# Dessin de graphe assisté par un algorithme génétique

Maurin Nadal, Guy Melançon

**MajecSTIC 2012**
**Lille, France, du 29 au 31 octobre 2012**

# Dessin de graphe assisté par un algorithme génétique

Maurin Nadal[1]
Guy Melançon[1]

1 : CNRS, LaBRI, UMR 5800 & INRIA Bordeaux Sud-Ouest, F-33400 Talence, France
Contact : {nadal, melancon}@labri.fr

---

**Résumé**
Les interfaces de visualisation interactive de graphes représentent aujourd'hui une perspective intéressante pour l'analyse de données, et par extension pour les systèmes d'aide à la décision.
Le but de ce projet est d'assister un utilisateur novice dans le cadre du dessin de graphe. Actuellement, une des principales difficultés pour l'utilisateur consiste à choisir l'algorithme de dessin qui conviendra le mieux à son graphe. En effet, il existe un très grand nombre de méthodes possibles et toutes ne sont pas facilement accessibles.
La solution envisagée consiste à fournir automatiquement plusieurs dessins viables d'un même graphe. Ces dessins sont générés par un algorithme par modèle de force (système masse-ressort) modifié afin d'être paramétrable sommet par sommet. Les jeux de paramètres sont fournis par un algorithme génétique.
Cet article présente principalement une preuve de concept de la possibilité d'utiliser un tel processus pour dessiner tout type de graphe, et plus particulièrement des graphes fortement contraints (angles droits ou parallélismes).
Les points abordés sont le modèle masse-ressort utilisé et les modifications qui lui ont été apportées, les caractéristiques principales de l'algorithme génétique mis en œuvre, la métrique de similarité permettant l'évaluation des dessins générés au cours de l'apprentissage et enfin le cas d'application proposé comme preuve de concept.

**Abstract**
Interactive visualization interfaces for graph are an interesting perspective for data analysis, and by extension for decision support system.
The aim of this project is to assist user in drawing graphs. Currently, one of the main difficulties for the user is to choose the best fitted algorithm for his graph. Indeed, there are lots of different algorithms and few of them are easy to use.
The suggested solution is to generate different correct drawings for the same graph. Those drawings are generated by a modified force-directed placement algorithm for which parameters are set vertex by vertex. Parameter sets are given by a genetic algorithm.
This article presents a proof of concept that this modified algorithm and the associated genetic algorithm are able to reproduce highly constrained drawing (parallel edges or right-angled) in a very different way that what force directed placement algorithms do.
The different highlighted points in this article are the mass-spring system and its modification, the genetic algorithm, the similarity method used to evaluate drawing and the proof of concept of the method.

**Mots-clés :** algorithme génétique, contrainte, graphe, modèle de force, similarité

**Keywords:** genetic algorithm, constraint, graph, force-directed, similarity

---

## 1. Introduction

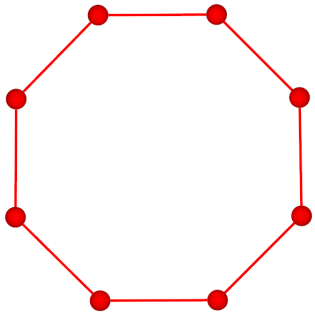
Cet article se place dans le domaine de la visualisation de données, dont la problématique générale consiste à permettre à l'être humain d'exploiter de grandes masses de données efficacement. La production d'une représentation visuelle des informations est pour cela un outil intéressant. En

