit. on pp. iv, 2).
- J. G. Bellingham, K. Rajan (Nov. 2007). "Robotics in remote and hostile environments." In: *Science* 318.5853, pp. 1098–102. ISSN: 1095-9203. DOI: 10.1126/science.1146230 (cit. on pp. vi, 27, 28).
- S. Benson-Amram, K. E. Holekamp (2012). "Innovative problem solving by wild spotted hyenas". In: *Proceedings of the Royal Society B: Biological Sciences* 279.1744, pp. 4087–4095 (cit. on p. 31).
- F. Benureau, P.-Y. Oudeyer (2013). "Autonomous reuse of motor exploration trajectories". In: *Development and Learning and Epigenetic Robotics (ICDL), 2013 IEEE Third Joint International Conference on*. IEEE, pp. 1–8 (cit. on p. 57).
- D. Berenson, N. Estevez, H. Lipson (2005). "Hardware evolution of analog circuits for in-situ robotic fault-recovery". In: *Proceedings of NASA/DoD Conference on Evolvable Hardware*, pp. 12–19 (cit. on pp. 6, 8).
- M. Bland (2000). *An introduction to medical statistics*. Ed. 3. Oxford University Press (cit. on p. 50).
- M. Blanke, J. Schröder (2006). *Diagnosis and fault-tolerant control*. Vol. 2. Springer (cit. on pp. 5, 27).
- D. Bonchev, G. A. Buck (2005). "Quantitative measures of network complexity". In: *Complexity in Chemistry, Biology, and Ecology*. Springer, pp. 191–235 (cit. on p. 51).
- J. Bongard (2002). "Evolving modular genetic regulatory networks". In: *Proceedings of IEEE-CEC*. Vol. 2. IEEE, pp. 1872–1877 (cit. on pp. 3, 9).
- J. Bongard, V. Zykov, H. Lipson (2006). "Resilient machines through continuous self-modeling". In: *Science* 314.5802, pp. 1118–1121 (cit. on pp. vii, 6–8, 28, 29, 32, 38).
- J. C. Bongard (2013). "Evolutionary robotics". In: *Communications of the ACM* 56.8, pp. 74–83 (cit. on pp. iii, 1, 6).
- A. J. Booker, J. E. Dennis Jr, P. D. Frank, D. B. Serafini, V. Torczon, M. W. Trosset (1999). "A rigorous framework for optimization of expensive functions by surrogates". In: *Structural optimization* 17.1, pp. 1–13 (cit. on p. 35).
- A. Borji, L. Itti (2013). "Bayesian optimization explains human active search". In: *Advances in Neural Information Processing Systems 26 (NIPS)*, pp. 55–63 (cit. on pp. vii, 30, 31, 35, 59).
- V. Braitenberg (2001). "Brain size and number of neurons: an exercise in synthetic neuroanatomy". In: *Journal of computational neuroscience* 10.1, pp. 71–77 (cit. on p. 16).
- N. Bredeche, J.-M. Montanier, W. Liu, A. F. Winfield (2012). "Environment-driven distributed evolutionary adaptation in a population of autonomous robotic agents". In: *Mathematical and Computer Modelling of Dynamical Systems* 18.1, pp. 101–129 (cit. on pp. ix, 58).
- E. Broadbent, R. Stafford, B. MacDonald (2009). "Acceptance of healthcare robots for the older population: review and future directions". In: *International Journal of Social Robotics* 1.4, pp. 319–330 (cit. on pp. vi, 27, 28).
- E. Brochu, V. M. Cora, N. De Freitas (2010). "A tutorial on Bayesian optimization of expensive cost functions, with application to active user modeling and hierarchical reinforcement learning". In: *arXiv preprint arXiv:1012.2599* (cit. on pp. vii, 35, 36).
- J. Bullinaria (2001). "Simulating the evolution of modular neural systems". In: *Proceedings of the twenty-third annual conference of the Cognitive Science Society*. Lawrence Erlbaum, pp. 146–151 (cit. on p. 18).

- E. Bullmore, O. Sporns (Mar. 2009). "Complex brain networks: graph theoretical analysis of structural and functional systems." In: *Nature reviews. Neuroscience* 10.3, pp. 186–98. ISSN: 1471-0048. DOI: 10.1038/nrn2575 (cit. on pp. 16, 51).
- Caccavale, F. and L. Villani, eds. (2002). *Fault Diagnosis and Fault Tolerance for Mechatronic Systems: Recent Advances*. springer (cit. on p. 6).
- R. Calandra, A. Seyfarth, J. Peters, M. P. Deisenroth (2014). "An experimental comparison of Bayesian optimization for bipedal locomotion". In: *Proceedings of 2014 IEEE International Conference on Robotics and Automation (ICRA)* (cit. on pp. vii, 6, 7, 35, 36, 59).
- J. Carlson, R. R. Murphy (2005). "How UGVs physically fail in the field". In: *IEEE Transactions on Robotics* 21.3, pp. 423–437 (cit. on pp. iii, vi, 27, 28).
- S. Carroll (2001). "Chance and necessity: the evolution of morphological complexity and diversity". In: *Nature* 409.6823, pp. 1102–1109 (cit. on pp. v, 2, 17).
- — (2005). *Endless forms most beautiful: The new science of evo devo and the making of the animal kingdom*. W.W. Norton & Company (cit. on p. 40).
- D. J. Chalmers (1990). "The evolution of learning: An experiment in genetic connectionism". In: *Connectionist Models Summer School* (cit. on pp. 5, 10, 13).
- O. Chapelle, B. Schlkopf, A. Zien (2010). *Semi-Supervised Learning*. 1st. The MIT Press. ISBN: 0262514125, 9780262514125 (cit. on pp. iv, 2).
- B. Chen, D. Hall, D. Chklovskii (2006). "Wiring optimization can relate neuronal structure and function". In: *Proceedings of the National Academy of Sciences* 103.12, p. 4723 (cit. on pp. 16, 18, 19).
- N. Cheney, J. Clune, H. Lipson (2014). "Evolved Electrophysiological Soft Robots". In: *Proc. of Artificial Life*, pp. 222–229. ISBN: 9780262326216 (cit. on p. 4).
- N. Cheney, R. MacCurdy, J. Clune, H. Lipson (2013). "Unshackling evolution: evolving soft robots with multiple materials and a powerful generative encoding". In: *Proc. GECCO*. ACM, New York, NY, pp. 167–174 (cit. on p. 38).
- C. Cherniak, Z. Mokhtarzada, R. Rodriguez-Esteban, K. Changizi (Jan. 2004). "Global optimization of cerebral cortex layout." In: *Proceedings of the National Academy of Sciences* 101.4, pp. 1081–6. ISSN: 0027-8424. DOI: 10.1073/pnas.0305212101 (cit. on pp. 16, 18, 19, 52).
- S. Chernova, M. Veloso (2004). "An evolutionary approach to gait learning for four-legged robots". In: *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Vol. 3. IEEE, pp. 2562–2567 (cit. on p. 7).
- D. Chklovskii, T. Schikorski, C. Stevens (2002). "Wiring optimization in cortical circuits". In: *Neuron* 34.3, pp. 341–347 (cit. on pp. 16, 18).
- D. B. Chklovskii (Oct. 2004). "Exact solution for the optimal neuronal layout problem." In: *Neural computation* 16.10, pp. 2067–2078. ISSN: 0899-7667. DOI: 10 . 1162 / 0899766041732422 (cit. on pp. viii, 19, 52).
- D. J. Christensen, U. P. Schultz, K. Stoy (2013). "A distributed and morphology-independent strategy for adaptive locomotion in self-reconfigurab le modular robots". In: *Robotics and Autonomous Systems* 61.9, pp. 1021–1035 (cit. on pp. 6–8).
- H. Chung, A. Asnodkar, C. Teuscher (2012). "A structural analysis of evolved complex networks-on-chip". In: *Proceedings of the Fifth International Workshop on Network on Chip Architectures*. ACM, pp. 17–22 (cit. on p. 21).
- D. Cliff, P. Husbands, I. Harvey (1993). "Explorations in evolutionary robotics". In: *Adaptive behavior* 2.1, pp. 73–110 (cit. on pp. iii, 1).
- J. Clune, B. E. Beckmann, P. K. McKinley, C. Ofria (2010). "Investigating whether hyperneat produces modular neural networks". In: *Proceedings of the Genetic and Evolutionary Computation Conference*. ACM, pp. 635–642 (cit. on pp. 4, 18, 19).
- J. Clune*, J.-B. Mouret*, H. Lipson (2013). "The evolutionary origins of modularity". In: *Proceedings of the Royal Society B: Biological Sciences* 280, p. 20122863 (cit. on pp. viii, 15, 16, 38, 57).
- J. Clune, B. E. Beckmann, C. Ofria, R. T. Pennock (2009). "Evolving coordinated quadruped gaits with the HyperNEAT generative encoding". In: *Proc. CEC*. IEEE Press, Piscataway, NJ, pp. 2764–2771 (cit. on pp. iii, 1, 38–40, 46, 47).
- J. Clune, H. Lipson (2011). "Evolving 3D Objects with a Generative Encoding Inspired by Developmental Biology". In: *SIGEVOlution* 5.4, pp. 2–12. ISSN: 1931-8499. DOI: 10.1145/2078245.2078246 (cit. on pp. 9, 10, 14).
- J. Clune, C. Ofria, R. T. Pennock (2009). "The Sensitivity of HyperNEAT to Different Geometric Representations of a Problem". In: *Proc. GECCO*. New York, NY, USA: ACM, pp. 675–682. ISBN: 978-1-60558-325-9. DOI: 10.1145/1569901.1569995 (cit. on p. 38).
- J. Clune, K. O. Stanley, R. T. Pennock, C. Ofria (2011). "On the performance of indirect encoding across the continuum of regularity". In: *IEEE Transactions on Evolutionary Computation* 15.3, pp. 346–367 (cit. on pp. viii, 4, 22, 23, 38–40, 42, 47, 51).
- C. C. Coello, G. B. Lamont, D. A. Van Veldhuizen (2007). *Evolutionary algorithms for solving multi-objective problems*. Springer Science & Business Media (cit. on p. 57).
- D. A. Cohn, Z. Ghahramani, M. I. Jordan (1996). "Active learning with statistical models". In: *Journal of artificial intelligence research* (cit. on pp. iv, 2).
- O. Coleman, A. Blair, J. Clune (2014). "Automated Generation of Environments to Test the General Learning Capabilities of AI Agents". In: *Proc. of the International Conference on Genetic and Evolutionary Computation Conference (GECCO'14)*, pp. 161–168. ISBN: 9781450326629 (cit. on p. 5).
- I. Couckuyt, D. Deschrijver, T. Dhaene (2014). "Fast calculation of multiobjective probability of improvement and expected improvement criteria for Pareto optimization". In: *Journal of Global Optimization* 60.3, pp. 575–594 (cit. on p. 57).
- T. M. Cover, J. A. Thomas (1991). *Elements of information theory*. John Wiley & Sons, Inc. (cit. on p. 39).
- A. Crespi, K. Karakasiliotis, A. Guignard, A. Ijspeert (Apr. 2013). "Salamandra Robotica II: An Amphibious Robot to Study Salamander-Like Swimming and Walking Gaits". In: *IEEE Transactions on Robotics* 29.2, pp. 308–320. ISSN: 1552-3098. DOI: 10.1109/TRO.2012.2234311 (cit. on p. 38).
- A. Cully, J.-B. Mouret (2013). "Behavioral Repertoire Learning in Robotics". In: *Proceedings of Genetic and Evolutionary Computation Conference (GECCO)* (cit. on p. 56).
- A. Cully, J.-B. Mouret (2015). "Evolving a Behavioral Repertoire for a Walking Robot". In: *Evolutionary computation* (cit. on pp. 49, 56).
- S. Cussat-Blanc, J. Pollack (2012). "A cell-based developmental model to generate robot morphologies". In: *Proceedings of the fourteenth international conference on Genetic and evolutionary computation conference - GECCO '12*, pp. 537–544. DOI: 10.1145/2330163.2330240 (cit. on p. 3).
- G. Cybenko (1989). "Approximation by superpositions of a sigmoidal function". In: *Mathematics of Control, Signals, and Systems (MCSS)* 2.4, pp. 303–314 (cit. on pp. 10, 16).

- D. B. D'Ambrosio, K. O. Stanley (2008). "Generative encoding for multiagent learning". In: *Proceedings of GECCO*. ACM. New York, New York, USA, p. 819 (cit. on p. 4).
- R. Dawkins (1999). *The extended phenotype: The long reach of the gene*. Oxford University Press (cit. on p. 39).
- K. Deb (2001). *Multi-objective optimization using evolutionary algorithms*. Wiley (cit. on pp. vi, 3, 12, 19, 39, 49, 57).
- K. Deb, A. Pratap, S. Agarwal, T. Meyarivan (2002). "A fast and elitist multiobjective genetic algorithm: NSGA-II". In: *Evolutionary Computation* 6.2, pp. 182–197. ISSN: 1089-778X (cit. on pp. vi, 12, 28, 43).
- S. Degallier, L. Righetti, L. Natale, F. Nori, G. Metta, A. Ijspeert (2008). "A modular bio-inspired architecture for movement generation for the infant-like robot iCub". In: *Proc. of IEEE BioRob*, pp. 795–800 (cit. on p. 55).
- M. Dehmer, A. Mowshowitz (2011). "A history of graph entropy measures". In: *Information Sciences* 181.1, pp. 57–78 (cit. on p. 51).
- D. C. Dennett (2003). "The Baldwin effect: A crane, not a skyhook". In: *Evolution and learning: The Baldwin effect reconsidered*. Ed. by B. H. Weber and D. J. Depew, pp. 60–79 (cit. on pp. 5, 59).
- S. Derégnaucourt, P. P. Mitra, O. Fehér, C. Pytte, O. Tchernichovski (2005). "How sleep affects the developmental learning of bird song". In: *Nature* 433.7027, pp. 710–716 (cit. on p. 32).
- A. Di Ferdinando, R. Calabretta, D. Parisi (2001). "Evolving modular architectures for neural networks". In: *Connectionist models of learning, development and evolution: proceedings of the Sixth Neural Computation and Psychology Workshop*. Springer Verlag, pp. 253–262 (cit. on p. 18).
- K. P. Dial (2003). "Wing-assisted incline running and the evolution of flight". In: *Science* 299.5605, pp. 402–404 (cit. on pp. iv, 2).
- S. Doncieux, N. Bredeche, J.-B. Mouret, A. Eiben (2015). "Evolutionary Robotics: What, Why, and Where to". In: *Frontiers in Robotics and AI* 2.4 (cit. on pp. iii, 1, 6, 50, 53).
- S. Doncieux (2013). "Transfer learning for direct policy search: A reward shaping approach". In: *Development and Learning and Epigenetic Robotics (ICDL), 2013 IEEE Third Joint International Conference on*. IEEE, pp. 1–6 (cit. on p. 57).
- S. Doncieux, J.-A. Meyer (2004). "Evolving modular neural networks to solve challenging control problems". In: *Proceedings of the Fourth International ICSC Symposium on engineering of intelligent systems (EIS 2004)* (cit. on p. 3).
- S. Doncieux, J.-B. Mouret (2013). "Behavioral diversity with multiple behavioral distances". In: *Proc. CEC*. IEEE Press, pp. 1427–1434 (cit. on pp. 39, 61).
- S. Doncieux, J.-B. Mouret (2009). "Single Step Evolution of Robot Controllers for Sequential Tasks". In: *Proc. GECCO*. Montreal, Quebec, Canada: ACM, New York, NY, pp. 1771–1772. ISBN: 978-1-60558-325-9. DOI: 10 . 1145 / 1569901 . 1570152 (cit. on p. 48).
- — (July 2010). "Behavioral diversity measures for Evolutionary Robotics". In: *Proc. CEC*. IEEE Press, Piscataway, NJ, pp. 1–8 (cit. on p. 48).
- — (2014). "Beyond black-box optimization: a review of selective pressures for evolutionary robotics". In: *Evolutionary Intelligence* 7.2, pp. 71–93. ISSN: 1864-5909 (cit. on pp. 3, 22, 39, 49, 50).
- G. Dyke, R. Kat, C. Palmer, J. Kindere, D. Naish, B. Ganapathisubramani (2013). "Aerodynamic performance of the feathered dinosaur Microraptor and the evolution of feathered flight". In: *Nature Communications* 4 (cit. on pp. iv, 2).
- A. L. Edwards (1950). "Experimental design in psychological research." In: (cit. on p. 50).
- A. E. Eiben, J. E. Smith (2003). *Introduction to Evolutionary Computing*. Springer Berlin Heidelberg (cit. on p. 35).
- J. Ellson, E. Gansner, L. Koutsofios, S. C. North, G. Woodhull (2002). "Graphviz—open source graph drawing tools". In: *Graph Drawing*. Springer, pp. 483–484 (cit. on p. 52).
- F. Endres, J. Hess, N. Engelhard, J. Sturm, D. Cremers, W. Burgard (2012). "An Evaluation of the RGB-D SLAM System". In: *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)* (cit. on p. 28).
- M. S. Erden, K. Leblebicioğlu (2008). "Free gait generation with reinforcement learning for a six-legged robot". In: *Robotics and Autonomous Systems* 56.3, pp. 199–212 (cit. on pp. 7, 8).
- C. Espinosa-Soto, A. Wagner (2010). "Specialization can drive the evolution of modularity". In: *PLoS Computational Biology* 6.3, e1000719 (cit. on pp. v, vi, 4, 17–19, 22).
- W. G. Fenton, T. M. McGinnity, L. P. Maguire (2001). "Fault diagnosis of electronic systems using intelligent techniques: a review". In: *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews* 31.3, pp. 269–281 (cit. on p. 28).
- A. V. Fiacco, G. P. McCormick (1990). *Nonlinear programming: sequential unconstrained minimization techniques*. Vol. 4. Siam (cit. on p. 36).
- D. Floreano, P. Dürr, C. Mattiussi (Jan. 2008). "Neuroevolution: from architectures to learning". In: *Evolutionary Intelligence* 1.1, pp. 47–62. ISSN: 1864-5909. DOI: 10.1007/s12065-007-0002-4 (cit. on pp. 10–12).
- A. I. J. Forrester, A. J. Keane (2009). "Recent advances in surrogate-based optimization". In: *Progress in Aerospace Sciences* 45.1, pp. 50–79 (cit. on p. 35).
- S. Fortunato (Feb. 2010). "Community detection in graphs". en. In: *Physics Reports* 486.3-5, pp. 75–174. ISSN: 03701573. DOI: 10.1016/j.physrep.2009.11.002 (cit. on p. 51).
- M. J. Frank, E. D. Claus (2006). "Anatomy of a decision: striato-orbitofrontal interactions in reinforcement learning, decision making, and reversal." In: *Psychological review* 113.2, p. 300 (cit. on p. 11).
- P. M. Frank (1990). "Fault diagnosis in dynamic systems using analytical and knowledge-based redundancy: A survey and some new results". In: *Automatica* 26.3, pp. 459–474 (cit. on p. 5).
- P. M. Frank, B. Köppen-Seliger (1997). "New developments using AI in fault diagnosis". In: *Engineering Applications of Artificial Intelligence* 10.1, pp. 3–14 (cit. on p. 5).
- T. M. Fruchterman, E. M. Reingold (1991). "Graph drawing by force-directed placement". In: *Software: Practice and experience* 21.11, pp. 1129–1164 (cit. on p. 52).
- A. Fuchs, B. Goldner, I. Nolte, N. Schilling (Sept. 2014). "Ground reaction force adaptations to tripedal locomotion in dogs". In: *Veterinary journal* 201.3, pp. 307–15. ISSN: 1532-2971. DOI: 10.1016/j.tvjl.2014.05.012 (cit. on p. 28).
- J. Gauci, K. O. Stanley (2010). "Autonomous evolution of topographic regularities in artificial neural networks". In: *Neural Comput.* 22.7, pp. 1860–1898 (cit. on pp. 4, 23, 48).
- J. Gauci, K. O. Stanley (2011). "Indirect encoding of neural networks for scalable go". In: *Parallel Problem Solving from Nature–PPSN XI*. Springer, pp. 354–363 (cit. on p. 4).

- T. Geng, B. Porr, F. Wörgötter (2006). "Fast biped walking with a sensor-driven neuronal controller and real-time online learning". In: *The International Journal of Robotics Research* 25.3, pp. 243–259 (cit. on p. 7).
- J. Gerhart, M. Kirschner (2007). "The theory of facilitated variation". In: *Proc. Natl. Acad. Sci. USA.* 104.Suppl 1, pp. 8582–8589 (cit. on pp. iv, 2, 3, 39).
- B. Girard, N. Tabareau, Q. Pham, A. Berthoz, J.-J. Slotine (2008). "Where neuroscience and dynamic system theory meet autonomous robotics: a contracting basal ganglia model for action selection". In: *Neural Networks* 21.4, pp. 628–641 (cit. on pp. 11, 12).
- P. Godfrey-Smith (2002). "Environmental complexity and the evolution of cognition". In: *The evolution of intelligence*, pp. 233–249 (cit. on p. 59).
- A. D. Goldberg, C. D. Allis, E. Bernstein (2007). "Epigenetics: a landscape takes shape". In: *Cell* 128.4, pp. 635–638 (cit. on p. 40).
- K. Goldberg, B. Chen (2001). "Collaborative control of robot motion: robustness to error". In: *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Vol. 2, pp. 655–660. ISBN: 0-7803-6612-3. DOI: 10.1109/IROS.2001.976244 (cit. on p. 6).
- J. Gomes, A. L. Christensen (2013). "Generic behaviour similarity measures for evolutionary swarm robotics". In: *Proceedings of the 15th annual conference on Genetic and evolutionary computation*. ACM, pp. 199–206 (cit. on p. 61).
- S. Gould, R. Lewontin (1979). "The spandrels of San Marco and the Panglossian paradigm: a critique of the adaptationist programme". In: *Proceedings of the Royal Society of London. Series B, Biological Sciences* 205.1161, pp. 581–598 (cit. on pp. 15, 25).
- S. Gould, E. Vrba (1982). "Exaptation; a missing term in the science of form". In: *Paleobiology* 8.1, pp. 4–15 (cit. on pp. 21, 58).
- J. J. Grefenstette, A. C. Schultz, D. E. Moriarty (1999). "Evolutionary algorithms for reinforcement learning". In: *Journal of Artificial Intelligence Research*, pp. 241–276 (cit. on p. 6).
- T. L. Griffiths, C. Lucas, J. Williams, M. L. Kalish (2009). "Modeling human function learning with Gaussian processes". In: *Advances in Neural Information Processing Systems 21 (NIPS)*, pp. 553–560 (cit. on p. 35).
- S. Grossberg (June 2000). "The complementary brain: unifying brain dynamics and modularity." In: *Trends in cognitive sciences* 4.6, pp. 233–246. ISSN: 1879-307X (cit. on p. 16).
- F. Gruau (1994). "Automatic definition of modular neural networks". In: *Adapt. Behav.* 3.2, pp. 151–183 (cit. on pp. 38, 40).
- F. Gruau, D. Whitley (1993). "Adding learning to the cellular development of neural networks: Evolution and the Baldwin effect". In: *Evolutionary computation* 1.3, pp. 213–233. ISSN: 1063-6560 (cit. on pp. 5, 9, 10).
- R. Guimera, L. Amaral (2005). "Functional cartography of complex metabolic networks". In: *Nature* 433.7028, pp. 895–900 (cit. on pp. v, 17, 51).
- K. Gurney, T. J. Prescott, P. Redgrave (2001). "A computational model of action selection in the basal ganglia. II. Analysis and simulation of behaviour". In: *Biological cybernetics* 84.6, pp. 411–423 (cit. on p. 12).
- M. Hausknecht, P. Khandelwal, R. Miikkulainen, P. Stone (2012). "HyperNEAT-GGP : A HyperNEAT-based Atari General Game Player". In: *Proc. of GECCO' 12*, pp. 217–224. ISBN: 9781450311779 (cit. on p. 49).
- M. Hayashida, T. Akutsu (2010). "Comparing biological networks via graph compression". In: *BMC systems biology* 4.Suppl 2, S13 (cit. on pp. 12, 51).
- S. Haykin (1998). *Neural Networks: A Comprehensive Foundation*. 2nd. Prentice Hall. ISBN: 0132733501 (cit. on pp. iv, viii, 2, 16, 52).
- D. O. Hebb (1949). *The organization of behavior*. Wiley (cit. on p. 9).
- V. Heidrich-Meisner, C. Igel (Oct. 2009). "Neuroevolution strategies for episodic reinforcement learning". In: *Journal of Algorithms* 64.4, pp. 152–168 (cit. on p. 6).
- T. Hemker, M. Stelzer, O. Von Stryk, H. Sakamoto (2009). "Efficient walking speed optimization of a humanoid robot". In: *The International Journal of Robotics Research* 28.2, pp. 303–314 (cit. on p. 7).
- G. Hinton, S. Nowlan (1987). "How learning can guide evolution". In: *Complex systems* 1, pp. 495–502 (cit. on pp. 5, 59).
- A. Hintze, C. Adami (Feb. 2008). "Evolution of complex modular biological networks." In: *PLoS computational biology* 4.2, e23. ISSN: 1553-7358. DOI: 10.1371/journal.pcbi.0040023 (cit. on pp. v, 17).
- J. H. Holland (1975). *Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence.* U Michigan Press (cit. on pp. iii, 1).
- H. H. Hoos, T. Stützle (2005). *Stochastic local search: Foundations and applications*. Morgan Kaufmann (cit. on pp. vii, 28).
- J. J. Hopfield (1982). "Neural networks and physical systems with emergent collective computational abilities". In: *Proceedings of the national academy of sciences* 79.8, pp. 2554–2558 (cit. on p. 16).
- G. S. Hornby, H. Lipson, J. B. Pollack (2003). "Generative representations for the automated design of modular physical robots". In: *IEEE Trans. Robot. Automat.* 19.4, pp. 703–719 (cit. on pp. 38–40).
- G. S. Hornby, J. B. Pollack (2002). "Creating high-level components with a generative representation for body-brain evolution". In: *Artificial Life* 8.3, pp. 223–246. ISSN: 1064-5462 (cit. on pp. 4, 9, 39).
- G. Hornby (2005). "Measuring, enabling and comparing modularity, regularity and hierarchy in evolutionary design". In: *Proceedings of the 2005 conference on Genetic and evolutionary computation*. ACM, p. 1736 (cit. on pp. 10, 51).
- G. Hornby, S. Takamura, T. Yamamoto, M. Fujita (2005). "Autonomous evolution of dynamic gaits with two quadruped robots". In: *IEEE Transactions on Robotics* 21.3, pp. 402–410 (cit. on pp. 7, 28, 37, 38, 40, 57).
- J. C. Houk, J. L. Adams, A. G. Barto (1995). "A model of how the basal ganglia generate and use neural signals that predict reinforcement". In: *Models of information processing in the basal ganglia*, pp. 249–270 (cit. on p. 11).
- T. Hu, W. Banzhaf (Jan. 2010). "Evolvability and Speed of Evolutionary Algorithms in Light of Recent Developments in Biology". In: *J. Artificial Evol. App.* 2010, 1:1–1:28. ISSN: 1687-6229. DOI: 10.1155/2010/568375 (cit. on pp. iv, 2, 3, 38, 39).
- J. Huizinga, J.-B. Mouret, J. Clune (2014). "Evolving Neural Networks That Are Both Modular and Regular : HyperNeat Plus the Connection Cost Technique". In: *Proc. of the International Conference on Genetic and Evolutionary Computation (GECCO'14)* (cit. on p. 51).
- I. Hupkens, A. Deutz, K. Yang, M. Emmerich (2015). "Faster Exact Algorithms for Computing Expected Hypervolume Improvement". In: *Proc. of Evolutionary Multi-Criterion Optimization*. Springer (cit. on p. 57).

