Combining System Design and Path Planning

Laurent Denarie, Kevin Molloy, Marc Vaisset, Thierry Simeon, Juan Cortés

Combining System Design and Path Planning

Laurent Denarie[1], Kevin Molloy[1], Marc Vaisset[1],
Thierry Siméon[1], and Juan Cortés[1]

LAAS-CNRS, Université de Toulouse, CNRS, Toulouse, France
juan.cortes@laas.fr

Abstract. This paper addresses the simultaneous design and path planning problem, in which features associated to the bodies of a mobile system have to be selected to find the best design that optimizes its motion between two given configurations. Solving individual path planning problems for all possible designs and selecting the best result would be a straightforward approach for very simple cases. We propose a more efficient approach that combines discrete (design) and continuous (path) optimization in a single stage. It builds on an extension of a sampling-based algorithm, which simultaneously explores the configuration-space costmap of all possible designs aiming to find the best path-design pair. The algorithm filters out unsuitable designs during the path search, which breaks down the combinatorial explosion. Illustrative results are presented for relatively simple (academic) examples. While our work is currently motivated by problems in computational biology, several applications in robotics can also be envisioned.

Keywords: robot motion planning, sampling-based algorithms, computational biology, protein design

1 Introduction

System design and path planning problems are usually treated independently. In robotics, criteria such as workspace volume, workload, accuracy, robustness, stiffness, and other performance indexes are treated as part of the system design [1, 2]. Path planning algorithms are typically applied to systems with completely fixed geometric and kinematic features. In this work, we propose an extension of the path planning problem, in which some features of the mobile system are not fixed *a priori*. The goal is to find the best design (i.e. values for the variable features) to optimize the motion between given configurations.

A brute-force approach to solve this problem would consist of individually solving motion planning problems for all possible designs, and then selecting the design providing the best result for the (path-dependent) objective function. However, because of the combinatorial explosion, only simple problems involving a small number of variable design features can be treated using this naive approach. We propose a more sophisticated approach that simultaneously considers system design and path planning. A related problem is the optimization of geometric and kinematic parameters of a robot to achieve a given end-effector

trajectory, usually referred to as kinematic synthesis [3]. Nonetheless, the problem we address in this work (see Section 2.1 for details) is significantly different, since we assume that all kinematic parameters and part of the geometry of the mobile system are provided as input. The design concerns a discrete set of features that can be associated to the bodies of the mobile system, such as shape or electrostatic charge, aiming to find the best possible path between two given configurations provided a path cost function. Very few works have considered such a hybrid design and path planning problem. One of the rare examples is a recently proposed method for UAVs path planning [4] where the optimal path planning algorithm considers several possible flying speeds and wing reference areas to minimize path risk and time. Since the considered configuration space is two-dimensional, the proposed solution is based on an extension of Dijkstra's algorithm working on a discrete representation of the search-space. This type of approach cannot be applied in practice to higher-dimensional problems, as the ones we address.

Sampling-based algorithms have been developed since the late 90s for path planning in high-dimensional spaces [5, 6], which are out of reach for deterministic, complete algorithms. Our work builds on this family of algorithms, which we extend to treat a combinatorial component in the search-space, associated to the systems design, while searching for the solution path. Our approach presents some similarities with methods that extend sampling-based path planning algorithms to solve more complex problems such as manipulation planning [7] or multi-modal motion planning [8], which also involve search-spaces with hybrid structure. As in these other works, the proposed algorithm simultaneously explores multiple sub-spaces aiming to find solutions more efficiently. Nevertheless, the hybrid design problem addressed here is completely different.

This paper presents the Simultaneous Design And Path-planning algorithm (SDAP), which is based on the T-RRT algorithm [9]. As explained in Section 2.2, the choice of T-RRT as a baseline is guided by the type of cost function we apply for the evaluation of path quality. Nevertheless, other sampling-based algorithms can be extended following a similar approach.

We demonstrate the good performance of the method on relatively simple, academic examples (Section 4). These simple examples allow us to apply the naive exhaustive method, whose results can be used as a reference to evaluate the performance and the quality of the solutions produced by the SDAP algorithm. Results show that SDAP is able to find the best path-design pairs, requiring much less computing time than the naive method. This advantage increases with the complexity of the problem.

Although the application of the proposed approach to problems of practical interest is out of the scope of this paper, we note that our motivation comes from problems in computational biology. For more than a decade, robotics-inspired algorithms have been applied to this area [10–13]. The method presented in this paper aspires to solve problems related to computational protein design [14, 15], which aims to create or modify proteins to exhibit some desired properties. Progress in this field promises great advances in pharmacology, biotechnology,

and nanotechnology. Protein design is extremely challenging, and although there have been some considerable strides in the last years [16, 17], the problem remains largely open. Current approaches focus on a static picture (i.e. search the amino-acid sequence that stabilizes a given structure), whereas dynamic aspects related to protein function are rarely considered. Our goal behind this work is to develop new methods to optimize functional protein motions. In addition to computational protein design, applications of the proposed approach in robotics can be envisioned, as briefly mentioned in the conclusion.