effet, l'œil humain est très efficace pour identifier rapidement les éléments importants d'une image. Par exemple, la localisation d'un carré bleu au milieu d'un grand nombre de ronds rouges est très rapide. Dans cette optique, l'objectif principal lors de la création d'une représentation graphique consiste à déterminer les éléments importants et à les associer à une caractéristique graphique permettant de les identifier rapidement. Ces caractéristiques peuvent concerner la position, la forme, la taille, la couleur, etc.
Une des difficultés actuelles de la visualisation consiste à permettre à l'utilisateur novice de créer de nouvelles représentations lui permettant d'exploiter ses données. Comme Grammel et al. le présentent dans leur article de 2010 [11], les méthodes actuelles ne permettent pas de dépasser cette difficulté, et cela nuit à l'accessibilité de la visualisation de données auprès du grand public.
Ces travaux concernent un sous-domaine de la visualisation, le dessin de graphe. Un graphe $G$ est constitué d'un ensemble de sommets, noté $V$, associé à un ensemble d'arêtes $E \subset V \times V$. Les sommets représentent des entités et les arêtes une relation entre ces dernières. Par exemple, pour modéliser un réseau social, chaque membre correspond à un sommet et une arête relie deux membres lorsqu'ils sont en relation.
Un des problèmes principaux pour représenter un graphe concerne le positionnement des sommets. Cela revient à trouver une fonction $l : V \to \mathbb{R} \times \mathbb{R}$ pour un dessin en deux dimensions. Cette fonction est appelée layout du graphe. Il est bien évidemment possible d'envisager plusieurs dessins pour un même graphe.
Dans certains cas, il est aisé de trouver un positionnement des sommets qui fournit un dessin intéressant, ie. qui permet à l'utilisateur d'interpréter les données. Par exemple, pour un cycle (boucle où chaque sommet est relié uniquement à son successeur et son prédécesseur), il suffit de positionner les sommets en cercle (voir fig. 1a). De nombreux algorithmes permettent ainsi de dessiner très rapidement certaines classes de graphes.
Toutefois, la plupart des données réelles sont rarement aussi régulières et il n'existe alors pas de telles méthodes pour obtenir un dessin convenable. Une solution permettant de résoudre ce problème est fournie par des algorithmes qui appliquent itérativement de petits déplacements sur chaque sommet jusqu'à l'obtention d'un résultat exploitable. Certains algorithmes basés sur des systèmes masses-ressorts permettent ainsi de dessiner une grande variété de graphes.
Cet article présente l'utilisation d'une de ces méthodes couplée à un algorithme génétique dans le but de fournir à un utilisateur plusieurs dessins corrects et différents d'un même graphe. Il est en effet très difficile de déterminer automatiquement le dessin le plus adapté à un jeu de données. Ainsi, seul l'utilisateur peut faire ce choix à l'aide du questionnement "Est-ce que ce dessin me permet de répondre rapidement aux questions que je me pose ?". Cette approche a été introduite par la méthode SMILE, présentée en 1998 par Bield et al. [2]. L'argument principal consiste à dire qu'en général, l'utilisateur ne sait pas dire à priori quel serait un bon dessin pour son graphe. De ce fait, la seule manière d'y aboutir est de lui proposer des dessins différents pour qu'il puisse les comparer et choisir le meilleur. Cependant, SMILE n'intègre pas d'élément d'apprentissage et d'adaptation à de nouveaux graphes.

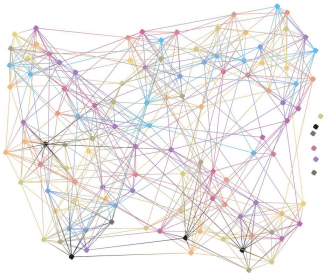
Les principaux obstacles rencontrés dans cette démarche sont :
– La mise au point d'une méthode permettant d'obtenir différents dessins d'un même graphe
– Un moyen efficace d'explorer l'espace des dessins
– Un évaluateur permettant de sélectionner automatiquement les "bons" dessins
Ces points sont abordés dans les parties 2 à 4. Ensuite vient la présentation d'une preuve de concept de cette méthode. La dernière partie concerne les différentes suites qui sont envisagées dans le cadre de ce travail.

## 2. Modification d'un algorithme par modèle de force

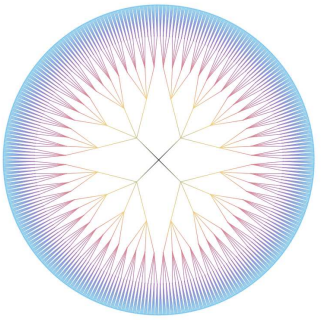
Une des grandes branches actuelles des méthodes de dessins de graphes correspond aux algorithmes par modèle de force. Ces derniers sont basés sur des systèmes masse-ressort pour lesquels un point d'équilibre minimisant une fonction d'énergie est recherché. Les principales actions exercées sur les sommets proviennent d'une force d'attraction qui associe un ressort à chaque arête et d'une force de répulsion entre chaque paire de sommets. Des forces mineures sont aussi régulièrement utilisées, comme par exemple une gravité (tous les sommets sont attirés vers le centre du