- A. J. Ijspeert, A. Crespi, D. Ryczko, J. M. Cabelguen (2007). "From swimming to walking with a salamander robot driven by a spinal cord model". In: *Science* 315.5817, pp. 1416–1420 (cit. on pp. 13, 41, 55).
- A. J. Ijspeert (2008). "Central pattern generators for locomotion control in animals and robots: a review". In: *NeuralNetworks* 21.4, pp. 642–653 (cit. on pp. iv, vii, 2).
- A. J. Ijspeert, J. Hallam, D. Willshaw (1999). "Evolving swimming controllers for a simulated lamprey with inspiration from neurobiology". In: *Adaptive Behavior* 7.2, pp. 151–172 (cit. on pp. iii, 1).
- R. Isermann (1998). "On fuzzy logic applications for automatic control, supervision, and fault diagnosis". In: *Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on* 28.2, pp. 221–235 (cit. on p. 5).
- M. Ito (2008). "Control of mental activities by internal models in the cerebellum". In: *Nature Reviews Neuroscience* 9.4, pp. 304–313 (cit. on p. 32).
- R. Jacobs, M. Jordan (1992). "Computational consequences of a bias toward short connections". In: *Journal of Cognitive Neuroscience* 4.4, pp. 323–336 (cit. on p. 18).
- B. Jakimovski, E. Maehle (2010). "In situ self-reconfiguration of hexapod robot OSCAR using biologically inspired approaches". In: *Climbing and Walking Robots. InTech* (cit. on p. 5).
- N. Jakobi (1997). "Evolutionary robotics and the radical envelope-of-noise hypothesis". In: *Adaptive behavior* (cit. on p. 6).
- N. Jakobi, P. Husbands, I. Harvey (1995). "Noise and the reality gap: The use of simulation in evolutionary robotics". In: *Proceedings of the European Conference on Artificial Life (ECAL)*, pp. 704–720 (cit. on p. 28).
- S. L. Jarvis, D. R. Worley, S. M. Hogy, A. E. Hill, K. K. Haussler, R. F. Reiser II (2013). "Kinematic and kinetic analysis of dogs during trotting after amputation of a thoracic limb". In: *American journal of veterinary research* 74.9, pp. 1155–1163 (cit. on p. 28).
- J.-M. Jehanno, A. Cully, C. Grand, J.-B. Mouret (n.d.). "Design of a Wheel-Legged Hexapod Robot for Creative Adaptation". In: *Proceedings of CLAWAR*, to appear (cit. on p. 56).
- Y. Jin (2011). "Surrogate-assisted evolutionary computation: Recent advances and future challenges". In: *Swarm and Evolutionary Computation* 1.2, pp. 61–70 (cit. on p. 35).
- Y. Jin, J. Branke (June 2005). "Evolutionary optimization in uncertain environments-a survey". In: *IEEE Transactions on Evolutionary Computation* 9.3, pp. 303–317 (cit. on p. 37).
- T. D. Johnston (1982). "Selective Costs and Benefits in the Evolution of Learning". In: *Advances in the Study of Behavior*. Ed. by R. A. H. Jay S. Rosenblatt Colin Beer and Marie-Claire Busnel. Vol. 12. Academic Press, pp. 65–106 (cit. on p. 59).
- D. R. Jones, M. Schonlau, W. J. Welch (1998). "Efficient global optimization of expensive black-box functions". In: *Journal of Global optimization* 13.4, pp. 455–492 (cit. on p. 35).
- T. Junttila, P. Kaski (2007). "Engineering an efficient canonical labeling tool for large and sparse graphs". In: *Proceedings of the Ninth Workshop on Algorithm Engineering and Experiments and the Fourth Workshop on Analytic Algorithms and Combinatorics*. SIAM, pp. 135–149 (cit. on pp. viii, 12, 52).
- S. Kajita, B. Espiau (2008). "Handbook of Robotics". In: Springer. Chap. Legged Robots, pp. 361–389 (cit. on pp. iii, 1).
- G. Karafotias, M. Hoogendoorn, A. Eiben (2014). "Parameter Control in Evolutionary Algorithms: Trends and Challenges". In: *IEEE Transactions on Evolutionary Computation*, pp. 1–1. ISSN: 1089-778X, 1089-778X, 1941-0026. DOI: 10.1109/TEVC.2014.2308294 (cit. on pp. viii, 50).
- G. Karlebach, R. Shamir (Oct. 2008). "Modelling and analysis of gene regulatory networks". In: *Nature Reviews Molecular Cell Biology* 9.10, pp. 770–780. ISSN: 1471-0072, 1471-0080. DOI: 10.1038/nrm2503 (cit. on p. 51).
- N. Kashtan, U. Alon (Sept. 2005). "Spontaneous evolution of modularity and network motifs". In: *Proceedings of the National Academy of Sciences* 102.39, pp. 13773–13778. ISSN: 0027-8424. DOI: 10.1073/pnas.0503610102 (cit. on pp. vi, 4, 18, 19, 22, 49).
- N. Kashtan, E. Noor, U. Alon (Aug. 2007). "Varying environments can speed up evolution." In: *Proceedings of the National Academy of Sciences* 104.34, pp. 13711–13716. ISSN: 0027-8424. DOI: 10.1073/pnas.0611630104 (cit. on pp. vi, 18, 19, 49).
- H. Katebi, K. A. Sakallah, I. L. Markov (2012). "Graph Symmetry Detection and Canonical Labeling: Differences and Synergies". In: *Proceedings of Turing-100* (cit. on pp. viii, 12, 52).
- S. A. Kauffman (1993). *The origins of order: Self organization and selection in evolution*. Oxford University Press, USA (cit. on p. 52).
- H. Kimura, T. Yamashita, S. Kobayashi (2001). "Reinforcement learning of walking behavior for a four-legged robot". In: *Proceedings of IEEE Conference on Decision and Control (CDC)*. Vol. 1. IEEE, pp. 411–416 (cit. on pp. 6, 7).
- M. W. Kirschner, J. C. Gerhart (2006). *The plausibility of life: Resolving Darwin's dilemma*. Yale University Press (cit. on pp. iv, 2, 3, 58, 59).
- M. Kirschner, J. Gerhart (1998). "Evolvability". In: *Proc. Natl. Acad. Sci. USA*. 95.15, pp. 8420–8427 (cit. on pp. 2, 3, 38).
- J. Kluger, J. Lovell (2006). *Apollo 13*. Mariner Books. ISBN: 978-0618619580 (cit. on pp. iii, 1, 28).
- J. Knowles (2006). "ParEGO: a hybrid algorithm with on-line landscape approximation for expensive multiobjective optimization problems". In: *Evolutionary Computation, IEEE Transactions on* 10.1, pp. 50–66 (cit. on p. 57).
- J. Kober, J. Peters (2013). "Reinforcement Learning in Robotics: A Survey". In: *Reinforcement Learning: State of the Art*. Springer, pp. 579–610 (cit. on pp. 2, 6, 28, 29, 55, 57).
- J. Kodjabachian, J.-A. Meyer (1998). "Evolution and development of neural controllers for locomotion, gradient-following, and obstacle-avoidance in artificial insects". In: *IEEE Transactions on Neural Networks* 9.5, pp. 796–812 (cit. on pp. iii, 1, 3, 38, 56).
- N. Kohl, P. Stone (2004). "Policy gradient reinforcement learning for fast quadrupedal locomotion". In: *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*. Vol. 3. IEEE, pp. 2619–2624 (cit. on pp. vii, 6, 7, 28, 57).
- T. Kohonen (2001). *Self-organizing maps*. Vol. 30. Springer Science & Business Media (cit. on pp. viii, 52).
- M. Komosiński, A. Rotaru-Varga (2001). "Comparison of different genotype encodings for simulated three-dimensional agents". In: *Artificial Life* 7.4, pp. 395–418 (cit. on pp. 38, 40).
- S. Koos, A. Cully, J.-B. Mouret (2013). "Fast damage recovery in robotics with the T-resilience algorithm". In: *International Journal of Robotics Research* 32.14, pp. 1700–1723 (cit. on pp. 6–8, 38, 40, 57).
- S. Koos, J.-B. Mouret, S. Doncieux (2013). "The transferability approach: Crossing the reality gap in evolutionary robotics". In: *Evolutionary Computation, IEEE Transactions on* 17.1, pp. 122–145 (cit. on pp. 6, 27, 28, 57, 61).

- K. P. Körding, D. M. Wolpert (2004). "Bayesian integration in sensorimotor learning". In: *Nature* 427.6971, pp. 244–247 (cit. on p. 31).
- I. Koren, C. M. Krishna (2007). *Fault-tolerant systems*. Morgan Kaufmann (cit. on pp. iii, 1, 5).
- M. Kovač (2013). "The Bioinspiration Design Paradigm: A Perspective for Soft Robotics". In: *Soft Robotics* 1.1, pp. 28–37. ISSN: 2169-5172. DOI: 10.1089/soro.2013.0004 (cit. on pp. viii, 49).
- A. Kraskov, H. Stögbauer, P. Grassberger (2004). "Estimating mutual information". In: *Physical Review E* 69.6, p. 066138 (cit. on p. 39).
- M. LaBarbera (1990). "Principles of design of fluid transport systems in zoology". In: *Science* 249.4972, p. 992 (cit. on p. 18).
- S. B. Laughlin, T. J. Sejnowski (Sept. 2003). "Communication in neuronal networks." In: *Science (New York, N.Y.)* 301.5641, pp. 1870–4. ISSN: 1095-9203. DOI: 10.1126/science.1089662 (cit. on p. 18).
- Y. LeCun, L. Bottou, G. B. Orr, K.-R. Müller (1998). "Effiicient BackProp". In: *Neural Networks: Tricks of the Trade*. Springer-Verlag, pp. 9–50 (cit. on p. 16).
- S. Lee, J. Yosinski, K. Glette, H. Lipson, J. Clune (2013). "Evolving Gaits for Physical Robots with the HyperNEAT Generative Encoding: The Benefits of Simulation". In: *Applications of Evolutionary Computation*. Ed. by A. Esparcia-Alcázar. Vol. 7835. LNCS. Springer Berlin Heidelberg, pp. 540–549. ISBN: 978-3-642-37191-2. DOI: 10.1007/978-3-642-37192-9_54 (cit. on pp. 37, 38, 42).
- J. Lehman, K. Stanley (2011a). "Abandoning objectives: Evolution through the search for novelty alone". In: *Evolutionary Computation* 19.2, pp. 189–223 (cit. on pp. 5, 12, 39, 49, 55).
- J. Lehman, K. O. Stanley (2011b). "Evolving a diversity of virtual creatures through novelty search and local competition". In: *Proceedings of the 13th annual conference on Genetic and evolutionary computation*. ACM, pp. 211–218 (cit. on p. 57).
- — (2011c). "Improving evolvability through novelty search and self-adaptation". In: *Proc. CEC*. IEEE Press, Piscataway, NJ, pp. 2693–2700 (cit. on pp. 39, 47, 48).
- — (2013). "Evolvability Is Inevitable: Increasing Evolvability without the Pressure to Adapt". In: *PloS One* 8.4, e62186 (cit. on pp. 4, 39, 47).
- J. Lehman, K. O. Stanley, R. Miikkulainen (2013). "Effective diversity maintenance in deceptive domains". In: *Proceedings of the 15th annual conference on Genetic and evolutionary computation*. ACM, pp. 215–222 (cit. on p. 61).
- E. A. Leicht, M. E. J. Newman (2008). "Community structure in directed networks". In: *Physical review letters*, pp. 118703–118707 (cit. on pp. vi, 19, 51, 52).
- R. E. Lenski, C. Ofria, T. C. Collier, C. Adami (1999). "Genome complexity, robustness and genetic interactions in digital organisms". In: *Nature* 400.6745, pp. 661–664 (cit. on p. 19).
- R. E. Lenski, C. Ofria, R. T. Pennock, C. Adami (2003). "The evolutionary origin of complex features". In: *Nature* 423.6936, pp. 139–144 (cit. on pp. 19, 52).
- M. A. Lewis, A. H. Fagg, A. Solidum (1992). "Genetic programming approach to the construction of a neural network for control of a walking robot". In: *Proc. IEEE ICRA*. IEEE Press, Piscataway, NJ, pp. 2618–2623 (cit. on pp. iii, 1, 38).
- J. Li, J. Storie, J. Clune (2014). "Encouraging creative thinking in robots improves their ability to solve challenging problems". In: *Proceedings of the 2014 conference on Genetic and evolutionary computation*. ACM, pp. 193–200 (cit. on p. 61).
- M. Li, P. Vitányi (2008). *An introduction to Kolmogorov complexity and its applications*. Springer (cit. on p. 12).
- C.-M. Lin, C.-H. Chen (2007). "Robust fault-tolerant control for a biped robot using a recurrent cerebellar model articulation controller". In: *Systems, Man, and Cybernetics, Part B: Cybernetics* 37.1, pp. 110–123 (cit. on p. 6).
- H. Lipson (2007). "Principles of modularity, regularity, and hierarchy for scalable systems". In: *Journal of Biological Physics and Chemistry* 7.4, pp. 125–128 (cit. on pp. iv, v, 2, 4, 12, 17, 19, 21).
- H. Liu, H. Iba (June 2004). "A hierarchical approach for adaptive humanoid robot control". In: *Proc. CEC*. Vol. 2. IEEE Press, Piscataway, NJ, 1546–1553 Vol.2 (cit. on p. 40).
- D. J. Lizotte, T. Wang, M. H. Bowling, D. Schuurmans (2007). "Automatic Gait Optimization with Gaussian Process Regression." In: *Proceedings of the the International Joint Conference on Artificial Intelligence (IJCAI)*. Vol. 7, pp. 944–949 (cit. on pp. vii, 6, 7, 35, 57, 59).
- S. Mahdavi, P. Bentley (2003). "An evolutionary approach to damage recovery of robot motion with muscles". In: *Advances in Artificial Life*, pp. 248–255 (cit. on p. 6).
- S. Mahdavi, P. Bentley (2006). "Innately adaptive robotics through embodied evolution". In: *Autonomous Robots* 20.2, pp. 149–163 (cit. on p. 8).
- P. Marrow, M. Heath, I. I. Re (1999). "Evolvability: Evolution, Computation, Biology". In: *Proc. GECCO (GECCO-99 Workshop on Evolvability)*. ACM, New York, NY, pp. 30–33 (cit. on p. 38).
- B. Matérn (1960). "Spatial variation. Stochastic models and their application to some problems in forest surveys and other sampling investigations." In: *Meddelanden fran statens Skogsforskningsinstitut* 49.5 (cit. on p. 36).
- C. Mattiussi, D. Floreano (Oct. 2007). "Analog Genetic Encoding for the Evolution of Circuits and Networks". In: *Evolutionary Computation* 11.5, pp. 596–607. ISSN: 1089-778X. DOI: 10.1109/TEVC.2006.886801 (cit. on p. 9).
- B. D. McKay (1981). "Practical graph isomorphism". In: *Congressus Numerantium* 30, pp. 45–87 (cit. on pp. viii, 12, 52).
- D. Meunier, R. Lambiotte, E. T. Bullmore (Jan. 2010). "Modular and Hierarchically Modular Organization of Brain Networks." In: *Frontiers in neuroscience* 4.December, p. 200. ISSN: 1662-453X. DOI: 10.3389/fnins.2010.00200 (cit. on p. 16).
- J.-A. Meyer, A. Guillot (2008). "Biologically Inspired Robots". In: *Handbook of Robotics*. Vol. G. Springer, pp. 1315–1422 (cit. on pp. viii, 49).
- J.-A. Meyer, P. Husbands, I. Harvey (1998). "Evolutionary robotics: A survey of applications and problems". In: *Evolutionary Robotics*. Springer, pp. 1–21 (cit. on pp. iii, 1).
- R. Milo, S. Shen-Orr, S. Itzkovitz, N. Kashtan, D. Chklovskii, U. Alon (Oct. 2002). "Network Motifs: Simple Building Blocks of Complex Networks". en. In: *Science* 298.5594, pp. 824–827. ISSN: 0036-8075, 1095-9203. DOI: 10.1126/science.298.5594.824 (cit. on p. 51).
- M. L. Minsky, S. A. Papert (1987). *Perceptrons - Expanded Edition: An Introduction to Computational Geometry*. MIT press (cit. on p. 10).
- M. Mitchell, S. Forrest, J. H. Holland (1992). "The royal road for genetic algorithms: Fitness landscapes and GA performance". In: *Proceedings of the first european conference on artificial life*. Cambridge: The MIT Press, pp. 245–254 (cit. on p. 52).

- V. Mnih, K. Kavukcuoglu, D. Silver, A. a. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, D. Hassabis (2015). "Human-level control through deep reinforcement learning". In: *Nature* 518.7540, pp. 529–533. ISSN: 0028-0836. DOI: 10.1038/nature14236 (cit. on pp. vii, 49).
- J. Mockus (2013). *Bayesian approach to global optimization: theory and applications*. Kluwer Academic (cit. on pp. vii, 6, 30, 35).
- A. P. Moczek (2008). "On the origins of novelty in development and evolution". In: *BioEssays* 30.5, pp. 432–447 (cit. on pp. iv, 2).
- E. Mones, L. Vicsek, T. Vicsek (Mar. 2012). "Hierarchy Measure for Complex Networks". In: *PLoS ONE* 7.3, e33799. DOI: 10.1371/journal.pone.0033799 (cit. on p. 51).
- S. C. Morris (1993). "The fossil record and the early evolution of the Metazoa". In: *Nature* 361, pp. 219–225 (cit. on pp. iv, 2).
- R. Morrison, K. De Jong (1999). "A test problem generator for non-stationary environments". In: *Proc. CEC*. Vol. 3, pp. 2047–2053 (cit. on p. 37).
- G. Morse, S. Risi, C. R. Snyder, K. O. Stanley (2013). "Single-unit pattern generators for quadruped locomotion". In: *Proc. GECCO*. ACM, New York, NY, pp. 719–726 (cit. on pp. vii, 38, 41–43, 46, 56).
- K. Mostafa, C. Tsai, I. Her (2010). "Alternative Gaits for Multiped Robots with Leg Failures To Retain Maneuverability". In: *International Journal of Advanced Robotic Systems* 7.4, p. 31 (cit. on p. 5).
- C. Moulin-Frier, P.-Y. Oudeyer (2013). "Exploration strategies in developmental robotics: a unified probabilistic framework". In: *Development and Learning and Epigenetic Robotics (ICDL), 2013 IEEE Third Joint International Conference on*. IEEE, pp. 1–6 (cit. on p. 57).
- V. Mountcastle (1997). "The columnar organization of the neocortex." In: *Brain* 120.4, pp. 701–722 (cit. on pp. v, 17).
- J.-B. Mouret (2011a). "Novelty-based Multiobjectivization". In: *New Horizons in Evolutionary Robotics: Extended Contributions from the 2009 EvoDeRob Workshop*. Springer, pp. 139–154 (cit. on p. 61).
- J.-B. Mouret, J. Clune (2012). "An Algorithm to Create Phenotype-Fitness Maps". In: *Proceedings of the Thirteenth International Conference on the Simulation and Synthesis of Living Systems (ALIFE 13)*, 593–594 (extended abstract) (cit. on p. 49).
- J.-B. Mouret, S. Doncieux (2009). "Evolving modular neural-networks through exaptation". In: *IEEE Congress on Evolutionary Computation, CEC'09*, pp. 1570–1577 (cit. on pp. ix, 49, 61).
- J.-B. Mouret, S. Doncieux (2010). "Sferes_v2: Evolvin' in the Multi-Core World". In: *Proceedings of IEEE Congress on Evolutionary Computation (CEC)*, pp. 4079–4086 (cit. on pp. viii, 3, 28, 49).
- J.-B. Mouret, S. Doncieux (2012b). "Encouraging Behavioral Diversity in Evolutionary Robotics: an Empirical Study". In: *Evolutionary Computation* 1.20 (cit. on pp. 6, 12, 15).
- — (Jan. 2012a). "Encouraging Behavioral Diversity in Evolutionary Robotics: an Empirical Study". In: *Evolutionary Computation* 20.1, pp. 91–133 (cit. on pp. v, viii, ix, 3, 4, 6, 22, 39, 43, 48–50, 52, 53, 57, 61).
- J.-B. Mouret (2011b). "Novelty-based multiobjectivization". In: *New Horizons in Evolutionary Robotics*. Springer, pp. 139–154 (cit. on pp. 3, 12, 39, 49, 57).
- J.-B. Mouret, S. Doncieux (2008). "MENNAG: a modular, regular and hierarchical encoding for neural-networks based on attribute grammars". In: *Evolutionary Intelligence* 1.3, pp. 187–207 (cit. on pp. ix, 3).
- J.-B. Mouret, S. Doncieux, B. Girard (2010). "Importing the computational neuroscience toolbox into neuro-evolution-application to basal ganglia". In: *Proceedings of GECCO*. ACM, pp. 587–594 (cit. on pp. v, 9, 12).
- J.-B. Mouret, S. Doncieux, J.-A. Meyer (2006). "Incremental Evolution of Target-Following Neuro-controllers for Flapping-Wing Animats". In: *From Animals to Animats 9*, pp. 606–618 (cit. on p. 56).
- J.-B. Mouret, S. Koos, S. Doncieux (2013). "Crossing the reality gap: a short introduction to the transferability approach". In: *arXiv preprint arXiv:1307.1870* (cit. on p. 61).
- J.-B. Mouret, P. Tonelli (2014). "Artificial evolution of plastic neural networks: a few key concepts". In: *Growing Adaptive Machines*. Springer, pp. 251–261 (cit. on p. 5).
- A. Mowshowitz (1968a). "Entropy and the complexity of graphs: I. An index of the relative complexity of a graph". In: *The bulletin of mathematical biophysics* 30.1, pp. 175–204 (cit. on pp. 12, 52).
- — (1968b). "Entropy and the complexity of graphs: II. The information content of digraphs and infinite graphs". In: *The Bulletin of mathematical biophysics* 30.2, pp. 225–240 (cit. on pp. 12, 52).
- G. B. Müller (2007). "Evo–devo: extending the evolutionary synthesis". In: *Nature Reviews Genetics* 8.12, pp. 943–949 (cit. on pp. iv, 2, 3, 22).
- R. R. Murphy (2004). "Trial by fire [rescue robots]". In: *Robotics & Automation Magazine, IEEE* 11.3, pp. 50–61 (cit. on p. 28).
- K. Nagatani, S. Kiribayashi, Y. Okada, K. Otake, K. Yoshida, S. Tadokoro, T. Nishimura, T. Yoshida, E. Koyanagi, M. Fukushima, S. Kawatsuma (2013). "Emergency response to the nuclear accident at the Fukushima Daiichi Nuclear Power Plants using mobile rescue robots". In: *Journal of Field Robotics* 30.1, pp. 44–63 (cit. on p. 28).
- M. E. J. Newman (2006). "Modularity and community structure in networks". In: *Proceedings of the National Academy of Sciences* 103.23, pp. 8577–8582 (cit. on pp. vi, 19, 51).
- Y. Niv, D. Joel, I. Meilijson, E. Ruppin (Jan. 2002). "Evolution of Reinforcement Learning in Uncertain Environments: A Simple Explanation for Complex Foraging Behaviors". In: *Adaptive Behavior* 10.1, pp. 5–24. ISSN: 1059-7123. DOI: 10.1177/10597123020101001 (cit. on p. 10).
- S. Nolfi, D. Floreano (2001). *Evolutionary Robotics. The Biology, Intelligence, and Technology of Self-organizing Machines*. MIT Press (cit. on pp. iii, 1, 5).
- M. A. Oliveira, S. Doncieux, J.-B. Mouret, C. Peixoto Santos (2013). "Optimization of Humanoid Walking Controller: Crossing the Reality Gap". In: *Proceedings of Humanoids*, pp. 1–7 (cit. on p. 6).
- P.-Y. Oudeyer, F. Kaplan, V. V. Hafner (2007). "Intrinsic motivation systems for autonomous mental development". In: *Evolutionary Computation, IEEE Transactions on* 11.2, pp. 265–286 (cit. on pp. iv, 2).
- M. Parter, N. Kashtan, U. Alon (2007). "Environmental variability and modularity of bacterial metabolic networks". In: *BMC Evolutionary Biology* 7.1, p. 169 (cit. on p. 18).
- — (2008). "Facilitated variation: how evolution learns from past environments to generalize to new environments". In: *PLoS Computational Biology* 4.11 (cit. on pp. ix, 58).