2 Problem Formulation and Approach

This section defines the problem addressed in this work, along with some notation, and presents an overview of the proposed approach.

2.1 Problem Definition

Let us consider an articulated linkage $\mathcal{A}$ consisting of n rigid bodies, $A_1..A_n$. The kinematic parameters of $\mathcal{A}$ are static and are supplied as input. The geometry of the the rigid bodies A_i can admit some variability, as well as other physical properties (mass, electrostatic charge, ...). More precisely, a discrete set of m design features, $f_1..f_m$, is defined and each body $A_i \in \mathcal{A}$ is assigned a design feature $f_j \in \mathcal{F}$. We denote d as a vector of length n that represents the design features assigned to all the rigid bodies in $\mathcal{A}$, i.e. d defines a particular design. $\mathcal{D}$ denotes the set of possible combinations of assignments of features for $\mathcal{A}$, i.e. $\mathcal{D}$ defines all possible designs. $\mathcal{D}$ is referred to as the design space, which is a discrete space containing m^n elements. A given configuration of $\mathcal{A}$ is denoted by q. Let $\mathcal{C}$ denote the configuration space. Note that for each $q \in \mathcal{C}$, only a subset of the possible designs $\mathcal{D}$ can be assigned, since some designs are not compatible with some configurations due to self-constraints or environment constraints.

The workspace of $\mathcal{A}$ is constrained by a set of obstacles $O_i \in \mathcal{O}$. $\mathcal{C}^d_{free}$ denotes all valid, collision-free configurations of $\mathcal{A}$ for a given vector d of design features. A path P connecting two configurations q_{init} and q_{goal} of $\mathcal{A}$ with design d is defined as a continuous function $P : [0,1] \rightarrow \mathcal{C}$, such that $P(0) = q_{\text{init}}$ and $P(1) = q_{\text{goal}}$. The path is said to be collision-free if $\forall t \in [0,1], P(t) \in \mathcal{C}^d_{free}$. $\mathcal{C}_{free}$ is the union of all individual $\mathcal{C}^d_{free}$:

$$\mathcal{C}_{free} = \bigcup_{d \in \mathcal{D}} \mathcal{C}^d_{free} \ .$$

$\mathcal{P}_{free}$ denotes the set of all feasible, collision-free paths connecting q_{init} to q_{goal}, considering all possible designs ($\forall d \in \mathcal{D}$).

A cost function $c : \mathcal{C}_{free} \times \mathcal{D} \rightarrow \mathbb{R}_+$ associates to each pair (q, d) a positive cost value, $\forall q \in \mathcal{C}_{free}$ and $\forall d \in \mathcal{D}$. Another cost function $c_P : \mathcal{P}_{free} \times \mathcal{D} \rightarrow \mathbb{R}_+$ is also defined to evaluate the quality of paths. In this work, the path cost function c_P is itself a function of the configuration cost function c, i.e. c_P is a functional.

More precisely, we consider the *mechanical work* criterion as defined in [9, 18] to evaluate paths, which aims to minimize the variation of the configuration cost c along the path. This criterion is a suitable choice to evaluate path quality in many situations [9], and is particularly relevant in the context of molecular modeling. Nevertheless, other cost functions can be considered, such as the integral of c along the path. A discrete approximation, with constant step size $\delta = 1/l$, of the mechanical work (MW) cost of a path P for a system design d can be defined as:

$$c_P(P,d) = \sum_{k=1}^{l} \max\left\{0\,,\; c\left(P\left(\frac{k}{l}\right),d\right) - c\left(P\left(\frac{k-1}{l}\right),d\right)\right\} \quad . \tag{1}$$

The goal of our method is to find the best pair (P^*, d^*) such that:

$$c_P(P^*,d^*) = \min\{c_P(P,d) \,|\, P \in \mathcal{P}_{free}, d \in \mathcal{D}\} \quad . \tag{2}$$

2.2 Approach

A naive approach to solve the problem would be to compute the optimal cost path for each design $d \in \mathcal{D}$, and then choose the optimal design d^* that minimizes c_P. Such a brute-force approach can be applied in practice to simple problems involving a small number n of variable bodies and/or a few m design features (recall that the design space is size m^n). The method proposed below aims to solve the problem much more efficiently by combining both the discrete (design) and continuous (path) optimization in a single stage.

We assume that, for most problems of interest, the configuration space $\mathcal{C}$ is high-dimensional, so that exact/complete algorithms cannot be applied in practice to solve the path-planning part of the problem. For this, we build of sampling-based algorithms [19, 5], which have been very successful in the robotics community since the late 90s, and which have also been applied in other areas such as computational biology [10–13]. We also assume that the cardinality of the design space $\mathcal{D}$ is moderately high, such that a relatively simple combinatorial approach can be applied to treat the design part of the problem.