(a) Un cycle à 8 sommets



(b) Les rencontres de football entre universités américaines



(c) Un arbre complet de degré 4 et de hauteur 5

dessin), des forces aléatoires ou des effets d'inertie.
Ce type d'algorithme a été introduit par P. Eades en 1984 [7]. Dans cette version, le placement initial des sommets était aléatoire et seules les forces d'attraction et de répulsion étaient utilisées. Plusieurs variations ont vu le jour dans les années qui ont suivi. Quelques-unes ont particulièrement marqué la communauté. Celle de Kamada et Kawai en 1989 [13] a pour principal objectif de rendre proportionnelle la distance euclidienne entre deux sommets dans le dessin par rapport à leur distance dans le graphe. Cela permet de comparer très facilement les longueurs des chemins dans le graphe entre deux paires de sommets. De plus, cette fonction d'énergie produit des dessins esthétiquement agréables pour un large panel de graphes. Une autre version marquante a été celle de Fruchterman et Reingold, publiée en 1991 [9]. Les dessins obtenus par ces méthodes sont communément jugés corrects pour la plupart des graphes. Cela s'explique par la volonté générale d'optimiser une série de métriques esthétiques courantes et qui permettent d'obtenir un bon rendu visuel. Par exemple, il est important de conserver les symétries, car elles sont repérées très rapidement par l'œil humain, ce qui facilite l'identification de motifs.

Toutefois, ces algorithmes présentent certains inconvénients. Tout d'abord, ils demandent à l'utilisateur de spécifier un grand nombre de paramètres souvent abstraits. En général, les valeurs par défaut sont conservées car même s'il est possible d'affiner le comportement de l'algorithme, une modification trop importante diminue fortement la qualité du dessin obtenu. Bien évidemment, il est possible d'obtenir des dessins différents d'un même graphe en utilisant les différents algorithmes disponibles. Toutefois, la plupart sont difficilement accessibles et leur implémentation n'est pas toujours disponible. De ce fait, la majorité des utilisateurs n'utilisent qu'un faible nombre d'algorithmes différents. Et ils se satisfont du dessin fourni par ces derniers sans tenter de l'améliorer. Cette tendance des utilisateurs est très courante actuellement, et est critiquée par Brandes dans un keynote de 2011 [5] où il dénonce la stagnation de l'usage d'algorithmes datant de plusieurs années alors que de nouvelles méthodes bien plus puissantes sont disponibles.

De plus, chaque algorithme traite l'ensemble du graphe de la même manière. Si l'utilisateur souhaite dessiner certaines parties différemment du reste, cela n'est pas possible. Certaines méthodes permettent de dessiner des layouts locaux comme TopoLayout par exemple [1]. Le principe consiste à détecter des motifs connus dans les graphes afin d'utiliser des algorithmes adaptés à ces derniers. Ensuite, les différents dessins obtenus sont agencés efficacement au sein d'un grand graphe. Cette méthode permet un dessin très régulier des motifs détectés et une grande vitesse d'exécution. En effet, les algorithmes spécialisés sont beaucoup plus rapides puisqu'il leur suffit souvent de calculer la position de chaque sommet. L'apport de notre méthode consiste en l'apprentissage de nouveaux motifs ainsi que dans le traitement des graphes dans lesquels aucune régularité ne peut être trouvée.

Dans le cadre de ce projet, c'est aussi un algorithme par modèle de force, GEM [8], qui est utilisé. La modification proposée consiste à paramétrer l'algorithme sommet par sommet avec :
– Le coefficient de la force d'attraction
– Le coefficient de la force de répulsion
– La longueur idéale des arêtes incidentes au sommet

– Le coefficient de la force de gravité
– Le coefficient de la force aléatoire
– La masse associée au sommet

Toutefois, cela entraîne une démultiplication des valeurs à saisir ce qui rend impossible un réglage manuel. Un algorithme génétique est donc utilisé pour générer les différents jeux de paramètres.