- M. Pavlicev, G. P. Wagner (2012). "Coming to grips with evolvability". In: *Evol. Educ. Outreach* 5.2, pp. 231–244 (cit. on pp. 2, 38).
- A. Pérez-Escudero, G. Polavieja (2007). "Optimally wired subnetwork determines neuroanatomy of Caenorhabditis elegans". In: *Proceedings of the National Academy of Sciences* 104.43, pp. 17180–17185 (cit. on p. 19).
- L. Peshkin (2007). "Structure induction by lossless graph compression". In: *Data Compression Conference*. IEEE, pp. 53–62 (cit. on pp. 12, 51).
- R. Pfeifer, J. Bongard (2007). *How the body shapes the way we think: a new view of intelligence*. MIT press (cit. on pp. iii, 1, 9).
- R. Pfeifer, M. Lungarella, F. Iida (2007). "Self-organization, embodiment, and biologically inspired robotics". In: *science* 318.5853, pp. 1088–1093 (cit. on pp. iii, 1).
- F. Pfeiffer (2007). "The TUM walking machines". In: *Philosophical transactions. Series A, Mathematical, physical, and engineering sciences* 365.1850, pp. 109–31. DOI: 10 . 1098 / rsta . 2006 . 1922 (cit. on pp. viii, 49).
- M. Pigliucci (Jan. 2008). "Is evolvability evolvable?" In: *Nature Reviews Genetics* 9.1, pp. 75–82. ISSN: 1471-0064. DOI: 10 . 1038/nrg2278 (cit. on pp. v, 2, 4, 17, 38, 58).
- M. M. Polycarpou, A. J. Helmicki (1995). "Automated fault detection and accommodation: a learning systems approach". In: *IEEE Transactions on Systems, Man and Cybernetics* 25.11, pp. 1447–1458 (cit. on pp. 1, 5, 6).
- R. Potts (1998). "Variability Selection in Hominid Evolution". In: *Evolutionary Anthropology* 7.3, pp. 81–96 (cit. on p. 16).
- A. Pouget, J. M. Beck, W. J. Ma, P. E. Latham (2013). "Probabilistic brains: knowns and unknowns". In: *Nature neuroscience* 16.9, pp. 1170–1178 (cit. on p. 31).
- Z. Qu, C. M. Ihlefeld, Y. Jin, A. Saengdeejing (2003). "Robust fault-tolerant self-recovering control of nonlinear uncertain systems". In: *Automatica* 39.10, pp. 1763–1771. DOI: 10.1016/S0005-1098(03)00181-X (cit. on p. 6).
- R. Ariizumi, M. Tesch, H. Choset, F. Matsuno (Sept. 2014). "Expensive multiobjective optimization for robotics with consideration of heteroscedastic noise". In: *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2014)*, pp. 2230–2235. DOI: 10 . 1109 / IROS . 2014 . 6942863 (cit. on p. 57).
- M. Raibert, K. Blankespoor, G. Nelson, R. Playter (2008). "Bigdog, the rough-terrain quadruped robot". In: *Proc. IFAC*. IFAC Publications Office, Elsevier Ltd., Oxford, UK, pp. 10823–10825 (cit. on pp. iii, 1).
- M. Raibert (May 1986). "Legged robots". In: *Communications of the ACM* 29.6, pp. 499–514. ISSN: 00010782. DOI: 10.1145/5948.5950 (cit. on pp. iii, 1).
- A. Raj, Y.-h. Chen (Jan. 2011). "The wiring economy principle: connectivity determines anatomy in the human brain." In: *PloS one* 6.9, e14832. ISSN: 1932-6203. DOI: 10.1371/journal.pone.0014832 (cit. on pp. 18, 52).
- S. Ramón y Cajal (1899). *Textura del sistema nervioso del hombre y de los vertebrados [Texture of the Nervous System of Man and the Vertebrates]*. Nicolás Moya (cit. on pp. vi, 18).
- C. E. Rasmussen, C. K. I. Williams (2006). *Gaussian processes for machine learning*. MIT Press. ISBN: 0-262-18253-X (cit. on pp. 30, 35, 36).
- J. Reisinger, R. Miikkulainen (2007). "Acquiring evolvability through adaptive representations". In: *Proc. GECCO*. ACM, New York, NY, pp. 1045–1052 (cit. on pp. 39, 48).
- J. Reisinger, K. O. Stanley, R. Miikkulainen (2005). "Towards an Empirical Measure of Evolvability". In: *Proc. GECCO*. Washington, D.C.: ACM, New York, NY, pp. 257–264. DOI: 10 . 1145 / 1102256.1102315 (cit. on pp. 39, 47).
- P. Richerson, R. Bettinger, R. Boyd (2005). "Evolution on a restless planet: Were environmental variability and environmental change major drivers of human evolution?" In: *Handbook of evolution* 2, pp. 223–242 (cit. on p. 16).
- S. Risi, K. O. Stanley (2011). "Enhancing ES-HyperNEAT to Evolve More Complex Regular Neural Networks". In: *Proceedings of GECCO*. ACM, pp. 1539–1546 (cit. on pp. 4, 10).
- S. Risi, C. E. Hughes, K. O. Stanley (2010). "Evolving plastic neural networks with novelty search". In: *Adaptive Behavior* 18.6, pp. 470–491 (cit. on pp. 5, 10–12, 15).
- S. Risi, K. O. Stanley (2010). "Indirectly Encoding Neural Plasticity as a Pattern of Local Rules". In: *Proceedings of SAB*, pp. 533–543 (cit. on pp. 5, 10).
- — (2013). "Confronting the Challenge of Learning a Flexible Neural Controller for a Diversity of Morphologies". In: *Proc. GECCO*, pp. 255–262 (cit. on p. 40).
- D. A. Roff (1990). *Evolutionary Quantitative Genetics*. Springer. ISBN: 978-0-412-12971-1 (cit. on pp. iv, 2).
- N. Rougier, J. Vitay (2006). "Emergence of attention within a neural population". In: *Neural Networks* 19.5, pp. 573–581 (cit. on p. 12).
- M. S. Roulston (1999). "Estimating the errors on measured entropy and mutual information". In: *Physica D: Nonlinear Phenomena* 125.3, pp. 285–294 (cit. on p. 40).
- S. Russell, P. Norvig (2009). *Artificial Intelligence: A Modern Approach*. Anglais. 3rd ed. Upper Saddle River: Prentice Hall. ISBN: 9780136042594 (cit. on pp. iii, 1).
- J. Sacks, W. J. Welch, T. J. Mitchell, H. P. Wynn (1989). "Design and analysis of computer experiments". In: *Statistical science* 4.4, pp. 409–423 (cit. on p. 35).
- K. Sanderson (2010). "Mars rover Spirit (2003-10)". In: *Nature* 463.7281, p. 600 (cit. on pp. vi, 27, 28).
- M. Santello (1998). "Postural hand synergies for tool use". In: *The Journal of Neuroscience* 18.23, pp. 10105–10115 (cit. on p. 29).
- S. Schaal (2006). "Dynamic movement primitives-a framework for motor control in humans and humanoid robotics". In: *Adaptive Motion of Animals and Machines*. Springer, pp. 261–280 (cit. on pp. iv, 2).
- J. Schrum, R. Miikkulainen (2010). "Evolving agent behavior in multiobjective domains using fitness-based shaping". In: *Proceedings of the 12th annual conference on Genetic and evolutionary computation*. ACM, pp. 439–446 (cit. on p. 61).
- D. W. Scott (2009). *Multivariate density estimation: theory, practice, and visualization*. Vol. 383. John Wiley & Sons, Inc. (cit. on p. 43).
- J. Scott (2012). *Social network analysis*. Sage (cit. on p. 51).
- J. Secretan, N. Beato, D. B. D Ambrosio, A. Rodriguez, A. Campbell, K. O. Stanley (2008). "Picbreeder: evolving pictures collaboratively online". In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, pp. 1759–1768 (cit. on p. 4).
- B. Siciliano, O. Khatib (2008). *Springer handbook of robotics*. Springer-Verlag (cit. on pp. 28, 30).
- H. A. Simon (1991). *The architecture of complexity*. Springer (cit. on pp. iv, 2).
- T. W. Simpson, T. M. Mauery, J. J. Korte, F. Mistree (1998). "Comparison of response surface and kriging models for multidisciplinary design optimization". In: *American Institute of Aeronautics and Astronautics* 98.7, pp. 1–16 (cit. on p. 35).

- A. Smola, V. Vapnik (1997). "Support vector regression machines". In: *Advances in neural information processing systems* 9, pp. 155–161 (cit. on p. vi).
- J. Snoek, H. Larochelle, R. P. Adams (2012). "Practical Bayesian Optimization of Machine Learning Algorithms". In: *Advances in Neural Information Processing Systems 25 (NIPS)*, pp. 2951–2959 (cit. on pp. 35, 36).
- R. R. Sokal, F. J. Rohlf (1981). "Biometry: the principles and practice of statistics in biological research 2nd edition." In: (cit. on p. 50).
- R. Solé, S. Valverde (2006). "Are network motifs the spandrels of cellular complexity?" In: *Trends in ecology & evolution* 21.8, pp. 419–422 (cit. on p. 25).
- A. Soltoggio, B. Jones (2009). "Novelty of behaviour as a basis for the neuro-evolution of operant reward learning". In: *Proceedings of GECCO*. ACM, pp. 169–176 (cit. on pp. 5, 10, 12).
- A. Soltoggio (Feb. 2015). "Short-term plasticity as cause-effect hypothesis testing in distal reward learning". In: *Biological Cybernetics* 109.1. arXiv: 1402.0710, pp. 75–94. ISSN: 0340-1200, 1432-0770. DOI: 10.1007/s00422-014-0628-0 (cit. on p. 49).
- A. Soltoggio, J. J. A. Bullinaria, C. Mattiussi, D. Floreano, P. Dürr (2008). "Evolutionary advantages of neuromodulated plasticity in dynamic, reward-based scenarios". In: *Proceedings of ALIFE*. Vol. 11, pp. 569–576 (cit. on pp. vi, 5, 10, 11, 15).
- A. Soltoggio, P. Dürr, C. Mattiussi, D. Floreano (2007). "Evolving neuromodulatory topologies for reinforcement learning-like problems". In: *Proceedings of IEEE-CEC*, pp. 2471–2478 (cit. on p. 10).
- A. Soltoggio, J. J. Steil (2013). "Solving the distal reward problem with rare correlations". In: *Neural computation* 25.4, pp. 940–978 (cit. on pp. 49, 59).
- A. Sproewitz, R. Moeckel, J. Maye, A. Ijspeert (2008). "Learning to move in modular robots using central pattern generators and online optimization". In: *The International Journal of Robotics Research* 27.3-4, pp. 423–443 (cit. on pp. 6, 7).
- K. O. Stanley, R. Miikkulainen (2003). "A taxonomy for artificial embryogeny". In: *Artificial Life* 9.2, pp. 93–130 (cit. on pp. iv, 2, 3, 9, 22, 39, 47).
- K. O. Stanley (2007). "Compositional pattern producing networks: A novel abstraction of development". In: *Genetic Programming and Evolvable Machines* 8.2, pp. 131–162 (cit. on pp. vii, 4, 22, 38, 40, 41).
- K. O. Stanley, D. B. D'Ambrosio, J. Gauci (2009). "A hypercube-based encoding for evolving large-scale neural networks". In: *Artificial Life* 15.2, pp. 185–212 (cit. on pp. v, vi, 3, 4, 9, 10, 12, 15, 22, 23, 38, 42).
- K. O. Stanley, R. Miikkulainen (2002). "Evolving neural networks through augmenting topologies". In: *Evolutionary Computation* 10.2, pp. 99–127. ISSN: 1063-6560 (cit. on pp. iv, 2, 4, 38).
- M. L. Stein (1999). *Interpolation of spatial data: some theory for kriging*. Springer (cit. on p. 36).
- G. Striedter (2005). *Principles of brain evolution*. Sinauer Associates Sunderland, MA (cit. on pp. v, vi, 16–18).
- F. Stulp, O. Sigaud (2013). "Robot Skill Learning: From Reinforcement Learning to Evolution Strategies". In: *Paladyn, Journal of Behavioral Robotics* 4.1, pp. 49–61 (cit. on p. 6).
- N. P. Suh (1990). *The principles of design*. Vol. 226. Oxford University Press (cit. on pp. vi, 18).
- R. S. Sutton, A. G. Barto (Sept. 1998). *Reinforcement learning: An introduction*. The MIT press, p. 360. ISBN: 0262193981. DOI: 10.1016/S1364-6613(99)01331-5 (cit. on pp. iv, 2, 6, 10, 11).
- M. E. Taylor, P. Stone (2009). "Transfer learning for reinforcement learning domains: A survey". In: *The Journal of Machine Learning Research* 10, pp. 1633–1685 (cit. on p. 57).
- R. Tedrake, T. Zhang, H. Seung (2005). "Learning to walk in 20 minutes". In: *Proceedings of Yale workshop on Adaptive and Learning Systems* (cit. on pp. 6, 7).
- R. A. Téllez, C. Angulo, D. E. Pardo (2006). "Evolving the Walking Behaviour of a 12 DOF Quadruped Using a Distributed Neural Architecture". In: *Biologically Inspired Approaches to Advanced Information Technology*. Ed. by A. Ijspeert, T. Masuzawa, and S. Kusumoto. Vol. 3853. LNCS. Springer Berlin Heidelberg, pp. 5–19. ISBN: 978-3-540-31253-6. DOI: 10.1007/11613022_4 (cit. on p. 40).
- J. B. Tenenbaum, V. De Silva, J. C. Langford (2000). "A global geometric framework for nonlinear dimensionality reduction". In: *Science* 290.5500, pp. 2319–2323 (cit. on pp. viii, 52).
- M. Tesch, J. Schneider, H. Choset (2011). "Using response surfaces and expected improvement to optimize snake robot gait parameters". In: *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, pp. 1069–1074 (cit. on p. 57).
- — (2013). "Expensive multiobjective optimization for robotics". In: *Robotics and Automation (ICRA), 2013 IEEE International Conference on*. IEEE, pp. 973–980 (cit. on p. 57).
- E. Thelen (1995). "Motor development: a new synthesis". In: *American psychologist*, pp. 79–95 (cit. on p. 29).
- S. Thrun, M. Montemerlo, H. Dahlkamp, D. Stavens, A. Aron, J. Diebel, P. Fong, J. Gale, M. Halpenny, G. Hoffmann (2006). "Stanley: The robot that won the DARPA Grand Challenge". In: *Journal of field Robotics* 23.9, pp. 661–692 (cit. on p. 28).
- J. Togelius, T. Schaul, D. Wierstra, C. Igel, J. Schmidhuber (2009). "Ontogenetic and phylogenetic reinforcement learning". In: *Kuenstliche Intelligenz*, pp. 30–33 (cit. on pp. iv, 2).
- P. Tonelli, J.-B. Mouret (2011). "On the Relationships between Synaptic Plasticity and Generative Systems." In: *Proceedings of GECCO*. ACM, pp. 1531–1538 (cit. on p. 5).
- P. Tonelli, J.-B. Mouret (2013). "On the Relationships between Generative Encodings, Regularity, and Learning Abilities when Evolving Plastic Artificial Neural Networks". In: *PloS One* 8.11, e79138 (cit. on pp. 5, 41, 57).
- T. Trappenberg (2010). *Fundamentals of computational neuroscience*. Oxford University Press (cit. on p. 59).
- A. M. Turing (1950). "Computing machinery and intelligence". In: *Mind* 59.236, pp. 433–460 (cit. on pp. iii, iv, 1, 2, 6).
- J. Urzelai, D. Floreano (2001). "Evolution of adaptive synapses: Robots with fast adaptive behavior in new environments". In: *Evolutionary Computation* 9.4, pp. 495–524. ISSN: 1063-6560 (cit. on pp. 4, 5, 10, 49).
- V. K. Valsalam, R. Miikkulainen (2008). "Modular neuroevolution for multilegged locomotion". In: *Proc. GECCO*. ACM, New York, NY, pp. 265–272 (cit. on pp. 37, 40).
- P. Vamplew, R. Dazeley, A. Berry, R. Issabekov, E. Dekker (2011). "Empirical evaluation methods for multiobjective reinforcement learning algorithms". In: *Machine Learning* 84, pp. 51–80. ISSN: 08856125. DOI: 10.1007/s10994-010-5232-5 (cit. on p. 57).
- P. Verbancsics, K. O. Stanley (2011). "Constraining Connectivity to Encourage Modularity in HyperNEAT". In: *Proceedings of GECCO*. ACM, pp. 1483–1490 (cit. on pp. 3, 22).
- V. Verma, G. Gordon, R. Simmons, S. Thrun (2004). "Real-time fault diagnosis". In: *Robotics & Automation Magazine* 11.2, pp. 56–66 (cit. on p. 28).

- M. Visinsky, J. Cavallaro, I. Walker (Jan. 1994). "Robotic fault detection and fault tolerance: A survey". In: *Reliability Engineering & System Safety* 46.2, pp. 139–158 (cit. on p. 5).
- A. Wagner (Jan. 2008). "Robustness and evolvability: a paradox resolved". en. In: *Proceedings of the Royal Society B: Biological Sciences* 275.1630, pp. 91–100. ISSN: 0962-8452, 1471-2954. DOI: 10.1098/rspb.2007.1137 (cit. on p. 58).
- — (Oct. 2013). *Robustness and Evolvability in Living Systems*. en. Princeton University Press. ISBN: 9781400849383 (cit. on p. 58).
- G. P. Wagner, J. Mezey, R. Calabretta (2001). "Modularity. Understanding the development and evolution of complex natural systems". In: MIT Press. Chap. Natural selection and the origin of modules (cit. on pp. v, vi, 17, 18).
- G. P. Wagner, L. Altenberg (1996). "Perspective: Complex adaptations and the evolution of evolvability". In: *Evolution* 50.3, pp. 967–976 (cit. on pp. iv, 2, 3, 22, 38, 58).
- G. P. Wagner, M. Pavlicev, J. M. Cheverud (Dec. 2007). "The road to modularity." In: *Nature Reviews Genetics* 8.12, pp. 921–31. ISSN: 1471-0064. DOI: 10.1038/nrg2267 (cit. on pp. iv–vi, 2, 4, 17–19, 25).
- U. Wagner, S. Gais, H. Haider, R. Verleger, J. Born (2004). "Sleep inspires insight". In: *Nature* 427.6972, pp. 352–355 (cit. on p. 32).
- R. A. Watson, G. P. Wagner, M. Pavlicev, D. M. Weinreich, R. Mills (Apr. 2014). "The evolution of phenotypic correlations and "developmental memory"". en. In: *Evolution* 68.4, pp. 1124–1138. ISSN: 00143820. DOI: 10.1111/evo.12337 (cit. on pp. ix, 58).
- K. Weicker (2002). "Performance measures for dynamic environments". In: *Parallel Problem Solving from Nature – PPSN VII*. Springer Berlin Heidelberg, pp. 64–73 (cit. on p. 38).
- J. Weingarten, G. Lopes, M. Buehler, R. Groff, D. Koditschek (2004). "Automated gait adaptation for legged robots". In: *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*. Vol. 3, pp. 2153–2158 (cit. on p. 7).
- Q. Wen, D. B. Chklovskii (May 2008). "A cost-benefit analysis of neuronal morphology." In: *Journal of neurophysiology* 99.5, pp. 2320–8. ISSN: 0022-3077. DOI: 10.1152/jn.00280.2007 (cit. on p. 52).
- M. J. West-Eberhard (Feb. 2003). *Developmental Plasticity and Evolution*. en. Oxford University Press. ISBN: 9780199880737 (cit. on pp. iv, 2, 3, 58, 59).
- S. Whiteson (2012). "Evolutionary Computation for Reinforcement Learning". In: *Reinforcement Learning: State of the Art*. Springer, pp. 326–355 (cit. on pp. iv, 2).
- D. M. Wolpert, Z. Ghahramani, J. R. Flanagan (2001). "Perspective and Problems in Motor Learning". In: *Trends in Cognitive Sciences* 5.11, pp. 487–494 (cit. on pp. iv, 2).
- R. J. Woods, J. E. Barrick, T. F. Cooper, U. Shrestha, M. R. Kauth, R. E. Lenski (2011). "Second-order selection for evolvability in a large Escherichia coli population." In: *Science* 331.March, pp. 1433–1436. ISSN: 0036-8075. DOI: 10 . 1126 / science . 1198914 (cit. on p. 4).
- S. Wright (1932). "The roles of mutation, inbreeding, crossbreeding and selection in evolution". In: *Proceedings of the VI International Congress of Genetrics*, pp. 356–366 (cit. on p. 52).
- D. R. Yoerger (2008). "Underwater robotics". In: *Springer handbook of robotics*. Springer, pp. 987–1008 (cit. on pp. vi, 27, 28).
- J. Yosinski, J. Clune, D. Hidalgo, S. Nguyen, J. Zagal, H. Lipson (2011). "Evolving Robot Gaits in Hardware: the HyperNEAT Generative Encoding Vs. Parameter Optimization". In: *Proceedings of the European Conference on Artificial Life (ECAL)*, pp. 1–8 (cit. on pp. iii, 1, 7, 28, 37, 38, 42).
- H. Zenil, F. Soler-Toscano, K. Dingle, A. A. Louis (2013). "Graph Automorphism and Topological Characterization of Synthetic and Natural Complex Networks by Information Content". In: *arXiv preprint arXiv:1306.0322* (cit. on pp. 12, 52).
- V. Zykov, J. Bongard, H. Lipson (2004). "Evolving dynamic gaits on a physical robot". In: *Proceedings of Genetic and Evolutionary Computation Conference, Late Breaking Paper (GECCO)*. Vol. 4 (cit. on p. 40).

Curriculum vitæ

— **Date of birth** May 20th, 1981
— **Website** http://pages.isir.upmc.fr/~mouret/
— **Keywords** autonomous robotics, evolutionary robotics, system biology, optimization, machine learning
— **Highlights:**
- One paper in Nature (2015, senior author)
- ERC Starting grant (2015-2020)
- ANR "young researcher" grant (jan. 2013-dec. 2015)
- 15 articles in peer-reviewed journals, 27 articles in peer-reviewed conferences
- 2 PhD students who defended (+1 ongoing), 2 postdocs
- 2 best papers in major evolutionary computation conferences

Education

— **dec. 2008** PhD in computer science, Pierre and Marie University (UPMC), France
- Supervisors: Jean-Arcady Meyer and Séphane Doncieux
- Funding: PhD scholarship, received from the French ministry of research
- Title: *Multiple selective pressures for the evolution of neural networks in robotics*
  - Marc Schoenauer (INRIA Saclay Ile de France), reviewer
  - Yves Duthen (Univ. UTI1 Capitole, IRIT), reviewer
  - Hélène Paugam-Moisy (Univ. Lyon 2, LIRIS), examiner
  - Patrice Perny (UPMC/CNRS, LIP6), examiner
  - Philippe Bidaud (UPMC/CNRS, ISIR), examiner
  - Stéphane Doncieux (UPMC/CNRS, ISIR), co-supervisor
  - Jean-Arcady Meyer (UPMC/CNRS, ISIR), main supervisor

— **sep. 2005** Master "Artificial intelligence and decision", Pierre and Marie University, France (top 10%)
— **sep. 2004** Master (engineering school) "Scientific computation and image", EPITA, France (top 10%)

Academic positions

— **sep. 2014–sep. 2015** CNRS *délégation* (section 7) [sabbatical]
— **sep. 2009– ...** *Maître de conférences* (eq. assistant professor), Pierre and Marie University, Paris, France
— **sep. 2008–sep. 2009** Temporary assistant professor (ATER), Pierre and Marie University, Paris, France

Research stays

— **2014 (5 months)** Visiting professor, Technische Universität Darmstadt, Germany (invited by Jan Peters)
The Intelligent Autonomous System Lab, headed by J. Peters, focuses on machine learning for robotics.
— **2011 and 2012 (1 + 1 months)** Visiting professor, Cornell University, NY, USA (invited by Hod Lipson)
Cornell's Creative Machines lab, headed by H. Lipson, uses machine learning and evolutionary algorithms to make robots "creative" problem solvers.
— **2010 (3 months)** Visiting professor, University of Vermont, VT, USA (invited by Josh Bongard)
Josh Bongard published a paper about "resilient robotics" (Bongard et al., Science, 2006) which has been highly influential in my work.

Awards and nominations

— **Best video, AAAI video competition 2014:** Joost Huizinga, Jean-Baptiste Mouret, Jeff Clune. Evolving Neural Networks That Are Both Modular and Regular
— **Nominated for best student video, AAAI Video competition 2013:** A. Cully, S. Koos and J.-B. Mouret. Fast Damage Recovery with the T-Resilience Algorithm.
— **Winner of the GECCO 2012 Vizualizing Evolution competition** , with Jeff Clune
— **Highly recommended paper, category "Control of CLAWAR", CLAWAR 2011:** S. Koos and J-B. Mouret. "Online Discovery of Locomotion Modes for Wheel-Legged Hybrid Robots: a Transferability-based Approach". In: Proceedings of CLAWAR. 2011, pp. 70–77
— **Best paper, GECCO 2011, GDS track:** (20 accepted papers in the track) P. Tonelli and J.-B. Mouret. "On the Relationships between Synaptic Plasticity and Generative Systems". In: Proceedings of the 13th Annual Conference on Genetic and Evolutionary Computation (GECCO)
— **Nominated for best paper, GECCO 2010, ER/Alife track:** (3 of 20 nominated in track) S. Koos, J.-B. Mouret. "Crossing the Reality Gap in Evolutionary Robotics by Promoting Transferable Controllers". In: Proceedings of the 12th annual in proceedings on Genetic and evolutionary computation (GECCO). 2010, pp. 119–126
— **Best student paper, IEEE CEC 2009:** (448 accepted papers) J.-B. Mouret and S. Doncieux. "Evolving modular neural-networks through exaptation". In: IEEE Congress on Evolutionary Computation (CEC).

Selected software

See: http://github.com/jbmouret/
— **Sferes**$_{v2}$ Framework for multi-objective evolutionary computation and neuro-evolution. Template-based C++08, optimized for multi-core architectures. Used in more than 30 scientific papers, from our group and a few others in the world.
— **Limbo** Framework for multi-objective Bayesian optimization. Template-based C++11, optimized for multi-core architectures

Selected invited talks

— (international workshop) 10/2014: Nature-inspired techniques for robotics (satellite workshop of Parallel Problem Solving from Nature, PPSN), Ljubljana, Slovenia – *Damage recovery is a reality gap problem*

— (general public conference, about 70 attendees) 02/2014: "Forum des Sciences Cognitives"fp, Paris, France – *Robots résilients*

— (keynote, about 100 attendees) 11/2013, Entrepreneurs' organization, Mumbai (India) – *Why is autonomous robotics different from industrial robotics?* (Entrepreneurs' organization is one of the largest organization of entrepreneurs worldwide)

— (mini-conference, about 50 attendees) 04/2013 "Cognition, Adaptation et complexité: des êtres vivants aux robots", Sorbonne, Paris, France – *Les origines évolutionnistes de la modularité*

– (seminar, about 30 attendees) 03/2013, EcoEvo department, UPMC, Paris, France – *Les origines évolutionnistes de la modularité*

– (seminar, about 30 attendees) 02/2013, CIRB department, Collège de France, Paris, France – *The evolutionary origins of modularity*

Funding & grants

— **Resibots (ERC Starting Grant)** [2015-2020]
- Role: Principal Investigator (*porteur*)
- 1.5 M € (European Research Council)

— **Creadapt (ANR "young researcher")** [jan. 2013 – dec. 2015] ANR-12-JS03-0009
- Role: Principal Investigator (*porteur*)
- 249 k € (Agence Nationale pour la Recherche)
- The Creadapt project is centered on evolution for adaptation, both in biology and robotics. It includes the development of new algorithms and their implementation on a new wheel-leg hybrid robot.
- http://www.creadapt.net

— **EvoNeuro (ANR DEFI)** [dec. 2009 – jul. 2013] ANR-09-EMER-005-01/02
- Role: member of the ISIR team (co-PI)
- Partners: ISIR (UPMC), Neurobiologie des Processus Adaptatifs (UPMC)
- 583,017 k € (292 k € for ISIR)
- The EvoNeuro project aims at cross-fertilizing evolutionary robotics, neuro-evolution, and computational neurosciences.
- http://pages.isir.upmc.fr/EvoNeuro/

— **PhD scholarship** [sep. 2005 – sept. 2008] Awarded by the French Ministry of research

Supervision

In the references, names of the students/post-docs/engineer whom I directly supervised are underlined.

Post-doctoral students

— **nov. 2013–feb.2015: Danesh Tarapore** – Evolvability and creative adaptation.
- Funding: ANR (Agence Nationale pour la Recherche)
- Supervision rate: 100%
- Main publications:
  - (journal paper) Tarapore, D. and Mouret, J.-B. (2014). *Evolvability signatures of generative encodings: beyond standard performance benchmarks*. Information Sciences. To appear.
  - (accepted, journal paper) Cully, A., Clune, J. and D. Tarapore, and Mouret, J.-B. (2015). *Robots that can adapt like natural animals.* Nature. To appear.
  - (selective conf.) Tarapore, D. and Mouret, J.-B. (2014). *Comparing the evolvability of generative encoding schemes.* Artificial Life 14: Proceedings of the Fourteenth International Conference on the Synthesis and Simulation of Living Systems, MIT Press, publisher. Pages 55-62.

— **sep. 2011–sep. 2012 (post-doc / ATER): Sylvain Koos** – Applying the transferability approach to damage recovery.
- Funding: UPMC (ATER)
- Supervision rate: 100%
- Main publications:
  - (journal paper) Koos, S., Cully, A. and Mouret, J.-B. (2013). *Fast Damage Recovery in Robotics with the T-Resilience Algorithm.* International Journal of Robotics Research. Vol 32 No 14 Pages 1700-1723.

PhD students

— **2012-...: Antoine Cully** – Creative adaptation for damage recovery
- Co-advisor: S. Doncieux (JBM: 90%, SD: 10%)
- Funding: UPMC/DGA scholarship (EDITE)
- Defense planned in September 2015
- Main publications:
  - (accepted, journal paper) Cully, A., Clune, J. and D. Tarapore, and Mouret, J.-B. (2015). *Robots that can adapt like natural animals.* Nature. To appear.
  - (journal paper) Cully, A. and Mouret, J.-B. (2015). *Evolving a Behavioral Repertoire for a Walking Robot*. To appear in Evolutionary Computation.
  - (selective conf.) Jehanno, J.-M., Cully, A., Grand, C. and Mouret, J.-B. (2014). *Design of a Wheel-Legged Hexapod Robot for Creative Adaptation.* Proceedings of CLAWAR. Pages 267-276.
  - (journal paper) Koos, S., Cully, A. and Mouret, J.-B. (2013). *Fast Damage Recovery in Robotics with the T-Resilience Algorithm.* International Journal of Robotics Research. Vol 32 No 14 Pages 1700-1723.
  - (selective conf.) Cully, A. and Mouret, J.-B. (2013). *Behavioral Repertoire Learning in Robotics.* Proc. of Genetic and Evolutionary Computation Conference (GECCO). Pages 175-182.

— **2008–2012: Paul Tonelli** – On the relationships between generative encodings and synaptic plasticity in neuro-evolution (*Relations entre plasticité synaptique et régularité des codages en neuro-évolution*)
- Co-advisor: S. Doncieux (JBM: 75%, SD: 25%)
- Funding: scholarship from the Monaco principality
- PhD defended on the 25th of June (2012):

  - Nicolas Bredeche (Univ. Paris-Sud, LRI), reviewer
  - Yves Duthen (Univ. UT1 Capitole, IRIT), reviewer
  - Hélène Paugam-Moisy (Univ. Lyon 2, LIRIS), examiner
  - Olivier Sigaud (UPMC, ISIR), examiner
  - Stéphane Doncieux (UMPC, ISIR), supervisor
  - Jean-Baptiste Mouret (UPMC, ISIR), co-supervisor
- Main publications:
  - (journal paper) Tonelli, P. and Mouret, J.-B. (2013). *On the Relationships between Generative Encodings, Regularity, and Learning Abilities when Evolving Plastic Artificial Neural Networks.* PLoS One. Vol 8 No 11 Pages e79138.
  - (selective conf.) Tonelli, P. and Mouret, J.-B. (2011). *On the Relationships between Synaptic Plasticity and Generative Systems.* Proceedings of the 13th Annual Conference on Genetic and Evolutionary Computation. Pages 1531–1538. **(Best paper of the Generative and Developmental Systems (GDS) track).**
  - (selective conf.) Tonelli, P. and Mouret, J.-B. (2011). *Using a Map-Based Encoding to Evolve Plastic Neural Networks.* Proceedings of IEEE Symposium Series on Computational Intelligence. Pages 9-16.
  - (selective conf.) Tonelli, P. and Mouret, J.-B. and Doncieux, S. (2009). *Influence of Promoter Length on Network Convergence in GRN-based Evolutionary Algorithms.* Proceedings of The 10th European Conference on Artificial Life, Springer, publisher. Pages 302-309.

— **2008–2011: Sylvain Koos** – The transferability: a general approach to cross the reality gap, to generalize in machine learning, and to behavior adaptation in robotics (*L'approche par transférabilité : une réponse aux problèmes de passage à la réalité, de généralisation et d'adaptation*).

- Co-advisor: S. Doncieux (JBM: 50%, SD: 50%)
- funding: French ministry of research scholarship (EDITE)
- PhD defended on the 30th of November, 2011:
  - Pierre-Yves Oudeyer (INRIA Bordeaux Sud-Ouest), reviewer
  - Marc Schoenauer, (INRIA Saclay Ile-de-France), reviewer
  - Josh Bongard (University of Vermont, USA), examiner
  - Raja Chatila (UPMC, ISIR), examiner
  - Stéphane Doncieux (UPMC, ISIR), supervisor
  - Jean-Baptiste Mouret (UPMC, ISIR), co-supervisor
- Main publications:
  - (journal paper) Koos, S., Cully, A. and Mouret, J.-B. (2013). *Fast Damage Recovery in Robotics with the T-Resilience Algorithm. International Journal of Robotics Research.* Vol 32 No 14 Pages 1700-1723.
  - (journal paper) Koos, S., Mouret, J.-B. and Doncieux, S. (2013). *The Transferability Approach: Crossing the Reality Gap in Evolutionary Robotics.* IEEE Transactions on Evolutionary Computation. Vol 17 No 1 Pages 122 - 145.
  - (selective conf.) Koos, S. and Mouret, J-B. (2011). *Online Discovery of Locomotion Modes for Wheel-Legged Hybrid Robots: a Transferability-based Approach.* Proceedings of CLAWAR, World Scientific Publishing Co., publisher. Pages 70-77.
  - (selective conf.) Pinville, T. and Koos, S., Mouret, J-B. and Doncieux, S. (2011). *How to Promote Generalisation in Evolutionary Robotics: the ProGAb Approach.* GECCO'11: Proceedings of the 13th annual conference on Genetic and evolutionary computation ACM, publisher. Pages 259–266.
  - (selective conf.) Koos, S., Mouret, J.-B. and Doncieux, S. (2010). *Crossing the Reality Gap in Evolutionary Robotics by Promoting Transferable Controllers.* GECCO'10: Proceedings of the 12th annual conference on Genetic and evolutionary computation ACM, publisher. Pages 119–126. Nominated for best paper in the ER/Alife track.
  - (selective conf.) Koos, S., Mouret, J.-B. and Doncieux, S. (2009). *Automatic system identification based on coevolution of models and tests.* IEEE Congress on Evolutionary Computation, 2009 (CEC 2009). Pages 560–567.

Research engineers

— **2013-2014 (senior engineer): J.-M. Jehanno** – Mechanical design of a wheel-legged hybrid robot; funding: ANR (Agence Nationale pour la Recherche)

Master students

— **2014 (3 months, M1)** Luigi Tedesco (ENSTA) - Learning with Bayesian optimization
— **2014 (3 months, M1)** Frédéric Lauron (Polytech'Paris-UPMC) – Robotic finch (CAD design)
— **2014 (3 months, M1)** Walid Abderrahmani (Polytech'Paris-UPMC) – Robotic finch (software)
— **2013 (3 months, M1)** Mathieu Nassar (Polytech'Paris-UPMC) – Repertoire learning
— **2012 (6 months, M2)** Antoine Cully (Polytech'Paris-UPMC), now PhD student in our group – damage recovery with a hexapod robot

Professional activities

Organization of international conferences

— **GECCO'2015** (about 500 participants) Co-chair of the "Generative and Developmental Systems" track (with Sebastian Risi, IT University of Copenhagen, DK) – track chairs are responsible of the review process (handling reviews and taking the acceptance decisions).
— **CLAWAR'2011** (about 130 participants) The 14th International Conference on Climbing and Walking Robots and the Support Technologies for Mobile Machines, Paris, France (local co-organizer)

— **SAB'2010** (about 150 participants) The 11th conference on Simulation of Adaptive Behavior: From animals to animats, Paris, France (local co-organizer)

Organization of international workshops

— **At ALIFE/ECAL** (2012: East Lansing, MI, USA, 2013: Taormina, Italy, 2014: New York, USA; 2015: York, UK) *Evolution in physical systems* (about 40 participants, co-organizers: J. Rieffel, N. Bredeche, E. Haasdijk)
— **At CNS 2012** (Atlanta, USA) *Modern evolutionary algorithms in computational neuroscience: tools to parametrize, explore model properties and design model structures* (about 50 participants, co-organizers: B. Girard, S. Doncieux, D. Sheynikhovich)
— **At GECCO 2012** (Philadelphia, USA) *Evolutionary Developmental Robotics: EvoDevoRobo* (about 60 participants, co-organizers: S. Doncieux, Y. Jin)
— **At IROS 2009** (Saint-Louis, USA) *Exploring New Horizons in Evolutionary Design of Robots* (about 40 participants, co-organizers: S. Doncieux, N. Bredeche, proceedings published as a Springer book)

Tutorials

— **ALIFE'2014, ECAL 2015 & GECCO 2015** [3 hours] Evolutionary Robotics (with S. Doncieux and N. Bredeche)
— **ALIFE'2014** [3 hours] Creating publication-quality figures with Python and Matplotlib (https://github.com/jbmouret/matplotlib_for_papers)

Program committee member

GECCO (about 500 participants, acceptance rate: about 35%) and ALIFE (about 500 participants, acceptance for oral presentations: about 25%) are the two main conferences about artificial evolution.

— **Genetic and Evolutionary Computation Conference** (GECCO, since 2010)
— **European Conference on Artificial Life (ECAL)** (since 2015)
— **Artificial Life (ALIFE)** (since 2014)
— **Simulation of Adaptive Behavior (SAB)** (since 2010)
— **European Conference on the Applications of Evolutionary Computation** (Evo*, since 2013)
— **International conference on Artificial Evolution** (EA, 2009, 2010)
— **International Workshop on Evolutionary and Reinforcement Learning for Autonomous Robot Systems** (ERLARS, since 2013)
— **International Conference on Climbing and Walking Robots and the Support Technologies for Mobile Machines** (CLAWAR, 2011)

Reviewer

— **IEEE Transactions on Evolutionary Computation** (top journal in evolutionary computation)
— **Evolutionary Computation** (2nd top journal in evolutionary computation)
— **Artificial Life** (top journal in artificial life)
— **PLoS One** (one of the top multi-disciplinary journals)
— **IEEE Transactions on Robotics** (1st/2nd journal in robotics)
— **Proceedings of the Royal Society B** . (one of the top biology journals)
— **Soft Computing**
— **Review editor in Frontiers in AI and Robotics** (evolutionary robotics section).

PhD committees

— **2015** Fabien Benureau (supervised by P.-Y. Oudeyer), Bordeaux University. *Self-Exploration of Sensorimotor Spaces in Robots*. Role: external examiner.
— **2014** Adam Davies (supervised by Richard Watson), University of Southampton. *On the interaction of function, constraint and complexity in evolutionary systems*. Role: external examiner and reviewer.

Teaching activities

— **2008-...** Full load of teaching (192+ hours/year) for master student in robotics (Polytech'Paris-UPMC); about 30% of classic classes and 70% of hands-on classes; topics:

- artificial intelligence for robotics
- object oriented programming
- C++ language
- Unix system programming
- supervision of student projects

Publication list

All my publications can be downloaded on:
http://pages.isir.upmc.fr/~mouret/website/publications.xhtml
Google scholar page: http://scholar.google.fr/citations?user=lp8V_UYAAAAJ&hl=en&oi=ao

Peer-reviewed journals

A. Cully, J. Clune, D. Tarapore, J.-B. Mouret (2015). "Robots that can adapt like natural animals". In: *Nature*, pp. 1–4.

A. Cully, J.-B. Mouret (2015). "Evolving a Behavioral Repertoire for a Walking Robot". In: *Evolutionary Computation Journal* 1, pp. 1–33.

S. Doncieux, N. Bredeche, J.-B. Mouret, A. E. Eiben (2015). "Evolutionary robotics: what, why, and where to". In: *Frontiers in Robotics and AI* 2. DOI: 10.3389/frobt.2015.00004.

K. O. Ellefsen, J.-B. Mouret, J. Clune (2015). "Neural Modularity Helps Organisms Evolve to Learn New Skils without Forgetting Old Skills". In: *PLoS Computational Biology* 1.4, e1004128.

R. Reuillon, C. Schmitt, R. De Aldama, J.-B. Mouret (2015). "A New Method to Evaluate Simulation Models: The Calibration Profile (CP) Algorithm". In: *Journal of Artificial Societies and Social Simulation* 18.1, p. 12.

D. Tarapore, J.-B. Mouret (2015). "Evolvability signatures of generative encodings: beyond standard performance benchmarks". In: *Information Sciences*, pp. 1–20.

S. Doncieux, J.-B. Mouret (2014). "Beyond black-box optimization: a review of selective pressures for evolutionary robotics". In: *Evolutionary Intelligence* 7.2, pp. 71–93. DOI: 10.1007/s12065-014-0110-x.

J. Clune*, J.-B. Mouret, H. Lipson (2013). "The evolutionary origins of modularity". In: *Proceedings of the Royal Society B* 280 (J. Clune and J.-B. Mouret contributed equally to this work), p. 20122863. DOI: 10.1098/rspb.2012.2863.

S. Koos, A. Cully, J.-B. Mouret (2013). "Fast Damage Recovery in Robotics with the T-Resilience Algorithm". In: *International Journal of Robotics Research* 32.14, pp. 1700–1723. DOI: 10 . 1177 / 0278364913499192.

S. Koos, J.-B. Mouret, S. Doncieux (Feb. 2013). "The Transferability Approach: Crossing the Reality Gap in Evolutionary Robotics". In: *IEEE Transactions on Evolutionary Computation* 17.1, pp. 122–145. DOI: 10.1109/TEVC.2012.2185849.

P. Tonelli, J.-B. Mouret (Nov. 2013). "On the Relationships between Generative Encodings, Regularity, and Learning Abilities when Evolving Plastic Artificial Neural Networks". In: *PLoS One* 8.11, e79138. DOI: 10.1371/journal.pone.0079138.

J.-B. Mouret, S. Doncieux (2012). "Encouraging Behavioral Diversity in Evolutionary Robotics: an Empirical Study". In: *Evolutionary Computation* 20.1, pp. 91–133.

— (2008b). "MENNAG: a modular, regular and hierarchical encoding for neural-networks based on attribute grammars". In: *Evolutionary Intelligence* 1.3, pp. 187–207.

E. de Margerie, J.-B. Mouret, S. Doncieux, J.-A. Meyer (2007). "Artificial evolution of the morphology and kinematics in a flapping-wing mini UAV". In: *Bioinspir. Biomim.* 2, pp. 65–82.

T. Geraud, J.-B. Mouret (Jan. 2004). "Fast road network extraction in satellite images using mathematical morphology and Markov random fields". In: *EURASIP Journal of Applied Signal Processing*, pp. 2503–2514.

Edited books

S. Doncieux, N. Bredeche, J.-B. Mouret (2011). *New Horizons in Evolutionary Robotics: Extended Contributions from the 2009 EvoDeRob Workshop*. Ed. by S. Doncieux, N. Bredeche, and J.-B. Mouret. Vol. 341. Studies in Computational Intelligence. Springer.

S. Doncieux, B. Girard, A. Guillot, J. Hallam, J.-A. Meyer, J.-B. Mouret (2010). *From Animals to Animats 11*. Vol. 6226. LNAI. Springer.

Peer-reviewed international conferences

J. Huizinga, J.-B. Mouret, J. Clune (2014). "Evolving Neural Networks That Are Both Modular and Regular: HyperNeat Plus the Connection Cost Technique". In: *Proceedings of GECCO*, pp. 1–8.

J.-M. Jehanno, A. Cully, C. Grand, J.-B. Mouret (2014). "Design of a Wheel-Legged Hexapod Robot for Creative Adaptation". In: *Proceedings of CLAWAR*, pp. 267–276.

D. Tarapore, J.-B. Mouret (2014). "Comparing the evolvability of generative encoding schemes". In: *Artificial Life 14: Proceedings of the Fourteenth International Conference on the Synthesis and Simulation of Living Systems*. MIT Press, pp. 55–62. DOI: http://dx.doi.org/10.7551/978-0-262-32621-6-ch011.

A. Cully, J.-B. Mouret (2013). "Behavioral Repertoire Learning in Robotics". In: *Genetic and Evolutionary Computation Conference (GECCO)*, pp. 175–182.

S. Doncieux, J. Mouret (2013). "Behavioral Diversity with Multiple Behavioral Distances". In: *Proc. of IEEE Congress on Evolutionary Computation, 2013 (CEC 2013)*, pp. 1–8.

J.-B. Mouret, J. Clune (2012). "An Algorithm to Create Phenotype-Fitness Maps". In: *Proceedings of the Thirteenth International Conference on the Simulation and Synthesis of Living Systems (ALIFE 13)*, 593–594 (extended abstract).

S. Koos, J.-B. Mouret (2011). "Online Discovery of Locomotion Modes for Wheel-Legged Hybrid Robots: a Transferability-based Approach". In: *Proceedings of CLAWAR*. World Scientific Publishing Co., pp. 70–77.

T. Pinville, S. Koos, J.-B. Mouret, S. Doncieux (2011). "How to Promote Generalisation in Evolutionary Robotics: the ProGAb Approach". In: *GECCO'11: Proceedings of the 13th annual conference on Genetic and evolutionary computation ACM, publisher*, pp. 259–266.

A. Terekhov, J.-B. Mouret, C. Grand (2011). "Stochastic optimization of a chain sliding mode controller for the mobile robot maneuvering". In: *Proceedings of IEEE / IROS Int. Conf. on Robots and Intelligents Systems*, pp. 4360–4365.

P. Tonelli, J.-B. Mouret (2011a). "On the Relationships between Synaptic Plasticity and Generative Systems". In: *Proceedings of the 13th Annual Conference on Genetic and Evolutionary Computation*. **(Best paper of the Generative and Developmental Systems (GDS) track)**, pp. 1531–1538. DOI: http://doi.acm.org/10.1145/2001576.2001782.

P. Tonelli, J.-B. Mouret (2011b). "Using a Map-Based Encoding to Evolve Plastic Neural Networks". In: *Proceedings of IEEE Symposium Series on Computational Intelligence*, pp. 9–16. DOI: 10.1109/EAIS.2011.5945909.

S. Doncieux, J.-B. Mouret (2010). "Behavioral diversity measures for Evolutionary Robotics". In: *WCCI 2010 IEEE World Congress on Computational Intelligence, Congress on Evolutionary Computation (CEC)*, pp. 1303–1310.

S. Koos, J.-B. Mouret, S. Doncieux (2010). "Crossing the Reality Gap in Evolutionary Robotics by Promoting Transferable Controllers". In: *GECCO'10: Proceedings of the 12th annual conference on Genetic and evolutionary computation ACM, publisher*, pp. 119–126.

J.-B. Mouret, S. Doncieux (2010). "SFERESv2: Evolvin' in the Multi-Core World". In: *WCCI 2010 IEEE World Congress on Computational Intelligence, Congress on Evolutionary Computation (CEC)*, pp. 4079–4086.

J.-B. Mouret, S. Doncieux, B. Girard (2010). "Importing the Computational Neuroscience Toolbox into Neuro-Evolution—Application to Basal Ganglia". In: *GECCO'10: Proceedings of the 12th annual conference on Genetic and evolutionary computation ACM, publisher*, pp. 587–594.