**3. L'algorithme génétique**

Un algorithme génétique permet d'explorer un grand espace de manière efficace. Il tente d'obtenir un ou plusieurs génomes permettant de fournir une solution à un problème donné. Un génome correspond à une série de données numériques ou textuelles. L'algorithme travaille sur une population de génomes (le nombre peut varier de quelques dizaines à plusieurs milliers en fonction de la complexité du problème). Le principe consiste à évaluer toute la population pour attribuer un score à chaque individu. Ensuite, la génération suivante est générée à l'aide de croisements et de mutations effectués sur les génomes ayant obtenu les meilleurs scores. Ce processus est itéré jusqu'à l'obtention d'un résultat satisfaisant. Dans notre cas, un génome correspond à un jeu de paramètres pour un sommet.

Des algorithmes génétiques ont déjà été utilisés dans le cadre du dessin de graphes. L'approche envisagée consiste à utiliser un génome qui contient vecteur de $|V|$ couples de valeur $(x, y)$ correspondant à la position de chaque sommet. Les différences se situent principalement au niveau de la fonction d'évaluation ainsi que sur les opérateurs génétiques (mutation et croisement) qui sont souvent très spécialisés afin de s'adapter au mieux au dessin de graphe. Les premiers résultats datent du début des années 90 avec les articles de Groves et al. [12], Kosak et al. [14] et Markus [15]. Plusieurs travaux ont depuis repris le même type de méthode. Cependant, ce type de génome pose plusieurs problèmes. Au niveau du croisement, lorsque deux layouts sont mélangés (la moitié des positions vient d'un premier layout et l'autre moitié d'un deuxième), les propriétés de chacun des layouts ne sont que rarement conservées. De plus, les calculs réalisés sur un graphe ne sont pas réutilisables pour un autre graphe. De ce fait, il faut toujours repartir de 0 lorsqu'un nouveau graphe doit être dessiné. La solution proposée permet d'éviter ces deux problèmes.

Tout d'abord, le fait de fournir des paramètres de dessins plutôt qu'une position pour chaque sommet permet d'abstraire beaucoup plus l'impact du génome sur le dessin. En effet, les paramètres définissent comment le sommet se place par rapport à ses voisins et les sommets qui l'entourent. De ce fait, utiliser des paramètres issus d'un génome différent permettra tout de même de conserver certaines caractéristiques de placement. Cela permet de fournir une bien meilleure stabilité lors de croisement entre deux génomes.
De plus, afin d'augmenter la réutilisabilité des génomes, deux informations sont ajoutées à chacun d'eux. Le contexte dans lequel il a été utilisé et le comportement qu'il a provoqué. Dans le cas des graphes, le contexte correspond à une série de métrique topologique (liées aux caractéristiques intrinsèques du graphe), comme le degré, le coefficient de clustering ou la betweeness centrality. Ces métriques sont issues de calculs réalisés sur le graphe et sont analysées de manière détaillée dans [6]. Le comportement est constitué de métriques graphiques : la longueur des arêtes, la distance au centre du graphe, la distance moyenne des plus proches sommets, etc. Ces métriques correspondent aux critères esthétiques rencontrés dans la littérature. Plusieurs articles de Purchase [16, 17] apportent un grand nombre de précisions sur ces métriques et leurs effets sur la compréhension que les utilisateurs ont des graphes.
Cette caractérisation de l'usage des génomes permettra par la suite d'utiliser ceux qui ont fourni de bons résultats pour des sommets similaires. De même, il sera possible d'étudier facilement les caractéristiques graphiques de génomes ayant été évalués favorablement. Ces informations sont stockées dans une base de données qui permet de centraliser l'ensemble des génomes intéressants testés lors des exécutions de l'algorithme génétique. En particulier, cela favorisera les échanges de génomes entre des graphes globalement différents mais présentant des similarités locales.

Le premier problème qui s'est posé lors de l'utilisation de cet algorithme génétique a été la mise

au point d'un évaluateur automatique. Un grand nombre de caractérisations existent déjà dans la littérature, en particulier dans le cadre des fonctions minimisées par les algorithmes par modèle de force. Plusieurs ont été essayées comme évaluateur, mais les résultats obtenus ont été assez décevants. En effet, l'algorithme trouvait des situations extrêmes (tous les sommets rassemblés en un même point) qui permettaient d'optimiser la fonction objectif. La piste finalement sélectionnée a consisté en l'utilisation d'une métrique de similarité pour recopier un bon layout.