A. Terekhov, J.-B. Mouret, C. Grand (2010a). "Stochastic multi-objective optimization for aggressive maneuver trajectory planning on loose surface". In: *Proceedings of IFAC: the 7th Symposium on Intelligent Autonomous Vehicles*, pp. 1–6.

A. Terekhov, J.-B. Mouret, C. Grand (2010b). "Stochastic optimization of a neural network-based controller for aggressive maneuvers on loose surfaces". In: *Proceedings of IEEE / IROS Int. Conf. on Robots and Intelligents Systems*, pp. 4782–4787.

S. Doncieux, J.-B. Mouret (2009). "Single Step Evolution of Robot Controllers for Sequential Tasks". In: *GECCO'09: Proceedings of the 11th annual conference on Genetic and evolutionary computation*. (abstract and poster). ACM, pp. 1771–1772.

S. Doncieux, J.-B. Mouret, N. Bredeche (Oct. 2009). "Exploring New Horizons in Evolutionary Design of Robots". In: *"Exploring New Horizons in Evolutionary Design of Robots" IROS Workshop*. Saint Louis, USA, pp. 5–12.

S. Koos, J.-B. Mouret, S. Doncieux (2009). "Automatic system identification based on coevolution of models and tests". In: *IEEE Congress on Evolutionary Computation, 2009 (CEC 2009)*, pp. 560–567.

J.-B. Mouret, S. Doncieux (2009a). "Evolving modular neural-networks through exaptation". In: *IEEE Congress on Evolutionary Computation, 2009 (CEC 2009)*. **(Best student paper award)**, pp. 1570–1577.

— (2009b). "Overcoming the bootstrap problem in evolutionary robotics using behavioral diversity". In: *IEEE Congress on Evolutionary Computation, 2009 (CEC 2009)*, pp. 1161–1168.

— (2009c). "Using Behavioral Exploration Objectives to Solve Deceptive Problems in Neuro-evolution". In: *GECCO'09: Proceedings of the 11th annual conference on Genetic and evolutionary computation*. ACM, pp. 627–634.

P. Tonelli, J.-B. Mouret, S. Doncieux (2009). "Influence of Promoter Length on Network Convergence in GRN-based Evolutionary Algorithms". In: *Proceedings of The 10th European Conference on Artificial Life*. Springer, pp. 302–309.

C. Grand, P. Martinelli, J.-B. Mouret, S. Doncieux (June 2008). "Flapping-Wing Mechanism for a bird-Sized UAVS: Design, Modeling and Control". In: *Proceedings of ARK'08 : the 11th Int. Symposium on ADVANCES IN ROBOT KINEMATICS*. Ed. by J. Lenarcic and P. Wenger. France: Springer, pp. 127–136.

J.-B. Mouret, S. Doncieux (2008a). "Incremental Evolution of Animats' Behaviors as a Multi-objective Optimization". In: *From Animals to Animats 10*. Vol. 5040. Springer, pp. 210–219.

J.-B. Mouret, S. Doncieux, J.-A. Meyer (2006). "Incremental Evolution of Target-Following Neuro-controllers for Flapping-Wing Animats". In: *From Animals to Animats: Proceedings of the 9th International Conference on the Simulation of Adaptive Behavior (SAB)*. Ed. by S. Nolfi, G. Baldassare, R. Calabretta, J. Hallamand, D. Marocco, J.-A. Meyer, O. Miglino, and D. Parisi. Rome, Italy, pp. 606–618.

Patents

P. Martinelli, T. Ravasi, C. Grand, S. Doncieux, J.-B. Mouret, E. de Margerie (2008). "Flying Device with Fapping Wings". PCT.

Book chapters, workshops, short papers and minimally-refeered conferences

J.-B. Mouret, P. Tonelli (2014). "Artificial Evolution of Plastic Neural Networks: a few Key Concepts". In: *Growing Adaptive Machines: combining Development and Learning in Artificial Neural Networks*. Ed. by T. Kowaliw, N. Bredeche, and R. Doursat. Vol. 557. Studies in Computational Intelligence (Springer), pp. 251–261.

J.-B. Mouret, S. Koos, S. Doncieux (2012). "Crossing the Reality Gap: a Short Introduction to the Transferability Approach". In: *Proceedings of the workshop "Evolution in Physical Systems"*. ALIFE.

S. Doncieux, J.-B. Mouret, N. Bredeche, V. Padois (2011). "Evolutionary Robotics: Exploring New Horizons". In: *Studies in Computational Intelligence, New Horizons in Evolutionary Robotics*. Ed. by S. Doncieux, N. Bredeche, and J.-B. Mouret. Vol. 341. Springer, pp. 3–25. DOI: 10.1007/978-3-642-18272-3_1.

S. Doncieux, J. Mouret, T. Pinville, P. Tonelli, B. Girard (2011). "The EvoNeuro Approach to Neuro-Evolution". In: *Workshop on Development and Learning in Artificial Neural Networks (DevLeANN)*. -.

J.-B. Mouret (2011). "Novelty-based Multiobjectivization". In: *New Horizons in Evolutionary Robotics: Extended Contributions from the 2009 EvoDeRob Workshop*. Springer, pp. 139–154.

J.-B. Mouret, P. Tonelli (2011). "Artificial Evolution of Plastic Neural Networks: a few Key Concepts". In: *DevLeaNN 2011: A Workshop on Development and Learning in Artificial Neural Networks*. -.

P. Adigbli, C. Grand, J.-B. Mouret, S. Doncieux (2007). "Nonlinear Attitude and Position Control of a Micro Quadrotor using Sliding Mode and Backstepping Techniques". In: *7th European Micro Air Vehicle Conference (MAV07)*. Toulouse, pp. 1–9.

E. de Margerie, J.-B. Mouret, S. Doncieux, J.-A. Meyer, T. Ravasi, P. Martinelli, C. Grand (2007). "Flapping-wing flight in bird-sized UAVs for the ROBUR project: from an evolutionary optimization to a real flapping-wing mechanism". In: *7th European Micro Air Vehicle Conference (MAV07)*. Toulouse, pp. 1–9.

S. Doncieux, J.-B. Mouret, J.-A. Meyer (2007). "Soaring behaviors in UAVs : 'animat' design methodology and current results". In: *7th European Micro Air Vehicle Conference (MAV07)*. Toulouse, pp. 1–10.

S. Doncieux, J.-B. Mouret, A. Angeli, R. Barate, J.-A. Meyer, E. de Margerie (2006). "Building an Artificial Bird: Goals and Accomplishments of the ROBUR Project". In: *Proceedings of the European Micro Aerial Vehicles (EMAV 2006) conference*. Braunschweig, pp. 1–12.

S. Doncieux, J.-B. Mouret, L. Muratet, J.-A. Meyer (2004). "The ROBUR project: towards an autonomous flapping-wing animat". In: *Proceedings of the Journées MicroDrones*. Toulouse, pp. 1–9.

J.-B. Mouret, S. Doncieux, L. Muratet, T. Druot, J.-A. Meyer (2004). "Evolution of neuro-controllers for flapping-wing animats". In: *Proceedings of the Journées MicroDrones*. Toulouse, pp. 1–14.

General Audience

J.-B. Mouret, N. Bredeche, S. Doncieux (2015). *La Robotique Évolutionniste*. Pour La Science.

J.-B. Mouret (Oct. 2005a). *Algorithmes évolutionnistes, deuxième partie : évolution artificielle de créatures*. Linux Magazine France, pp. 42–49.

— (Oct. 2005b). *Concepts fondamentaux des algorithmes évolutionnistes*. Linux Magazine France, pp. 34–41.

Supplementary figures for chapter 3

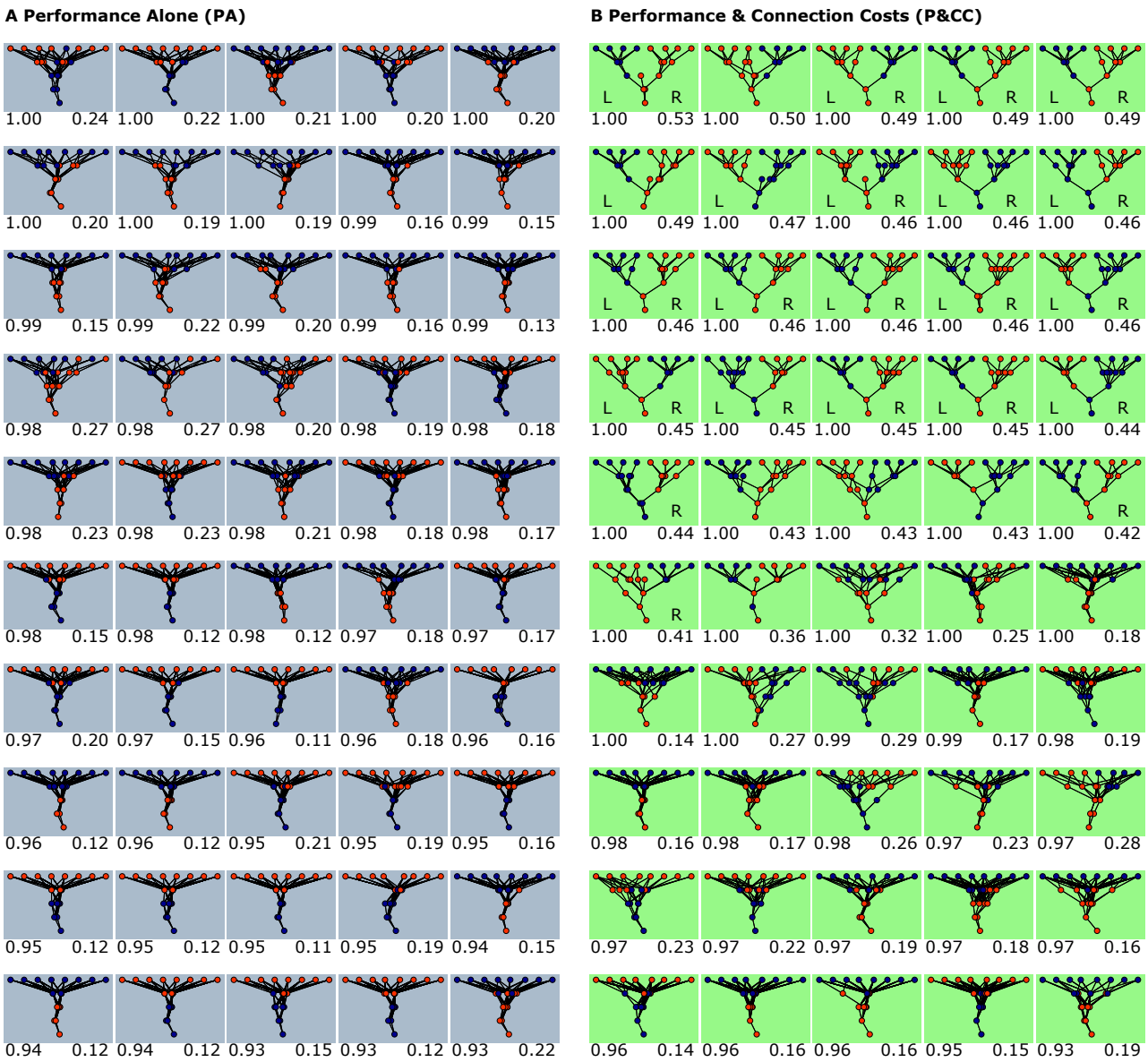


Figure C.1. **Network visualizations and data from the main experiment.** The highest-performing network at the end of each trial is visualized here, sorted first by fitness and second by modularity. For each network the performance (left-justified) and modularity score Q (right-justified) are shown. An "L" or "R" indicates that the network has a single neuron that perfectly solves the left or the right subproblem, respectively. **(A)** Networks from the treatment where the only selection pressure is on the performance of the network. **(B)** Networks from the treatment where there are two selection pressures: one to maximize performance and one to minimize connection costs.

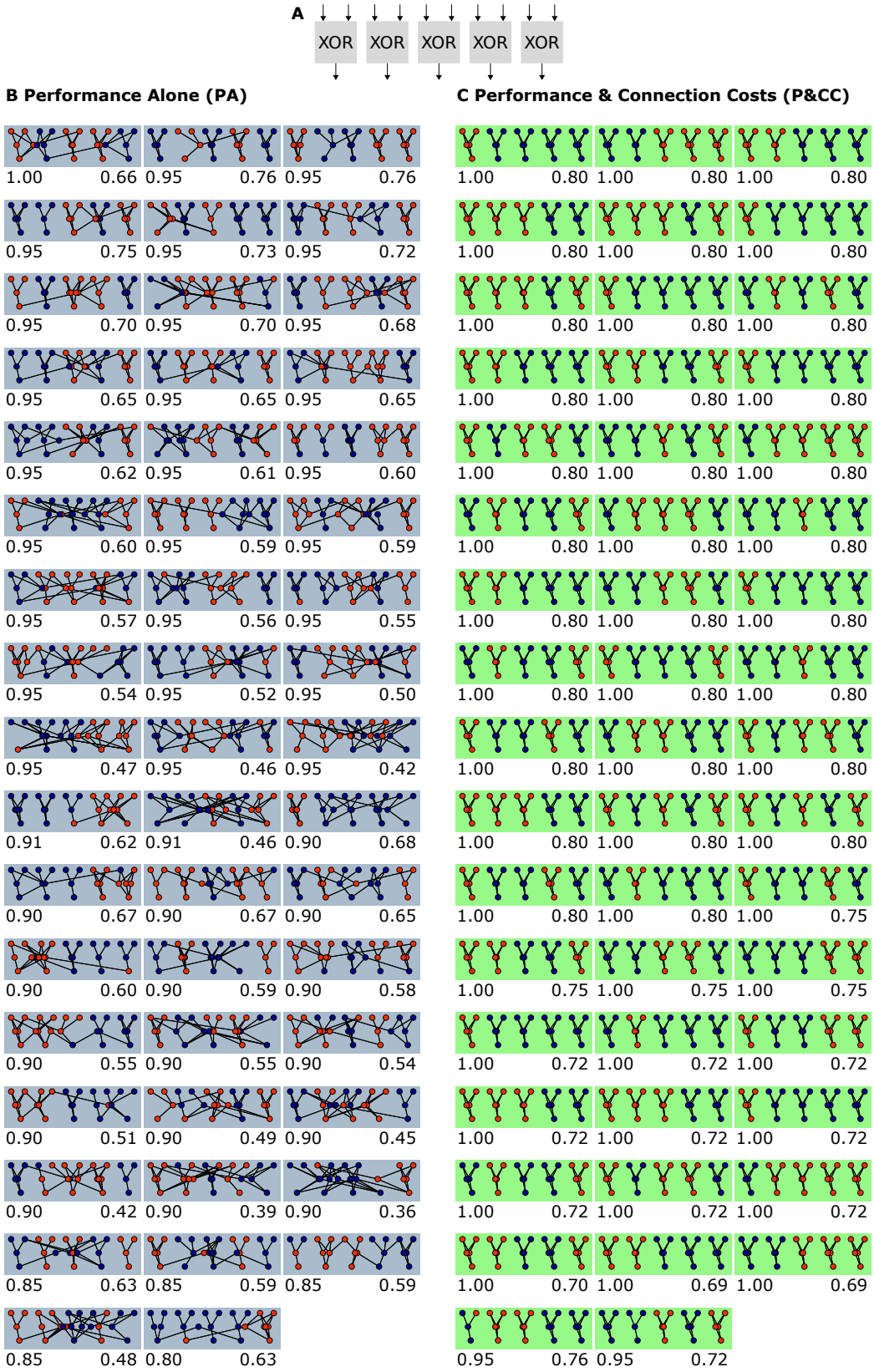


Figure C.2. **Network visualizations and data for the experiment with five decomposable XOR Boolean logic functions.** The highest-performing network at the end of each trial, sorted by fitness then modularity, and its performance (left-justified) and modularity score Q (right-justified). **(A)** The environmental challenge in this experiment. **(B and C)** Networks from the PA and P&CC treatments, respectively.

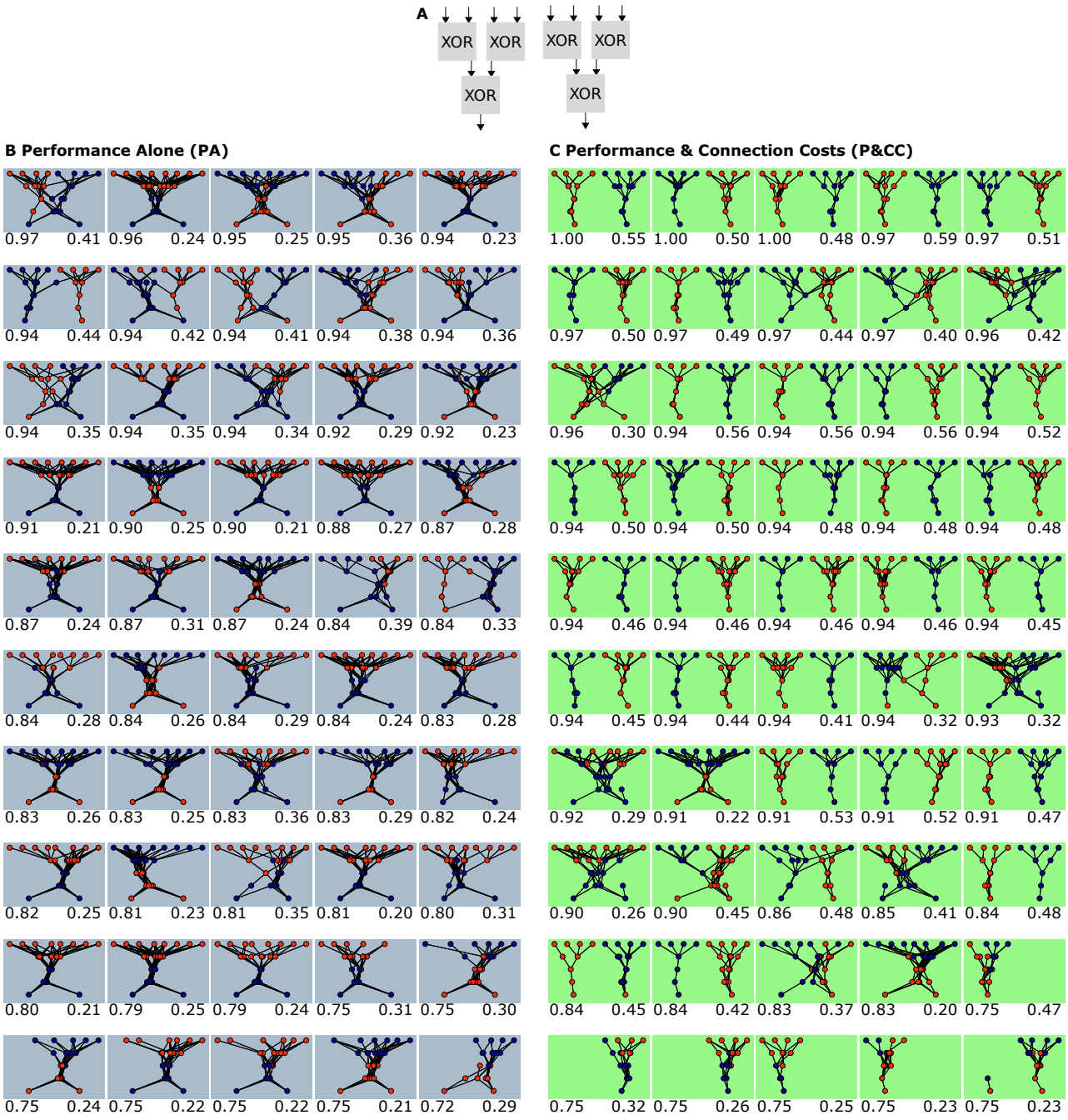


Figure C.3. **Network visualizations and data for the experiment with decomposable, hierarchically nested XOR Boolean logic functions.** The highest-performing network at the end of each trial is visualized here, sorted first by fitness and second by modularity. For each network the performance (left-justified) and modularity score Q (right-justified) are shown. **(A)** A depiction of the environmental challenge in this experiment. **(B)** Networks from the treatment where the only selection pressure is on the performance of the network. **(C)** Networks from the treatment where there are two selection pressures: one to maximize performance and one to minimize connection costs.

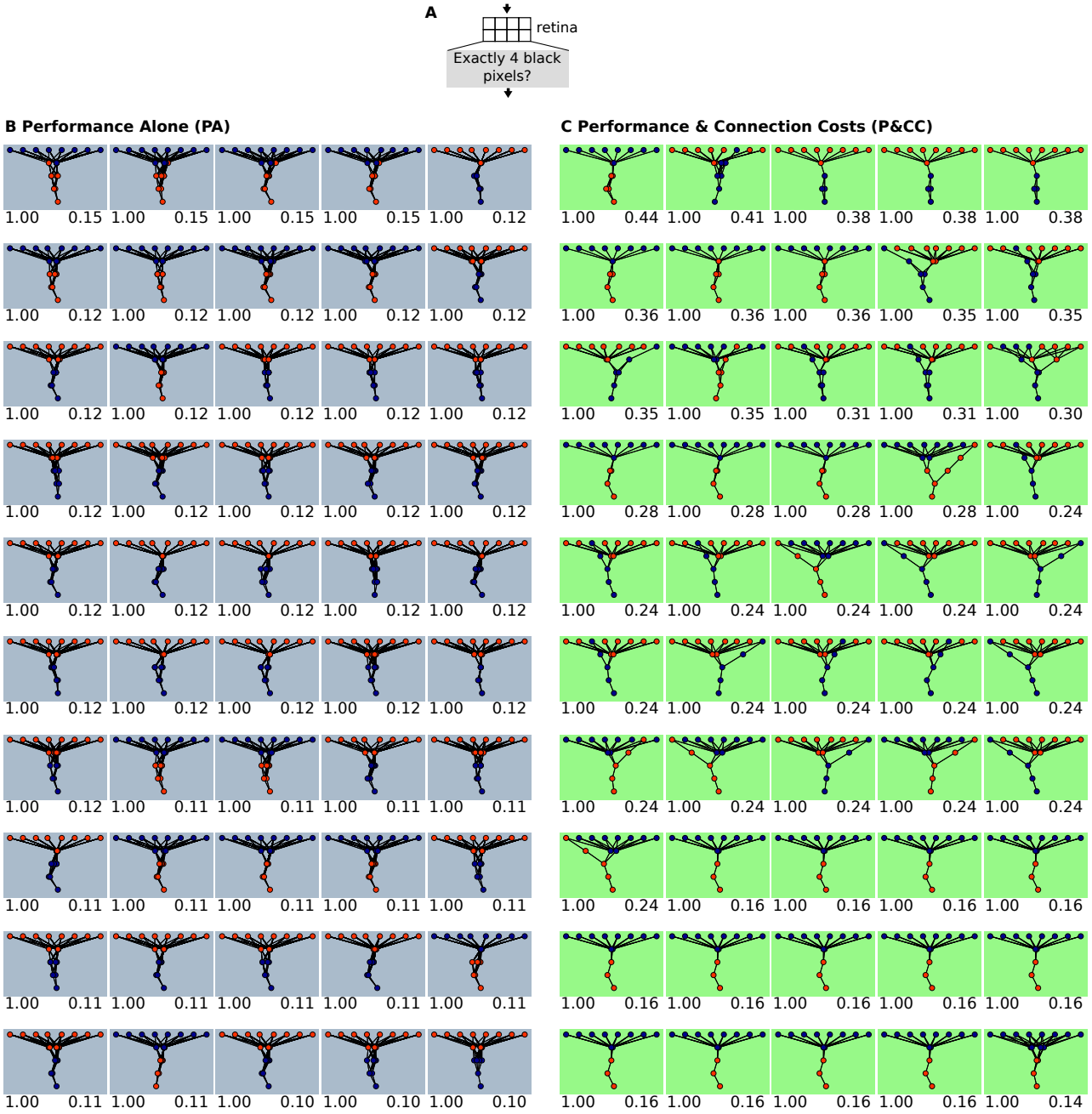


Figure C.4. **Network visualizations and data from the experiment with a non-modular problem. (A)** A non-modular version of the retina problem is created when networks have to answer whether any four pixels are on, which is non-modular because it involves information from anywhere in the retina. **(B, C)** The highest-performing network at the end of each trial is visualized here, sorted first by fitness and second by modularity. For each network the performance (left-justified) and modularity score Q (right-justified) are shown. (B) shows networks from the treatment where the only selection pressure is on the performance of the network. (C) shows networks from the treatment where there are two selection pressures: one to maximize performance and one to minimize connection costs.

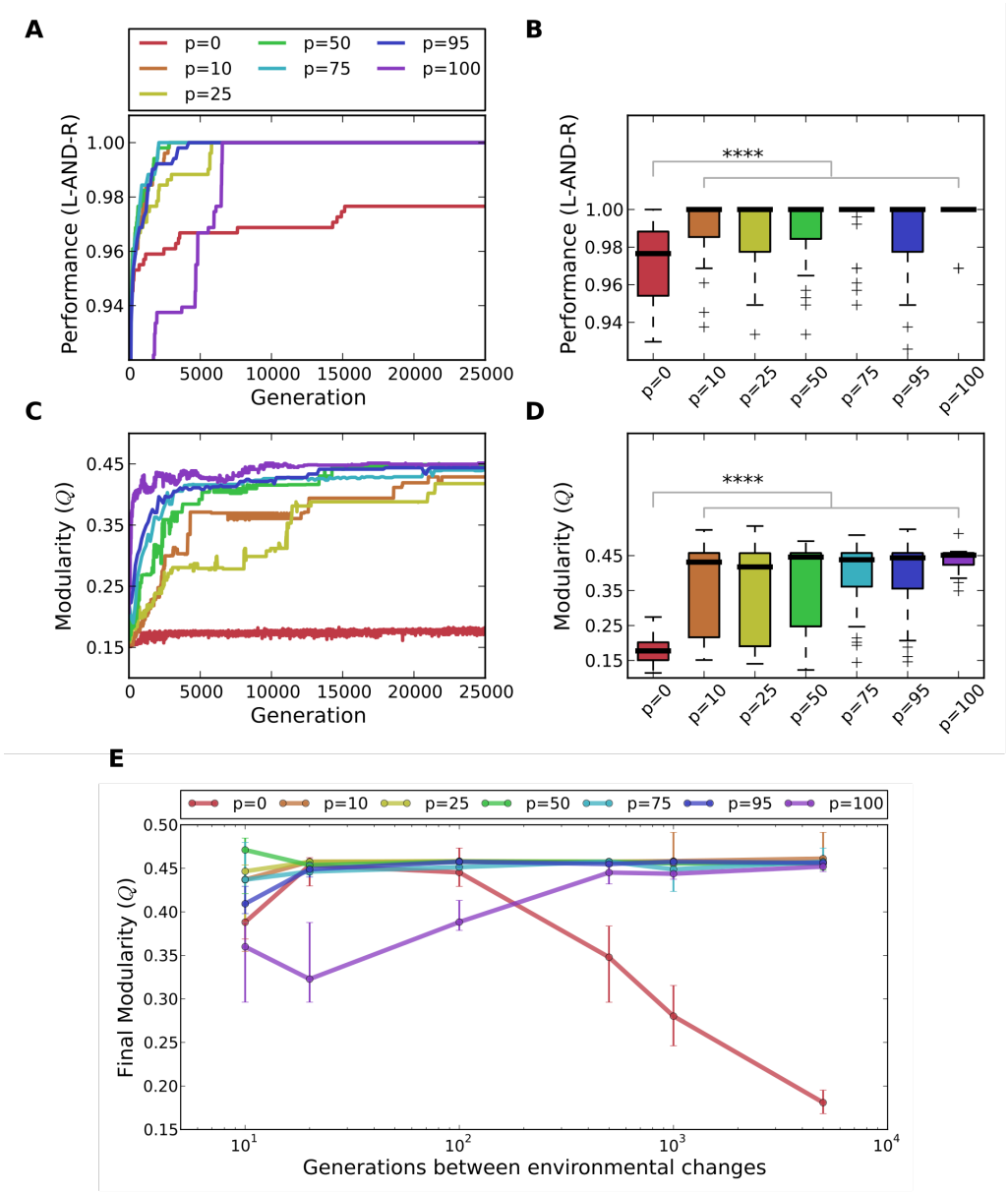


Figure C.5. **Modularity and performance results are robust to variation in the strength of selection to minimize connection costs.** **(A)** Median performance per generation of the highest-performing network, which is improved by adding a selection pressure to minimize connection costs. The value of p indicates the strength of the pressure (Methods), ranging from 0% (no pressure) through 10% (low pressure) to 100% (high pressure–equal to that for the main performance objective). **(B)** Performance of the highest-performing networks after 25000 generations for different values of p. The differences between PA ($p = 0\%$) and P&CC ($p > 0\%$) are highly significant (four asterisks indicate a p-value < 0.0001). **(C)** Median modularity per generation of the highest-performing network of each trial for different values of p. **(D)** Modularity of the highest-performing networks after 25000 generations, which is significantly higher when there is a selection pressure to reduce connection costs (i.e. when $p > 0\%$). **(E)** A changing environment. The x-axis represents the number of generations between a switch from the L-AND-R environment to the L-OR-R environment or vice-versa. This figure is the same as Fig. 3.4 from the main text, but shows results from additional values of p. The results for P&CC ($p > 0$) are consistent across different values of p provided that the strength of selection on network connection costs is not equal to that of network performance (i.e. for all $p < 100\%$). The $p = 100\%$ line is anomalous because in that case cost is as important as performance, leading to mostly low-performing networks with pathological topologies: this effect is strongest when the environment changes quickly because such changes frequently eliminate the fitness benefits of high performance.

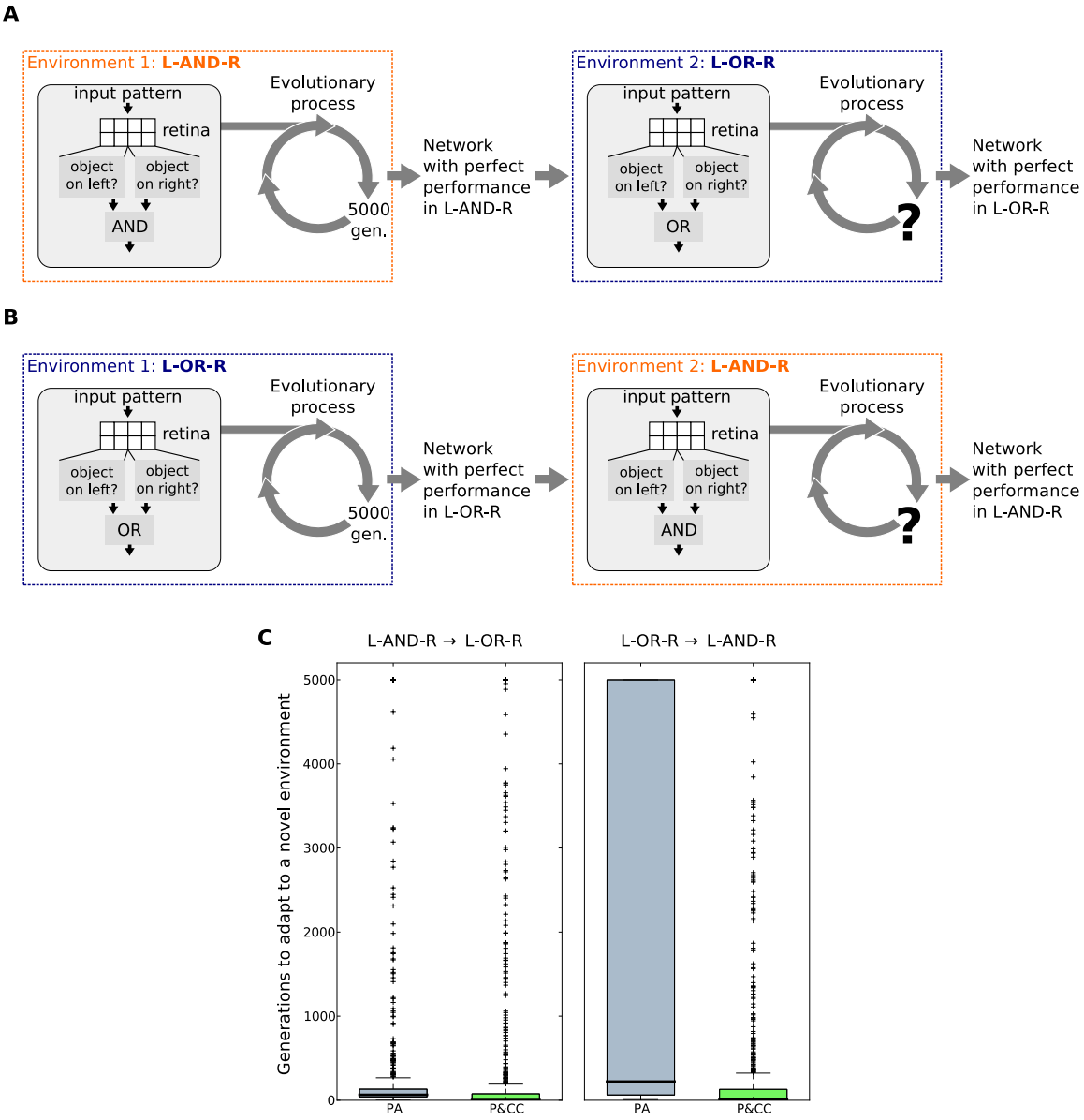


Figure C.6. **Environmental changes between two modular problems with shared subproblems. (A)** In the first evolutionary phase, networks evolve to answer whether a left *and* right object are present (the L-AND-R environment). The problem can be modularly decomposed, since the presence of left and right objects can first be separately detected before answering the larger question of whether they are both present. A left object is considered present if one of the eight left object patterns (shown in Fig. 3.2 in the main text) is exposed to the left side of the retina, and vice-versa for right objects. Networks that could perfectly solve the L-AND-R problem after 5000 generations are then transferred to an environment in which the challenge is to determine if a left *or* right object is present (the L-OR-R environment), which is a different overall problem that shares the subproblems of identifying left and right objects. Evolution continues until networks can perfectly solve the L-OR-R problem or until 5000 generations elapse. **(B)** The same experiment is repeated, but first evolving in the L-OR-R environment and then transferring to the L-AND-R environment. **(C)** A fully zoomed-out plot of Fig. 3.4 from the main text. Trials last 5000 generations or until a network exhibits perfect performance in the new environment. For each treatment, for each of 50 networks that perform perfectly in the first environment, we conduct 50 trials in the second environment, meaning that each column in this figure (and Fig. 3.4 from the main text) represents $50 \times 50 = 2500$ data points.

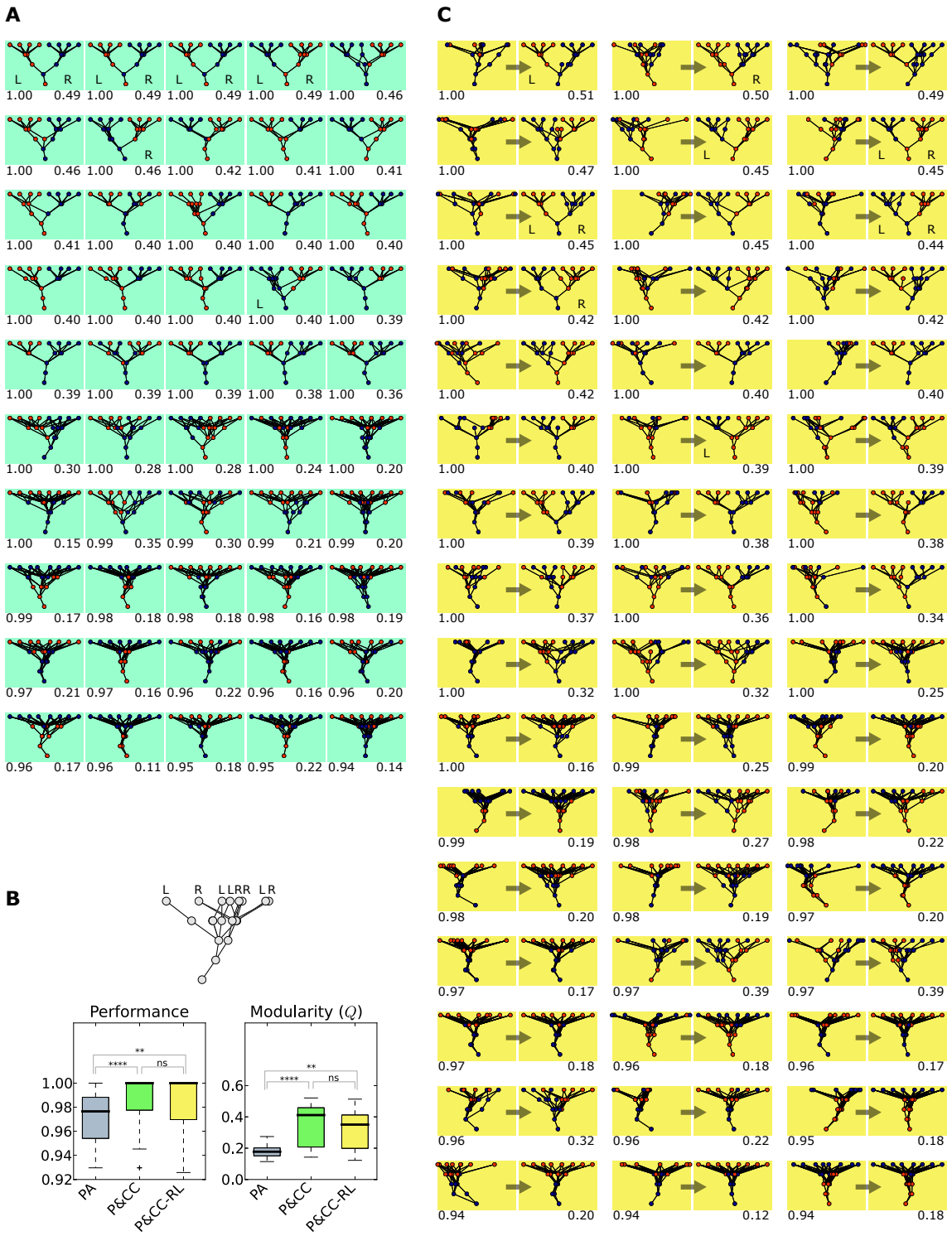


Figure C.7. **(A)** Repeating the main experiment wherein selection is based on performance and connection costs (P&CC), but with connection costs solely measured as the number of connections (P&CC-NC), produces networks that look qualitatively similar to those produced with the default P&CC cost function (compare the networks visualized here to those in Fig. SC.1). The modularity and performance scores are also qualitatively similar and not significantly different (modularity $Q = 0.36[0.22, 0.4]$ vs. default cost function $Q = 0.42[0.25, 0.45], p = 0.15$; performance $= 1.0[0.99, 1.0]$ vs. default cost function performance of $1.0[1.0, 1.0], p = 1.0$). Like the default cost function, the alternative cost function produced significantly more modularity ($p = 1 \times 10^{-8}$) and higher performance ($p = 1 \times 10^{-5}$) than PA. Also qualitatively similar between these cost functions is the percent of runs that have the inputs related to the left and right subproblems in separate modules after splitting the network once to maximize modularity (alternate: 52%, default: 56%, Fisher's exact test: $p = 0.84$). One minor qualitative difference is the percent of runs that have two neurons that each perfectly solve the left and right subproblems, respectively (alternate: 10%, default: 39%, Fisher's exact test: $p = 8 \times 10^{-9}$). **(B)** Randomizing the geometric location of input coordinates (P&CC-RL) eliminates the left-right geometric decomposition of the L-AND-R retina problem, yet such randomization does not change the result that a connection cost causes significantly higher performance and modularity. **(C)** Displayed are final, evolved P&CC-RL networks, which are shown both with randomized input coordinates (left) and with the left and right inputs in order to visualize problem decomposition (right). Despite the randomization of the input locations, many of the evolved modules still functionally decompose the left and right subproblems. Four asterisks indicate a value of $p < 0.0001$ and *ns* indicates no significant difference.

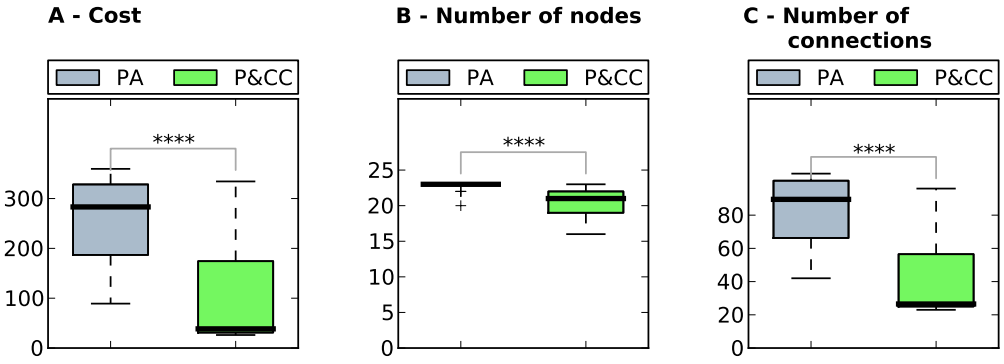


Figure C.8. **Networks evolved with a connection cost have lower costs (shorter lengths), fewer nodes, and fewer connections. (A)** The summed length of all connections is smaller for networks evolved in the P&CC regime than in the PA regime. The P&CC networks also have fewer nodes **(B)** and connections **(C)**. Three asterisks indicate a value of $p < 0.001$ and four indicate a value of $p < 0.0001$.

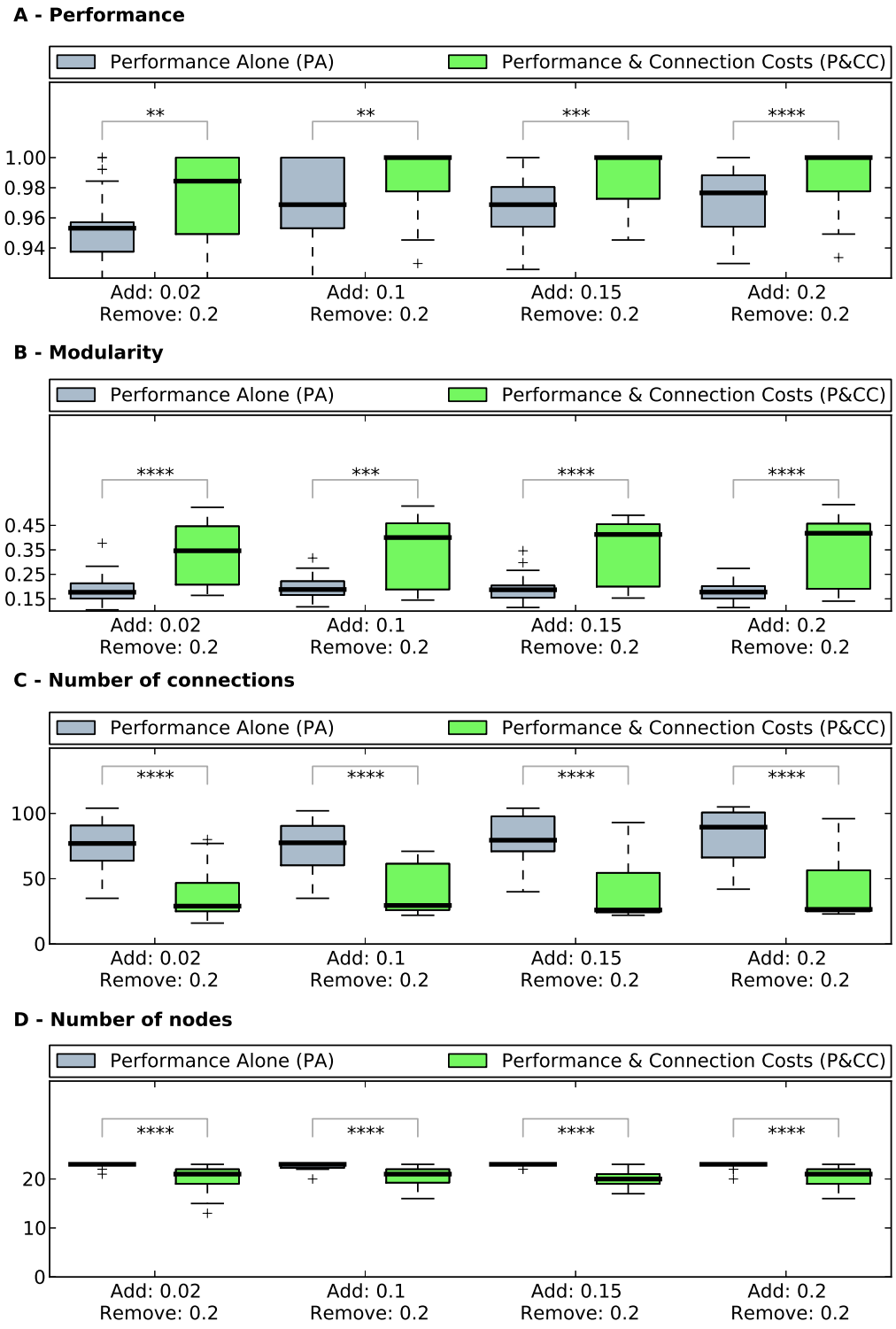


Figure C.9. Biasing the mutation rate towards having more remove-connection mutations than add-connection mutations does not decrease the number of connections, increase the modularity, or increase the performance of PA treatments. Setting the remove-connection mutation rate to be up to an order of magnitude higher than the add-connection mutation rate did not qualitatively change the results. The PA treatment with default values of 0.2 for the remove-connection and add-connection mutation rates did not have statistically different levels of modularity or performance from PA treatments with the same remove-connection mutation rate of 0.2, but a lower add-connection mutation rate of 0.15, 0.10, or 0.02 ($p > 0.05$). The P&CC treatment had significantly fewer connections and nodes, and significantly higher performance and modularity scores, than every PA treatment, irrespective its mutation rate bias. Two, three, and four asterisks indicate $p < 0.01, 0.001$, and 0.0001, respectively.

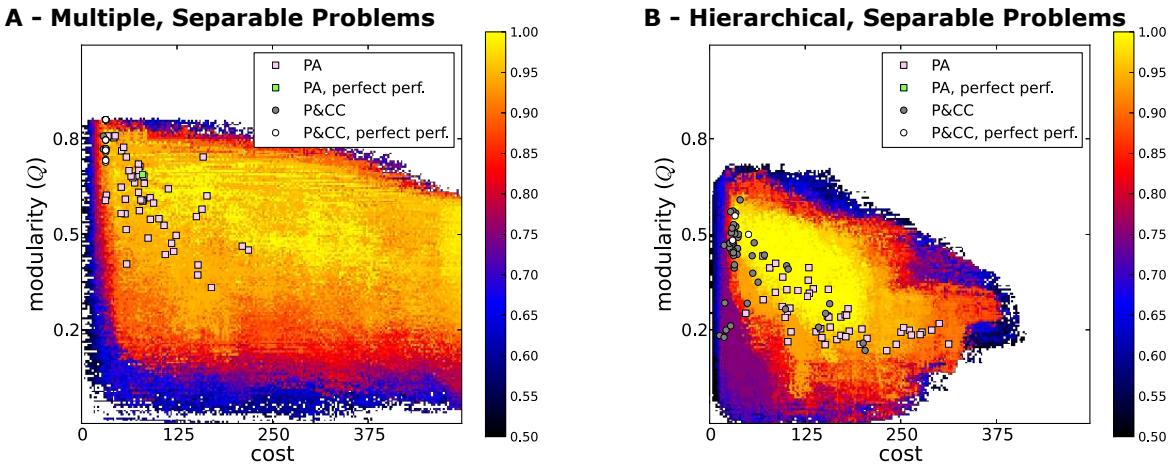


Figure C.10. **The highest-performing networks found for each combination of modularity and length for (A) the problem with five separable XOR functions and (B) the hierarchical XOR problem.** To understand the performance potential of networks with different combinations of modularity levels and summed connection lengths we invented the Multi-Objective Landscape Exploration (MOLE) algorithm, which is described in Methods. Colors indicate the highest-performing network found at that point in the modularity vs. cost space, with yellow representing perfect performance. The best-performing network at the end of each of the 50 PA and P&CC runs are overlaid on the map. Networks with perfect performance exist throughout the space, which helps explain why modularity does not evolve when there is selection based on performance alone. There is also an inverse correlation between length and modularity for high-performing networks: The lowest cost networks–those with the shortest summed lengths–that are high-performing are modular.