## 4. Métrique de similarité

Le problème de la similarité entre 2 layouts se rapproche beaucoup de la comparaison de formes en biologie. Une première approche consiste à trouver la meilleure superposition possible de deux ensembles de points. L'analyse procrustéenne est une méthode qui permet d'obtenir ce type de résultat. Le principe consiste à chercher la transformation linéaire (translation, mise à l'échelle, rotation puis symétrie) qui permet de minimiser la distance entre les positions de chaque point dans les deux ensembles. Les articles de Rohlf et Slice de 1990 [18] et de Goodall en 1991 [10] présentent plusieurs applications de cette méthode.
Cependant, cette comparaison est relativement coûteuse (complexité en temps en $O(n^3)$ avec $n$ le nombre de points). Cela vient principalement de la recherche de la rotation optimale, qui consiste à tester toutes les rotations possibles pour trouver celle qui permet de minimiser les distances entre les points respectifs. D'autres approches ont aussi été envisagées, en particulier la mise en place de systèmes de coordonnées facilitant la comparaison. Par exemple, les coordonnées de Bookstein [3, 4] permettent de représenter l'ensemble des éléments à partir de deux points de référence.
Cependant, ces méthodes issues de la biologie ne s'appliquent pas parfaitement aux graphes, car la recherche de la meilleure superposition ne permet que de prendre en compte une transformation globale. De plus, les points caractérisant les formes sont toujours positionnés sur la surface externe, ce qui n'est que rarement le cas pour un graphe.
Assez peu de méthodes de comparaison ont été mises au point spécifiquement pour les graphes. Dans leur article de 1998 Bield et al. [2] utilisent une mesure de la similarité afin de fournir à l'utilisateur un ensemble de dessins différents d'un même graphe. Elle consiste à rechercher un couplage conservant le voisinage et minimisant la distance entre chaque paire de points. Mais cela ne permet pas d'obtenir une position précise par rapport au reste du graphe pour chaque point.
Une nouvelle méthode de comparaison a donc été mise au point en s'inspirant des éléments présents dans les différents articles cités ci-dessus. Elle se base sur le principe des triangles semblables. En effet, si tous les triangles du graphe (en prenant 3 sommets parmi les $n$) sont semblables entre les deux layouts, ces derniers pourront être considérés comme globalement semblables. Le calcul de cette mesure se base, pour une paire de sommets $i, j$, sur le quotient entre leur distance dans le premier layout et leur distance dans le deuxième, notée $q(i,j)$. Soit :

$$q(i,j) = \frac{d_1(i,j)}{d_2(i,j)}$$

avec $d_1$ la distance dans le premier layout et $d_2$ dans le deuxième. La valeur de $q(i,j)$ lorsque $d_2(i,j)$ est nulle n'est pas définie. Le layout 2 correspond à celui dont l'algorithme essaye de se rapprocher, il est donc nécessaire que tous les sommets aient des positions distinctes dans ce layout. De plus, il est nécessaire d'avoir $i \neq j$, le cas d'égalité étant de toute façon inintéressant (un sommet est toujours à distance nulle de lui même).
La similarité est alors fournie par l'écart type de cette mesure divisé par sa valeur moyenne. Deux dessins seront semblables si cette valeur est nulle. La normalisation permet de supprimer l'impact du facteur d'échelle. En particulier, cela évite qu'un layout dans lequel tous les sommets sont rassemblés présente une similarité très forte à tout autre dessin du même graphe.

$$s_G = \frac{ecart - type(q)}{\overline{q}} = \frac{\sqrt{\dfrac{\sum\limits_{\{i,j\}\in S} (q(i,j) - \overline{q})^2}{|S|}}}{\overline{q}}$$

avec S l'ensemble des paires de sommets, soit $S = \{\{i,j\}/\{i,j\} \in V^2, i \neq j\}$ et $\overline{q}$ la valeur moyenne de $q$ sur $S$.