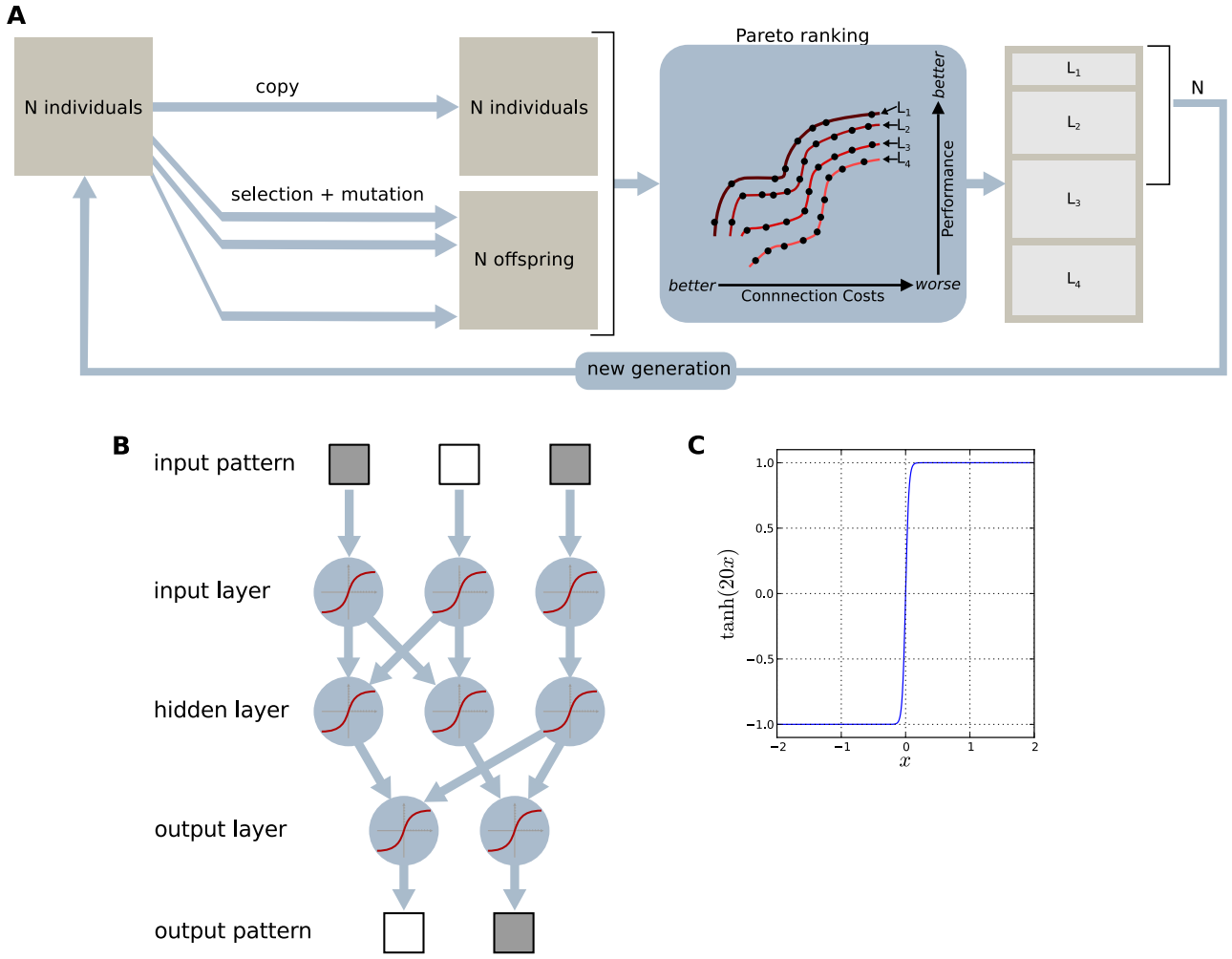


Figure C.11. **Overview of the model**. **(A)** The multi-objective evolutionary algorithm (NSGA-II). Starting with a population of N randomly generated individuals, an offspring population of N new individuals is generated using the best individuals of the current population. The union of the offspring and the current population is then ranked according to the stochastic Pareto dominance (explained in Methods, here represented by having organisms in different ranks connected by lines labeled L_1, L_2, etc.) and the best N individuals form the next generation. **(B)** An example network model. Networks can model many different biological processes, such as genetic regulatory networks, neural networks, metabolic networks, and protein interaction networks. Information enters the network when it is sensed as an input pattern. Nodes, which represent components of the network (e.g. neurons or genes), respond to such information and can activate or inhibit other components of the network to varying degrees. The strength of interactions between two nodes is represented by the weight of the connection between them, which is a scalar value, and whether the interaction is excitatory or inhibitory depends on whether the weight is positive or negative. In this figure all non-zero weights are represented by an arrow. The signal entering each node is passed through a transfer function to determine the output for that node. That output then travels through each of the node's outgoing connections and, after being scaled by the weight of that outgoing connection, serves as a component of the incoming signal for the node at the end of that connection. Eventually an output pattern is produced, which for a neural network could be muscle commands or for a genetic regulatory network could be proteins. **(C)** The transfer function for each node. The sum of the incoming signal (x) for a node is multiplied by 20 before being passed through the transfer function $tanh(x)$. Multiplying by 20 makes the transition steep and similar to a step function. The $tanh(x)$ function ensures that the output is in the range $[-1, 1]$.

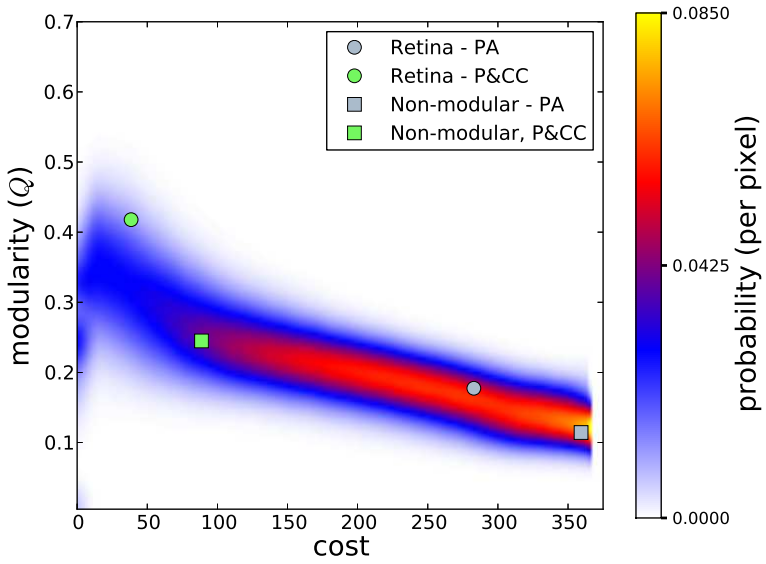


Figure C.12. **Randomly generated networks reveal an inverse correlation between modularity and cost, irrespective of network performance.** For each of the numbers in the range [0,100] we generated 4000 networks with that many connections by randomly adding an allowable connection according to the topology of the retina problem (Methods: Network Model Details). The modularity and cost of those networks are plotted here after being smoothed with Gaussian kernel density estimation and normalized per cost value (the plot looks qualitatively similar if the number of connections, instead of cost, is plotted on the x-axis). This plot reveals a default increase in modularity as network connection cost is lowered prior to the application of selection for performance. It thus suggests that selection for low-cost networks will tend to produce more modular networks unless selection for performance overrides this default tendency. This result helps explain why network modularity increased in the presence of a connection cost even on a non-modular problem (Figure 3.5a). The modularity produced by randomly generating networks thus provides a baseline level of modularity for a given network connection cost that selection for performance further affects. Interestingly, on the modularly decomposable retina problem, the evolved levels of modularity (circles) were higher than the average of randomly-generated networks with the same respective costs, and on the non-modular retina problem, the evolved modularity levels (squares) were slightly lower than observed in randomly generated networks with the same respective costs. The circles and squares show the median modularity and cost values for the four treatments. It is important to remember that additional pressures can produce modularity values that differ from those produced by randomly generated networks. The MOLE algorithm that generated Fig. 3.3, for example, explicitly searches for networks throughout the range of cost and modularity values and found many networks with combinations of these values that are rarely or never produced via random network generation.

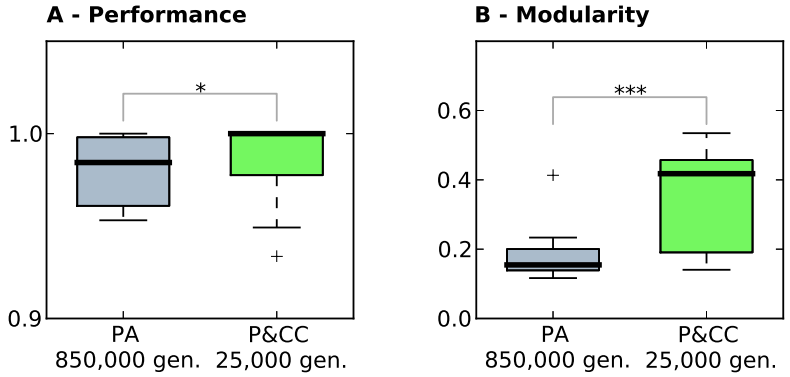


Figure C.13. **Extending the PA treatment to 850,000 generations does not eliminate the significant difference in modularity or performance compared to the P&CC treatment at 25,000 generations.** To see whether the PA treatment would catch up in modularity or performance over a longer evolutionary period, we extended 10 runs of the PA treatment to 850,000 generations, which is 34 times the number of generations in the original experiment. The additional evolutionary time did not significantly improve performance (p = 0.29) or eliminate the statistical difference in performance between PA and P&CC (p = 0.04 using Matlab's Mann-Whitney-Wilcoxon one-tailed rank sum test; median P&CC performance at 25,000 generations = 1.0 [1.0, 1.0]; median PA performance at 850,000 generations was 0.98 [0.96, 1.0]; brackets indicate +/- 95% bootstrapped confidence intervals of the median, calculated by resampling the data 5,000 times). PA modularity also did not increase with the additional generations (p = 0.79), and the levels of modularity in PA runs at 850,000 generations remained far below those of PA (p = 0.0022; P&CC modularity at 25,000 generations $= 0.42[0.25, 0.45]$; PA modularity at 850,000 generations $= 0.15[0.13, 0.22]$). One asterisk indicates a p-value < 0.05 and three indicates a p-value < 0.001).

Supplementary figures for chapter 4

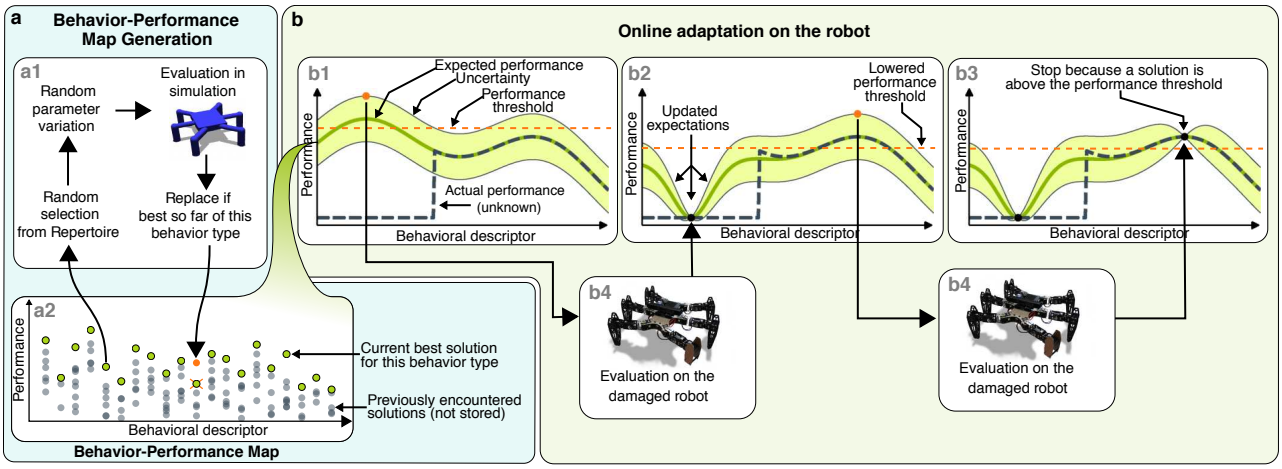
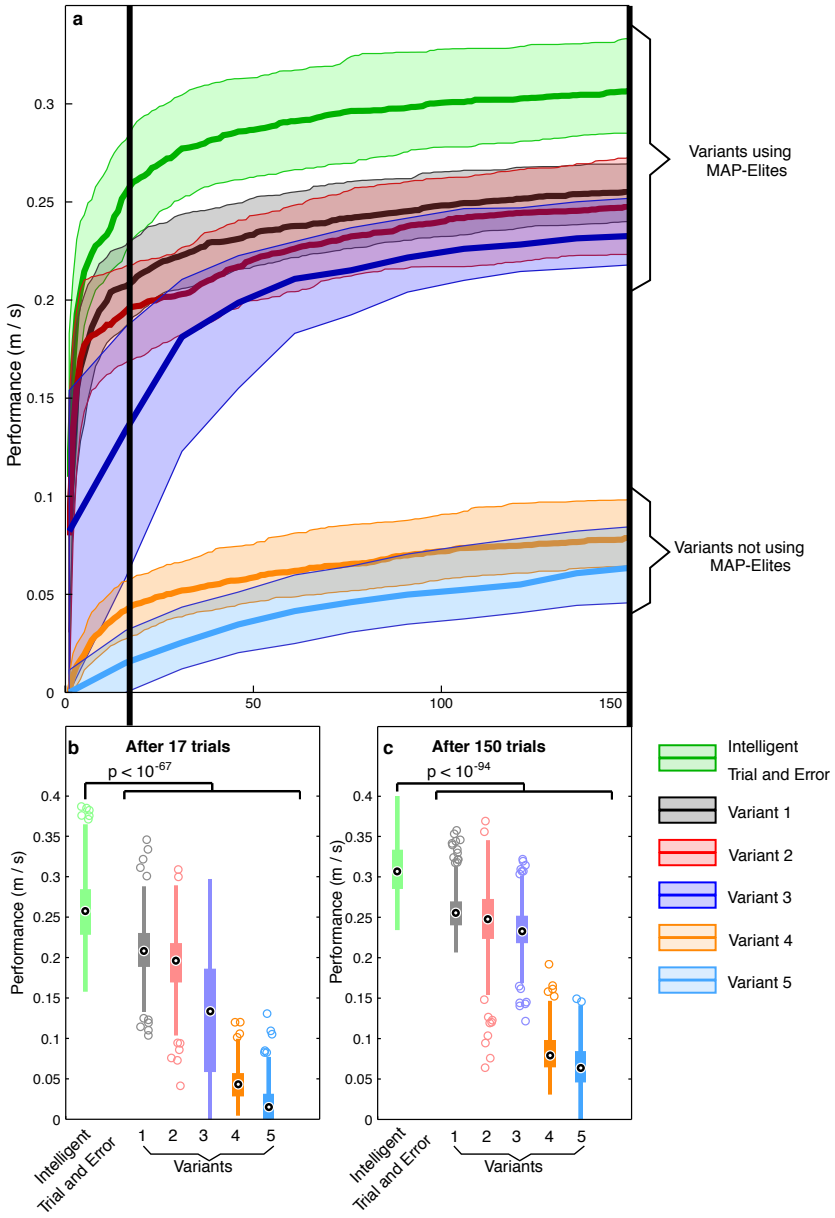


Figure D.1. An overview of the Intelligent Trial and Error Algorithm. (A) Behavior-performance map creation. After being initialized with random controllers, the behavioral map (A2), which stores the highest-performing controller found so far of each behavior type, is improved by repeating the process depicted in (A1) until newly generated controllers are rarely good enough to be added to the map (here, after 20 million evaluations). This step, which occurs in simulation, is computationally expensive, but only needs to be performed once per robot (or robot design) prior to deployment. In our experiments, creating one map involved 20 million iterations of (A1), which lasted roughly two weeks on one multi-core computer (Supplementary Methods, section "Running time"). **(B) Adaptation.** (B1) Each behavior from the behavior-performance map has an expected performance based on its performance in simulation (dark green line) and an estimate of uncertainty regarding this predicted performance (light green band). The actual performance on the now-damaged robot (black dashed line) is unknown to the algorithm. A behavior is selected to try on the damaged robot. This selection is made by balancing exploitation—trying behaviors expected to perform well—and exploration—trying behaviors whose performance is uncertain (Methods, section "acquisition function"). Because all points initially have equal, maximal uncertainty, the first point chosen is that with the highest expected performance. Once this behavior is tested on the physical robot (B4), the performance predicted for that behavior is set to its actual performance, the uncertainty regarding that prediction is lowered, and the predictions for, and uncertainties about, nearby controllers are also updated (according to a Gaussian process model, see Methods, section "kernel function"), the results of which can be seen in (B2). The process is then repeated until performance on the damaged robot is 90% or greater of the maximum expected performance for any behavior (B3). This performance threshold (orange dashed line) lowers as the maximum expected performance (the highest point on the dark green line) is lowered, which occurs when physical tests on the robot underperform expectations, as occurred in (B2).



Variant	Behavior-performance map creation	Priors on performance	Search algorithm	equivalent approach
Intelligent Trial and Error	MAP-Elites	yes	Bayesian Optimization	-
Variant 1	MAP-Elites	none	Random Search	-
Variant 2	MAP-Elites	none	Bayesian Optimization	-
Variant 3	MAP-Elites	none	Policy Gradient	-
Variant 4	none	none	Bayesian Optimization	Lizotte et al. (2007)
Variant 5	none	none	Policy Gradient	Kohl et al. (2004)

Variant	Behavior-performance map creation	Priors on performance	Search algorithm	equivalent approach
Intelligent Trial and Error	MAP-Elites	yes	Bayesian Optimization	-
Variant 1	MAP-Elites	none	random search	-
Variant 2	MAP-Elites	none	Bayesian optimization	-
Variant 3	MAP-Elites	none	policy gradient	-
Variant 4	none	none	Bayesian optimization	Lizotte et al. (2007)
Variant 5	none	none	policy gradient	Kohl et al. (2004)

Figure D.2. The contribution of each subcomponent of the Intelligent Trial and Error Algorithm. (A) Adaptation progress versus the number of robot trials. The walking speed achieved with Intelligent Trial and Error and several "knockout" variants that are missing one of the algorithm's key components. Some variants (4 and 5) correspond to state-of-the-art learning algorithms (policy gradient: Kohl et al. 2004; Bayesian optimization: Lizotte et al. 2007, Tesch et al., 2011, Calandra et al. 2014,). The bold lines represent the medians and the colored areas extend to the 25th and 75th percentiles. **(B, C) Adaptation performance after 17 and 150 trials.** Shown is the the speed of the compensatory behavior discovered by each algorithm after 17 and 150 evaluations on the robot, respectively. For all panels, data are pooled across six damage conditions (the removal of each of the 6 legs in turn). See Supplementary Experiment S2 for methods and analysis.

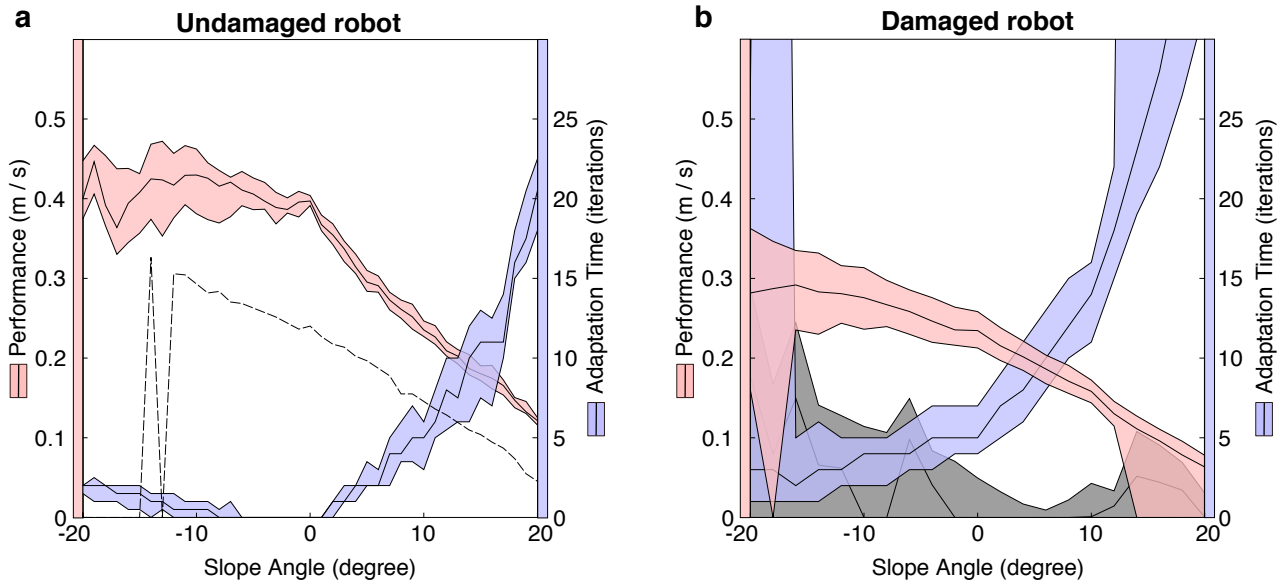


Figure D.3. **The Intelligent Trial and Error Algorithm is robust to environmental changes.** Each plot shows both the performance and required adaptation time for Intelligent Trial and Error when the robot must adapt to walk on terrains of different slopes. **(A) Adaptation performance on an undamaged robot.** On all slope angles, with very few physical trials, the Intelligent Trial and Error Algorithm (pink shaded region) finds fast gaits that outperform the reference gait (black dotted line). **(B) Adaptation performance on a damaged robot.** The robot is damaged by having each of the six legs removed in six different damage scenarios. Data are pooled from all six of these damage conditions. The median compensatory behavior found via Intelligent Trial and Error outperforms the median reference controller on all slope angles. The middle, black lines represent medians, while the colored areas extend to the 25th and 75th percentiles. In (A), the black dashed line is the performance of a classic tripod gait for reference. In (B), the reference gait is tried in all six damage conditions and its median (black line) and 25th and 75th percentiles (black colored area) are shown. See Supplementary Experiment S3 for methods and analysis.

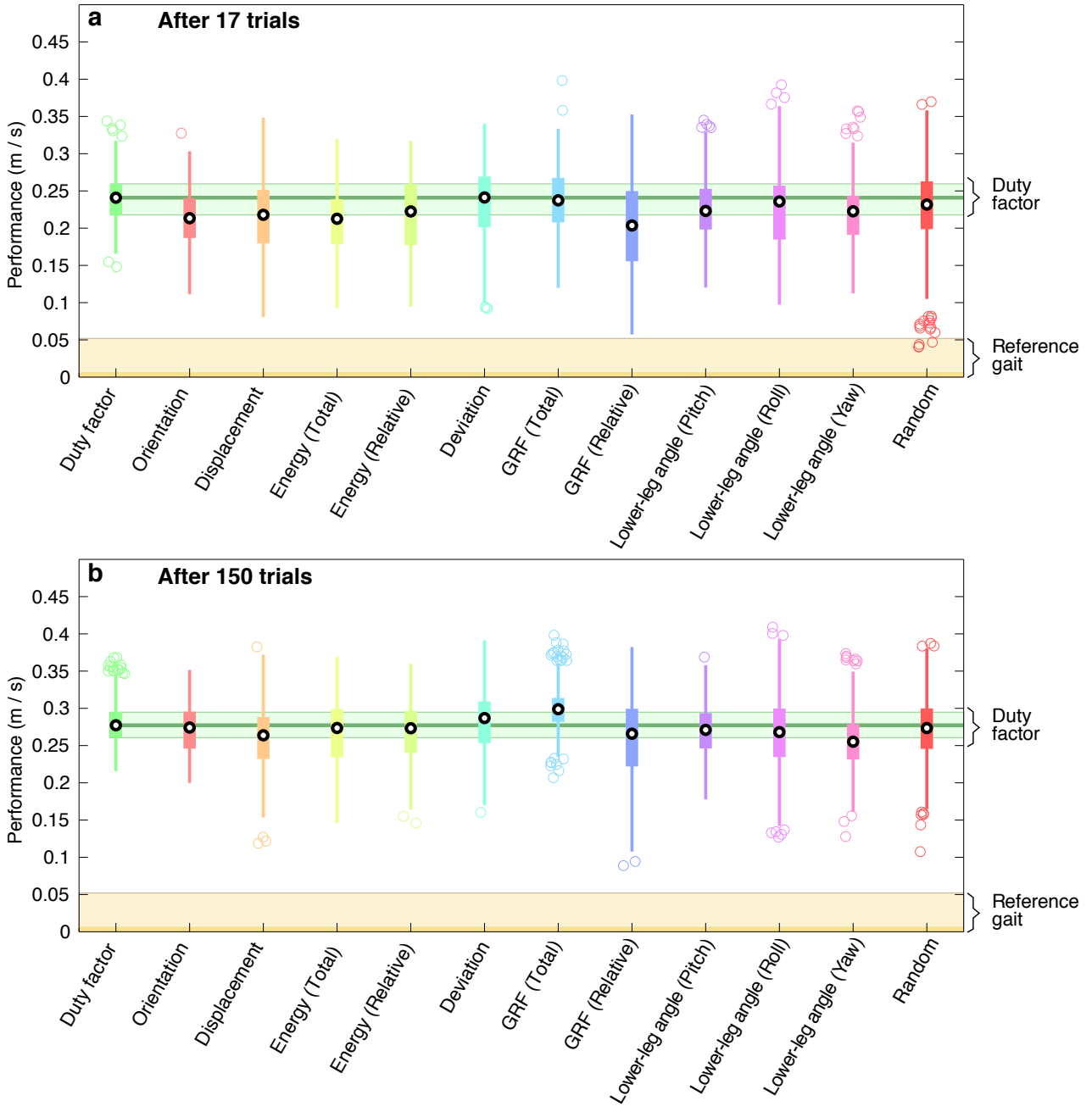


Figure D.4. **The Intelligent Trial and Error Algorithm is largely robust to alternate choices of behavior descriptors. (A, B)** The speed of the compensatory behavior discovered by Intelligent Trial and Error for various choices of behavior descriptors. Performance is plotted after 17 and 150 evaluations in panels A and B, respectively. Experiments were performed on a simulated, damaged hexapod. The damaged robot has each of its six legs removed in six different damage scenarios. Data are pooled across all six damage conditions. As described in Supplementary Experiment S5, the evaluated behavior descriptors characterize the following: (i) Time each leg is in contact with the ground (**Duty factor**); (ii) Orientation of the robot frame (**Orientation**); (iii) Instantaneous velocity of the robot (**Displacement**); (iv) Energy expended by the robot in walking (**Energy (Total)**, **Energy (Relative)**); (v) Deviation from a straight line (**Deviation**); (vi) Ground reaction force on each leg (**GRF (Total)**, **GRF (Relative)**); (vii) The angle of each leg when it touches the ground (**Lower-leg angle (Pitch)**, **Lower-leg angle (Roll)**, **Lower-leg angle (Yaw)**); and (viii) A random selection without replacement from subcomponents of all the available behavior descriptors (i-vii) (**Random**). For the hand-designed reference gait (yellow) and the compensatory gaits found by the default duty factor behavior descriptor (green), the bold lines represent the medians and the colored areas extend to the 25th and 75th percentiles of the data. For the other treatments, including the duty factor treatment, black circles represent the median, the colored area extends to the 25th and 75th percentiles of the data, and the colored circles are outliers. See Supplementary Experiment S5 for methods and analysis.