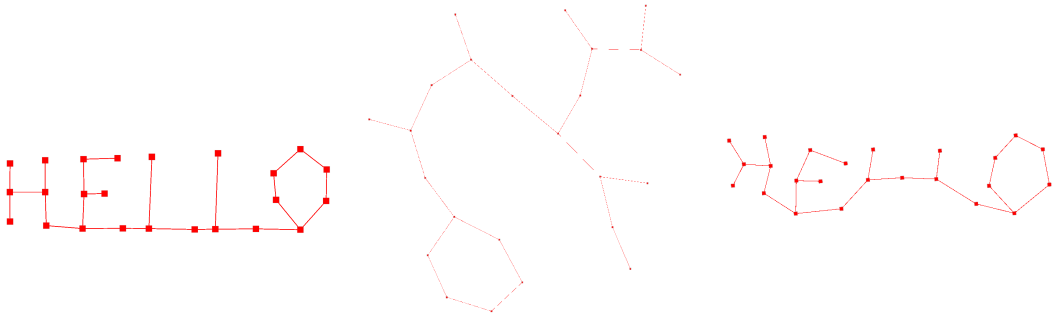
Pour obtenir un score pour chaque génome, cet écart-type doit être calculé localement à chaque sommet. Pour cela, le calcul pour le sommet $i$ est réalisé sur un sous-ensemble de $S$ noté $S_i$ correspondant à l'ensemble des paires contenant $i$, soit : $S_i = \{\{j,k\}/\{j,k\} \in S, i \in \{j,k\}\}$. Par contre, $\overline{q}$ reste la valeur moyenne de $q$ sur l'ensemble du graphe. La similarité associée au sommet $i$ est donc :

$$s_i = \frac{\sqrt{\dfrac{\sum\limits_{\{j,k\}\in S_i} (q(j,k) - \overline{q})^2}{|S_i|}}}{\overline{q}}$$

Dans les cas de rotations ou de symétries locales, tous les sommets impactés par la transformation resteront similaires entre eux, l'impact sur le score sera donc plus faible que dans le cas de l'analyse procustéenne. De plus, cette métrique a une complexité en temps en $O(|V|^2)$ puisque le coût le plus important consiste à calculer les distances pour chaque paire de sommets du graphe.

## 5. Cas d'application et preuve de concept

Le premier cas d'application de cette méthode a été la réplication d'un dessin de graphe représentant le mot "hello" (voir figure 1d). Ce graphe présente plusieurs intérêts, il est de taille restreinte (24 sommets pour 25 arêtes), la ligne du bas permet d'avoir un alignement couvrant l'ensemble de la largeur du graphe, il contient plusieurs angles droits, des arêtes de taille variable et une boucle. Comme montré sur la figure 1e, le dessin par défaut de GEM pour ce graphe est très éloigné de l'objectif souhaité. La figure 1f présente un résultat obtenu pendant une des exécutions.



(d) Le dessin objectif

(e) Le dessin obtenu avec la version originale de GEM

(f) Un des dessins obtenus à l'aide de la méthode

FIGURE 1 – Plusieurs dessins du graphe du cas d'application

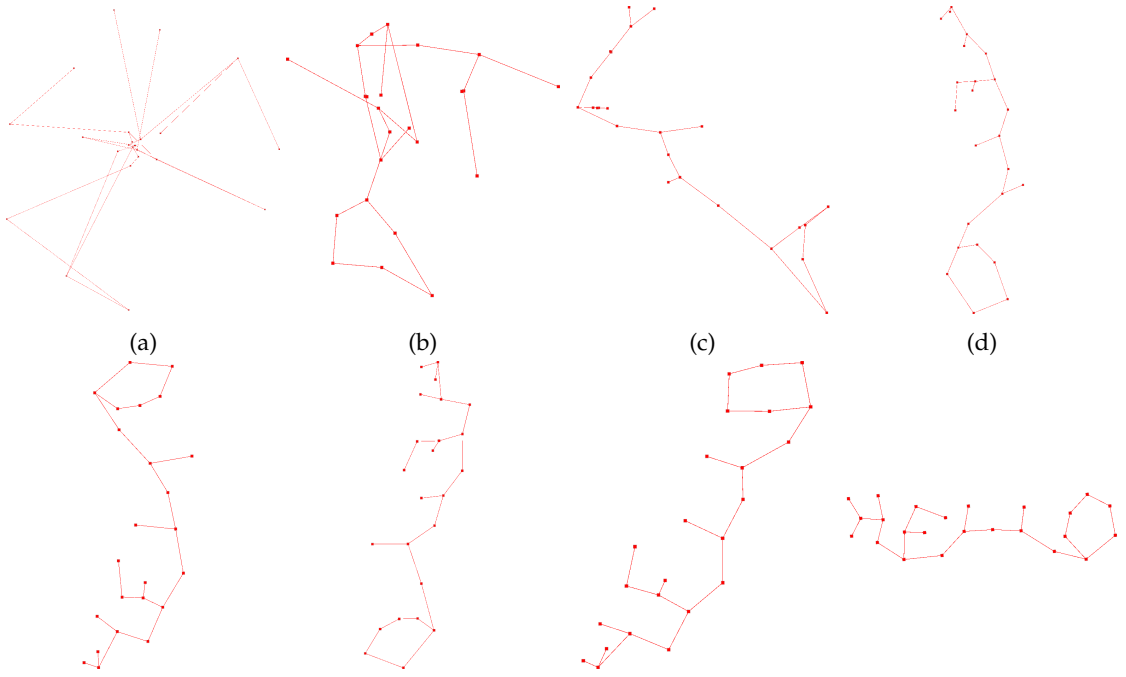
L'œil humain est très habitué à reconnaître rapidement des lettres ou des formes approchant ces dernières. Cela a facilité l'étude des premiers résultats. En effet, il a suffi d'afficher côte à côte les dessins obtenus pour sélectionner de visu les meilleurs. Une étude de leurs caractéristiques a ensuite permis de mettre au point une méthode de sélection automatique. Elle consiste à sélectionner les graphes qui minimisent la quantité suivante :

$$S_G = \min_{i\in V} s_i + \overline{s} + \max_{i\in V} s_i$$

Le temps d'exécution de l'algorithme en utilisant un seul processeur (2.8 Ghz) était d'environ 30 min, l'occupation mémoire étant stable et strictement négligeable (inférieure à 10 Mo). La figure 2 présente l'évolution d'une population lors d'une exécution.

## 6. Suite du projet

Ces premiers résultats sont très encourageants et viennent confirmer la possibilité d'utiliser l'algorithme de modèle de force choisi pour dessiner des graphes très différents de ceux obtenus avec



(a) (b) (c) (d)

(e) (f) (g) (h) Dessin issu d'une autre population

FIGURE 2 – L'amélioration progressive du dessin vers l'objectif

les paramètres par défaut. Cependant, le graphe ainsi dessiné ne présente que très peu d'intérêt car in ne fait que reproduire une solution fournie au préalable. Toutefois, cela permet d'acquérir des données qui permettront ensuite d'établir une méthode d'évaluation permettant de dire si un dessin est intéressant de manière autonome.

La prochaine étape va être de remplir progressivement la base de données avec des génomes permettant de dessiner correctement différents types de graphes. Des dessins de ces graphes communément appréciés par la communauté seront recopiés selon la méthode actuelle. Il sera ensuite possible de déduire un évaluateur à partir des métriques graphiques de ces bons dessins. Cela repose sur l'hypothèse pour laquelle si deux sommets sont similaires, une manière correcte de dessiner l'un conviendra aussi pour l'autre.

Des graphes issus du domaine de la bio-informatique seront utilisés pour valider cette procédure. De petits graphes seront utilisés au début afin d'apprendre les différents motifs. Ensuite, des graphes plus complets seront recopiés en se basant sur les résultats obtenus avec les premiers. Enfin, il faudra essayer de construire un évaluateur afin de tenter de dessiner les mêmes graphes sans utiliser la métrique de similarité. Il sera ensuite possible de tenter de dessiner de nouveaux réseaux métaboliques pour valider la méthode.