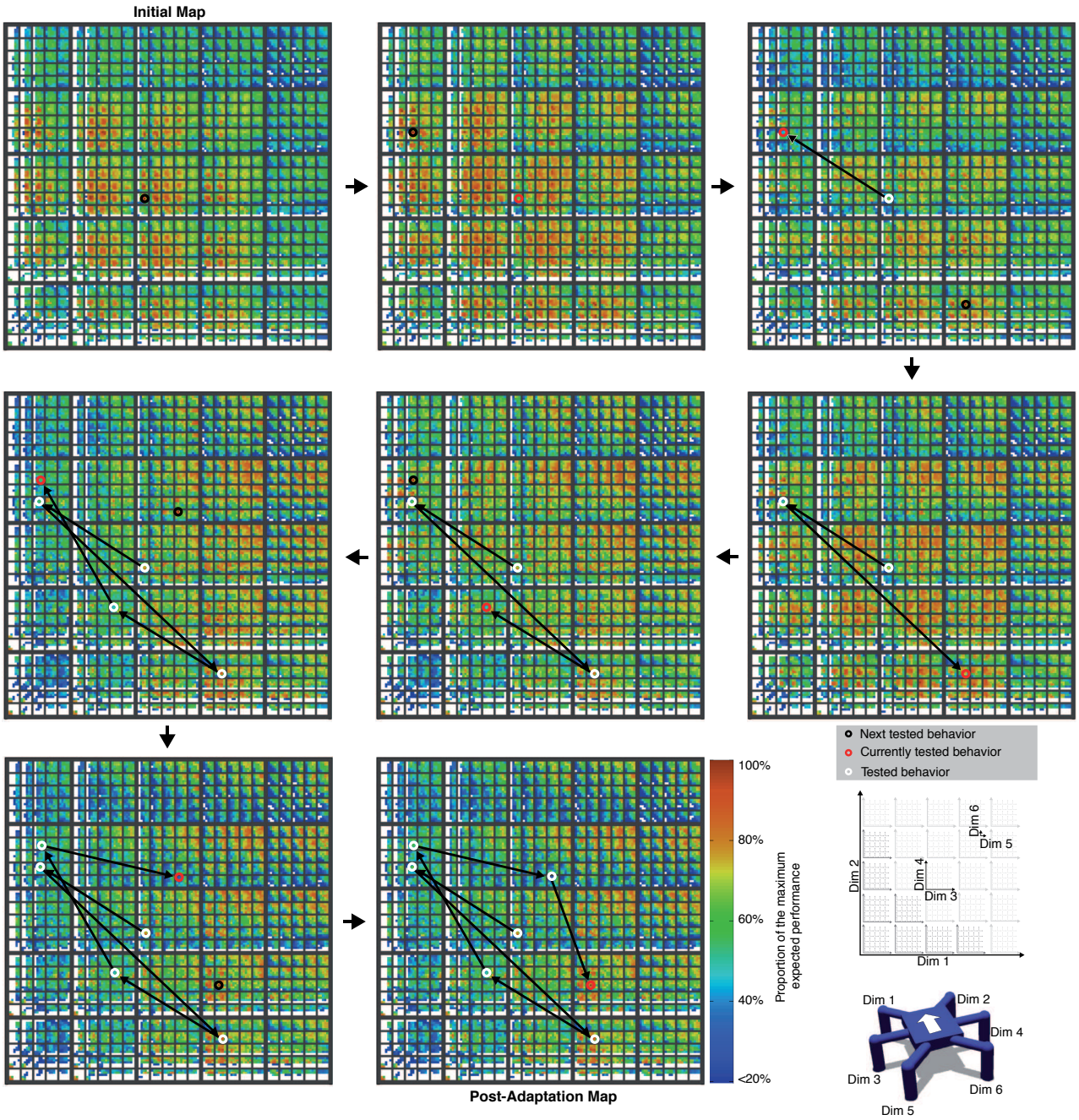


Figure D.5. **How the behavior performance map is explored to discover a compensatory behavior (normalized each iteration to highlight the *range* of remaining performance predictions).** Colors represent the performance prediction for each point in the map relative to the highest performing prediction in the map at that step of the process. A black circle indicates the next behavior to be tested on the physical robot. A red circle indicates the behavior that was just tested (note that the performance predictions surrounding it have changed versus the previous panel). Arrows reveal the order that points have been explored. The red circle in the last map is the final, selected, compensatory behavior. In this scenario, the robot loses leg number 3. The six dimensional space is visualized according to the inset legend.

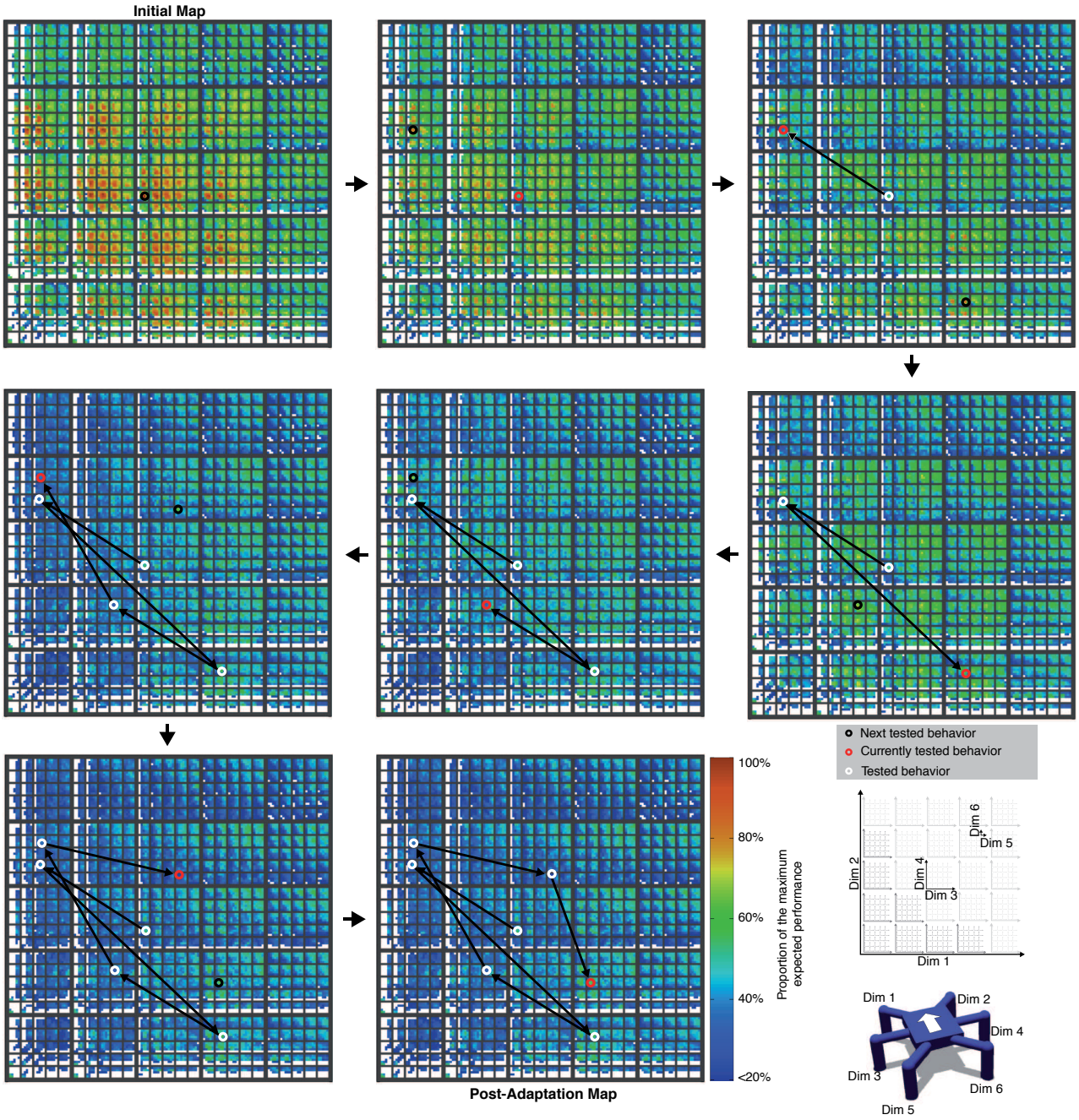


Figure D.6. How the behavior performance map is explored to discover a compensatory behavior (non-normalized to highlight that performance predictions decrease as it is discovered that predictions from the simulated, undamaged robot do not work well on the damaged robot). Colors represent the performance prediction for each point in the map relative to the highest performing prediction in the *first* map. A black circle indicates the next behavior to be tested on the physical robot. A red circle indicates the behavior that was just tested (note that the performance predictions surrounding it have changed versus the previous panel). Arrows reveal the order that points have been explored. The red circle in the last map in the sequence is the final, selected, compensatory behavior. In this scenario, the robot loses leg number 3. The six dimensional space is visualized according to the inset legend. The data visualized in this figure are identical to those in the previous figure: the difference is simply whether the data are renormalized for each new map in the sequence.

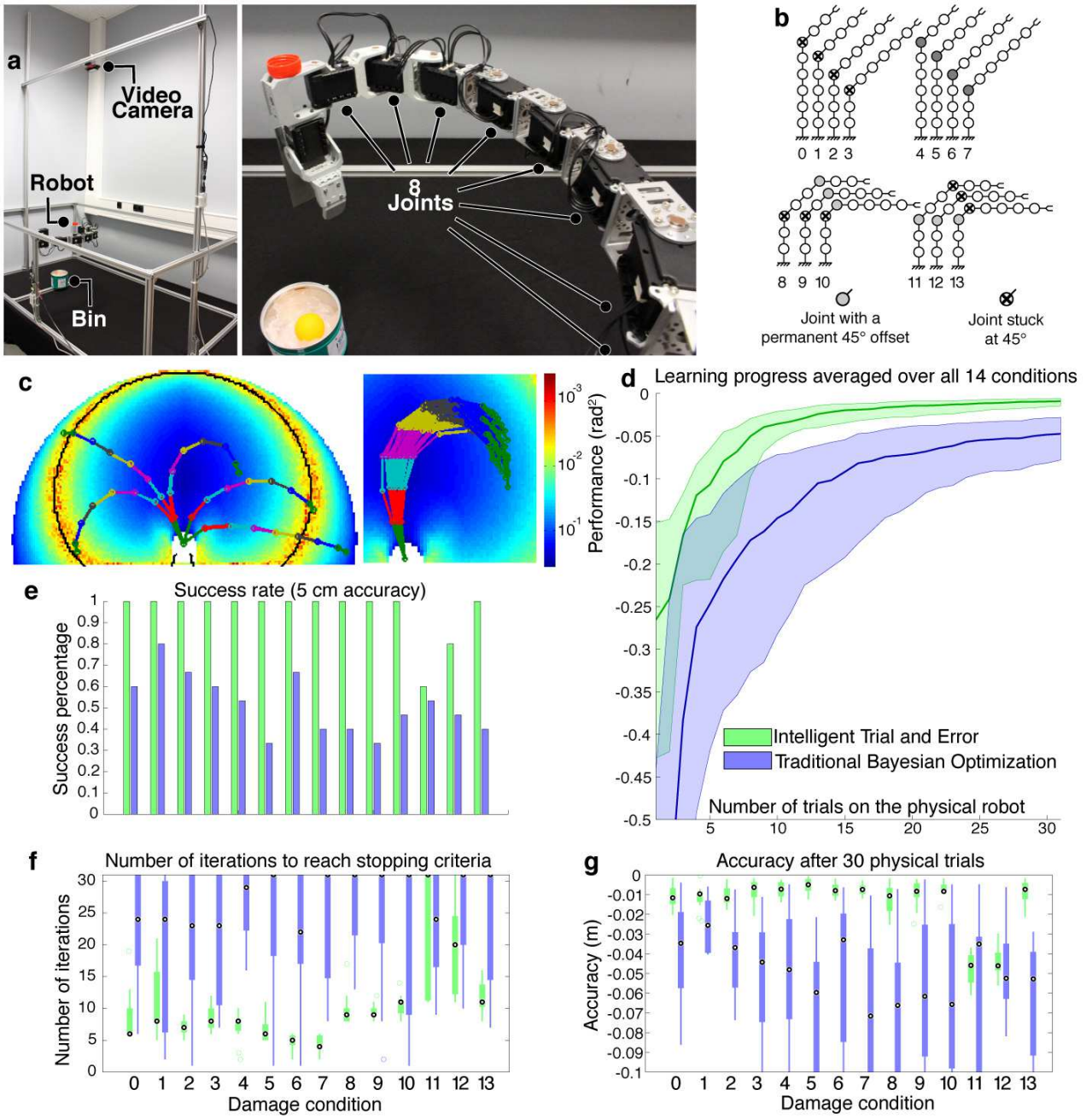


Figure D.7. **Intelligent Trial and Error works on a completely different type of robot: supplementary data from the robotic arm experiment. (A) The robotic arm experimental setup. (B) Tested damage conditions**. **(C) Example of behavior performance maps (colormaps) and behaviors (overlaid arm configurations) obtained with MAP-Elites.** Left: A typical behavior-performance map produced by MAP-Elites with 5 example behaviors, where a behavior is described by the angle of each of the 8 joints. The color of each point is a function of its performance, which is defined as having low variance in the joint angles (i.e. a zigzag arm is lower performing than a straighter arm that reaches the same point). Right: Neighboring points in the map tend to have similar behaviors, thanks to the performance function, which would penalize more jagged ways of reaching those points. That neighbors have similar behaviors justifies updating predictions about the performance of nearby behaviors after a testing a single behavior on the real (damaged) robot. **(D) Performance vs. trial number for Intelligent Trial and Error and traditional Bayesian optimization.** The experiment was conducted on the physical robot, with 15 independent replications for each of the 14 damage conditions. Performance is pooled from all of these $14 \times 15 = 210$ experiments. **(E) Success for each damage condition.** Shown is the success rate for the 15 replications for each damage condition, defined as the percentage of replicates in which the robot reaches within 5 cm of the bin center. **(F) Trials required to adapt.** Shown is the number of iterations required to reach within 5 cm of the basket center. **(G) Accuracy after 30 physical trials.** Performance after 30 physical trials for each damage condition (with the stopping criterion disabled). See Supplementary Experiment S1 for methods and analysis.

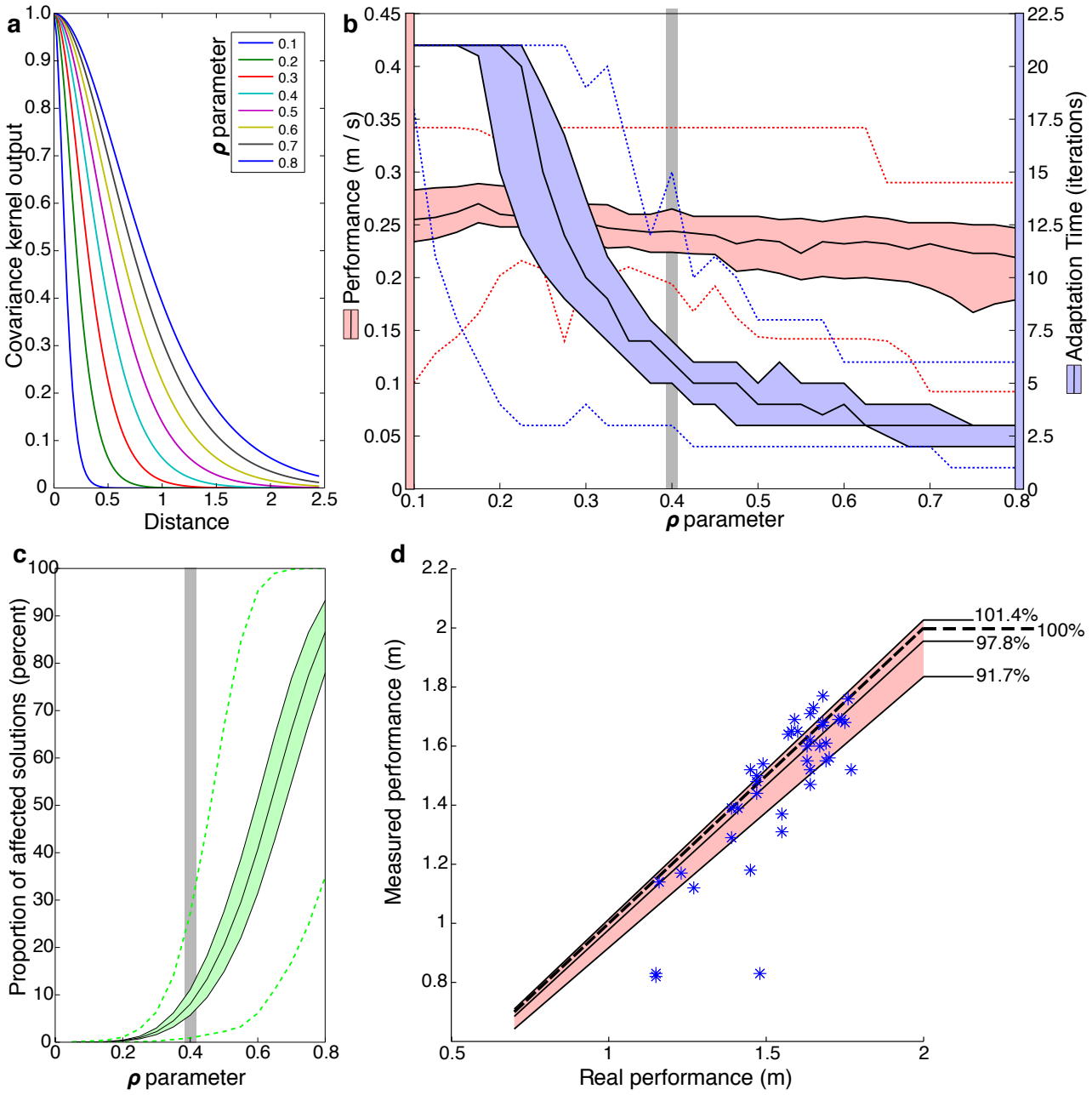


Figure D.8. The effect of changing the algorithm's parameters. (A) The shape of the Matérn kernel function for different values of the ρ parameter. (B) Performance and required adaptation time obtained for different values of ρ. For each ρ value, the R-BOA algorithm was executed in simulation with 8 independently generated behavior-performance maps and for 6 different damage conditions (each case where one leg is missing). **(C) The number of controllers in the map affected by a new observation according to different values of the ρ parameter. (D) The precision of the odometry value**. The distances traveled by the physical robot, as measured manually ("real performance") is compared to the measurements automatically provided by the simultaneous location and mapping (SLAM) algorithm("measured performance"). The dashed black line indicates the hypothetical case where SLAM measurements are error-free and thus are the same as manual measurements. In (B), (C) and (D), the middle, black lines represent medians and the borders of the shaded areas show the 25th and 75th percentiles. The dotted lines are the minimum and maximum values. The gray bars show the ρ value chosen for the hexapod experiments in the main text. See Supplementary Methods for additional details and analysis.

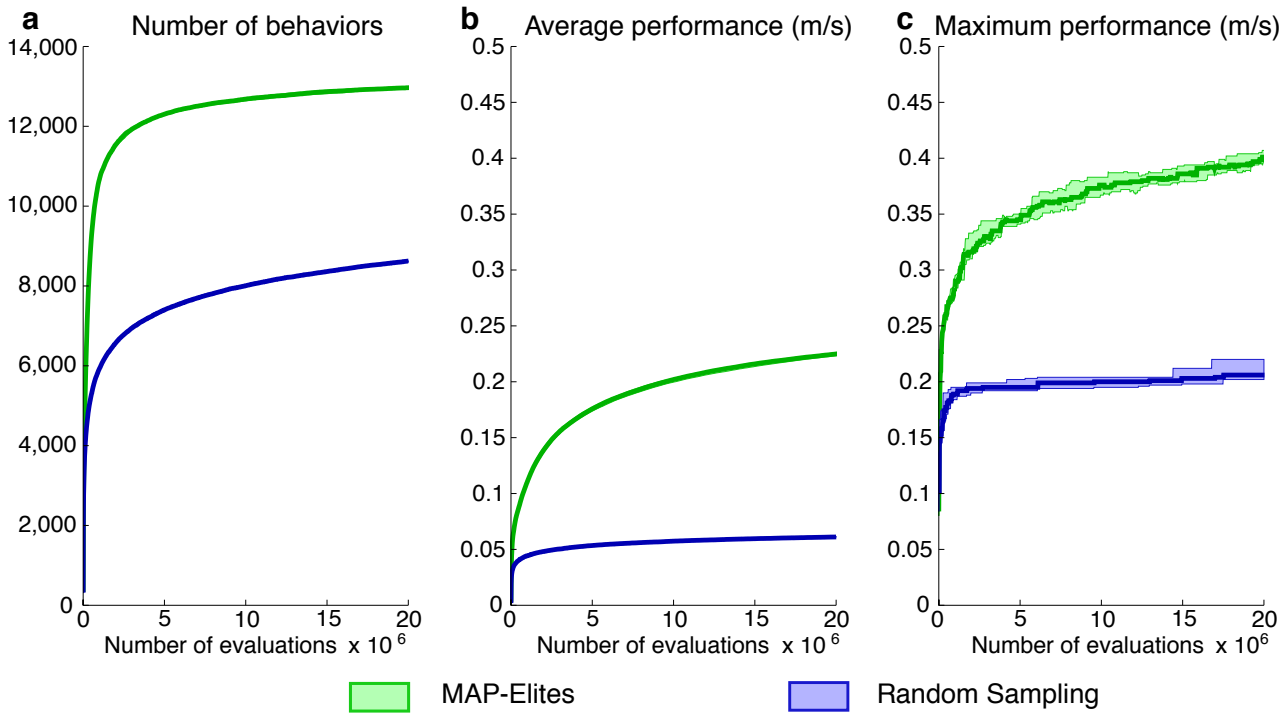


Figure D.9. **Comparing MAP-Elites and random sampling for generating behavior-performance maps. (A) The number of points in the map for which a behavior is discovered. (B) The mean performance of the behaviors in the map. (C) The maximum performance of the behaviors in the map.** For all these figures, the middle lines represent medians over 8 independently generated maps and the shaded regions extend to the 25th and 75th percentiles, even for (A) and (B), where the variance of the distribution is so small that it is difficult to see. See Supplementary Experiment S4 for methods and analysis.

Selection of articles

Here is a selection of five papers that represent the work presented in this HDR. As the scientific content of this HDR is based on these papers, they share most of the text and figures.

Tonelli, P. and Mouret, J.-B. (2013). ***On the Relationships between Generative Encodings, Regularity, and Learning Abilities when Evolving Plastic Artificial Neural Networks.*** **PLoS One. Vol 8 No 11 Pages e79138.**

⇒ chapter 2.

- impact: PLoS One is of the main multi-disciplinary journals (impact factor 2012: 3.730); this paper extends a conference paper that won a best paper award at GECCO 2011.

Clune*, J., Mouret*, J.-B. and Lipson, H. (2013). ***The evolutionary origins of modularity.*** **Proceedings of the Royal Society B, Vol 280, Pages 20122863.** (* J. Clune and J.-B. Mouret contributed equally to this work).

⇒ chapter 3.

- impact: Proceedings of the Royal Society B is one of the main journals in biology (impact factor: 5.683); in one year (publication in January 2013), this paper has been cited 98 times (Google Scholar). This works has been covered by several news outlets (e.g. MIT's Technology review, Fast Company, Carl Zimmer's blog on National Geographics). Following this paper, I was invited to give a talk in several laboratories (e.g. CICRB, Collège de France, Paris, France).

Koos, S., Cully, A. and Mouret, J.-B. (2013). ***Fast Damage Recovery in Robotics with the T-Resilience Algorithm.*** **International Journal of Robotics Research. Vol 32 No 14 Pages 1700-1723.**

⇒ chapter 4.

- impact: IJRR is the top journal in robotics (impact factor: 2.863). This work has been covered by several news outlets (IEEE Spectrum, Yahoo News, Discovery.com Gizmag, Gizmodo, ...). Following this paper, I was invited to talk at an international workshop (PPSN 2014) and to give a keynote for Entrepeneurs' organization (EO) in India.

Cully, A., Clune, J. and D. Tarapore, and Mouret, J.-B. (2015). ***Robots that can adapt like natural animals.*** **Nature. To appear.**

⇒ chapter 4.

- impact: Nature is one of the most prestigious multi-disciplinary journals (impact factor: 42.351). This paper is not published yet.

Tarapore, D. and Mouret, J.-B. (2015). ***Evolvability signatures of generative encodings: beyond standard performance benchmarks.*** **Information Sciences. To appear.**

⇒ chapter 5.

- impact: Information Sciences is a multi-disciplinary journal that covers all the topics related to intelligent systems (impact factor: 3.89).