## 7. Conclusion

Ces travaux visent à apporter aux utilisateurs novices une nouvelle manière d'approcher le dessin de graphe. En paramétrant finement et automatiquement un algorithme masse-ressort, nous avons montré qu'il est possible d'obtenir des dessins très différents d'un même graphe. Un algorithme génétique permet de fournir des paramétrages intéressants, en s'appuyant pour l'instant sur une métrique de similarité, puis sur une méthode d'évaluation autonome.

Cela va permettre de fournir plusieurs dessins viables d'un même graphe à l'utilisateur afin qu'il puisse déterminer quel layout s'adapte au mieux à ses besoins. En effet, il est difficile pour un novice de décrire à priori un dessin qui lui conviendrait. Or choisir dans la panoplie d'algorithmes de dessins disponibles puis paramétrer efficacement ces algorithmes nécessite une grande expertise du domaine et une idée précise du résultat attendu. De ce fait, la plupart des programmes actuels fournissent un unique dessin dont l'utilisateur doit se satisfaire.

La méthode présentée dans cet article apparaît donc comme un compromis intéressant entre ces deux situations. Cela pourrait permettre une amélioration de la qualité des dessins utilisés par les utilisateurs novices, et ainsi augmenter fortement l'intérêt de ce mode de représentation de données pour le grand public. Cependant, il reste encore de nombreuses avancées à faire pour que ces travaux arrivent à maturité, mais les pistes envisagées semblent prometteuses.

**Bibliographie**

1. D. Archambault, D. Auber, and T. Munzner. TopoLayout : Multi-Level Graph Layout by Topological Features. *IEEE transactions on visualization and computer graphics*, 2007.
2. T. Biedl, J. Marks, and K. Ryall. Graph multidrawing : Finding nice drawings without defining nice. *Graph Drawing*, 1998.
3. F. L. Bookstein. Principal warps : Thin-plate splines and the decomposition of deformations. *Pattern Analysis and Machine Intelligence*, 1989.
4. F. L. Bookstein. *Morphometric tools for landmark data : geometry and biology*, volume 10. Cambridge University Press, 1991.
5. U. Brandes. Keynote address : Why everyone seems to be using spring embedders for network visualization, and should not. *Visualization Symposium (PacificVis), 2011 IEEE*, 2011.
6. U. Brandes and T. Eriebach. *Network analysis Methodological foundations*, volume 3418. Springer, 2005.
7. P. Eades. A heuristic for graph drawing. *Congressus Numerantium*, 42 :149–160, 1984.
8. A. Frick, A. Ludwig, and H. Mehldau. A fast adaptive layout algorithm for undirected graphs (extended abstract and system demonstration). *Graph Drawing*, 1995.
9. T. M. J. Fruchterman and E. M. Reingold. Graph drawing by force-directed placement. *Software : Practice and Experience*, 21(11) :1129–1164, November 1991.
10. C. Goodall. Procrustes methods in the statistical analysis of shape. *Journal of the Royal Statistical Society. Series B*, 53(2) :285–339, 1991.
11. L. Grammel, M. Tory, and M.-A. Storey. How information visualization novices construct visualizations. *IEEE transactions on visualization and computer graphics*, 16(6) :943–52, 2010.
12. L. J. Groves, Z. Michalewicz, P. V. Elia, and C. Z. Janikow. Genetic algorithms for drawing directed graphs. *for Intelligent Systems*, 1990.
13. T. Kamada and S. Kawai. An algorithm for drawing general undirected graphs. *Information processing letters*, 31(April) :7–15, 1989.
14. C. Kosak, J. Marks, and S. Shieber. Automating the Layout of network diagrams with specified visual organization. *IEEE Transactions on Systems, Man, and Cybernetics*, 24(3) :440–454, 1994.
15. A. Markus. Experiments with Genetic Algorithms for Displaying Graphs. *Proceedings. 1991 IEEE Workshop on Visual Languages*, pages 62–67, 1991.
16. H. Purchase. Which aesthetic has the greatest effect on human understanding ? *Graph Drawing*, 1997.
17. H. Purchase. Metrics for Graph Drawing Aesthetics. *Journal ofVisualLanguages and Computing*, 13 :501–516, 2002.
18. F. J. Rohlf and D. Slice. Extensions of the Procrustes Method for the Optimal Superimposition of Landmarks. *Systematic Zoology*, 39(1) :40–59, 1990.