The idea is to explore $\mathcal{C}_{free}$ to find paths between q_{init} and q_{goal} simultaneously considering all possible deigns $d \in \mathcal{D}$. To reduce the number of configuration-design pairs (q, d) to be evaluated during the exploration, it is important to apply an effective filtering strategy. The choice of the particular sampling-based path planning algorithm and filtering strategy mainly depend on the type of objective function c_P being considered. The approach described below has been developed to find good-quality solutions with respect to the MW path evaluation criterion (1). In this work, we extend the T-RRT algorithm [9], which finds paths that tend to minimize cost variation by filtering during the exploration tree nodes that would produce a steep cost increase. Following a similar approach, alternative algorithms and the associated filtering strategies could be developed to optimize other path cost functions. For instance, variants of RRT* [20] or FMT* [21] could be considered for optimizing other types of monotonically increasing cost functions.

3 Algorithm

This section presents the SDAP algorithm, building upon a single-tree version of T-RRT. However, the approach is directly applicable to multi-tree variants [22]. First, the basic algorithm is introduced, followed by additional explanation on the tree extension strategy and a brief theoretical analysis.

3.1 SDAP Algorithm

The SDAP pseudo-code is shown in Algorithm 1. A search tree, $\mathcal{T}$, is created with q_{init} as the root node. The tree is grown in configuration space through a series of expansion operations. Each node s in $\mathcal{T}$ encodes a configuration q and a set of designs $D \subseteq \mathcal{D}$ for which the configuration is valid. Each node's set of designs D is a subset of its parent's designs, i.e. $Designs(s) \subseteq Designs(Parent(s))$.

During each iteration, a random configuration (q_{rand}) is generated (line 3). In T-RRT, a new node (q_{new}) is created by expanding the nearest node in $\mathcal{T}$ (q_{near}) in the direction of q_{rand} for a distance δ. q_{new} is then conditionally added to $\mathcal{T}$ based on a transition test. A common heuristic for this transition test is the Metropolis criterion, traditionally applied in Monte Carlo methods [23]. Transitions to lower cost nodes are always accepted and moves to higher costs nodes are probabilistically accepted. The probability to transition to a higher cost node is controlled by a temperature variable T. T-RRT dynamically controls T, as explained below.

SDAP modifies the expansion and transition test functions of the standard T-RRT algorithm in order to address the design and path planning problems simultaneously. At each iteration, SDAP attempts to expand at least one node per design in $\mathcal{D}$. The process is shown in Figure 1, where each design $d \in \mathcal{D}$ is encoded as a color on each node of the tree. During each iteration, SDAP expands a set of nodes that covers all designs $\mathcal{D}$. In other words, the `NearestNeighbors` function (line 4) returns a set, *Neighbors*, containing the closest node to q_{rand} for each design. In Figure 1, q_{rand} is shown in black and the 3 nodes in the set *Neighbors* are circled in red. Each node in *Neighbors* is extended towards q_{rand} (lines 6 - 10) creating new candidate nodes which are labeled s_1, s_2 , and s_3 in Figure 1. All 3 designs in s_1 fail the transition test, so the new node is not added to $\mathcal{T}$. For s_2 the blue design passes the transition test and the node is added to $\mathcal{T}$. Finally for s_3, 1 of the 3 designs (yellow) fails the transition test, resulting in a node with 2 designs being added to $\mathcal{T}$.

3.2 Controlling Tree Expansion

The transition test is governed by the temperature parameter (T). T-RRT automatically adjusts T during the exploration and has been shown highly effective in balancing tree exploration and tree refinement [9]. At each iteration, T-RRT adjusts T by monitoring the acceptance rate of new nodes. SDAP extends this idea by maintaining a separate temperature variable $T(d)$ for each design $d \in \mathcal{D}$. A given design d can appear in multiple nodes in *Neighbors*. For each design

Algorithm 1: SDAP Algorithm

input : the configuration space $\mathcal{C}$; the design space $\mathcal{D}$; the cost function c
the start state q_{init}; the goal state q_{goal}
number of iterations $MaxIter$
output: the tree $\mathcal{T}$

1 $\mathcal{T} \leftarrow$ `InitTree`$(q_{\text{init}}, \mathcal{D})$
2 **while** *not StoppingCriterion* $(\mathcal{T}, q_{\text{goal}}, MaxIter)$ **do**
3 $q_{\text{rand}} \leftarrow$`Sample(`$\mathcal{C}$`)`
4 $Neighbors \leftarrow$ `NearestNeighbors`$(\mathcal{T}, q_{\text{rand}}, \mathcal{D})$
5 TransitionTest.Init()
6 **for** $s_{\text{near}} \in Neighbors$ **do**
7 $q_{\text{new}} \leftarrow$ `Extend`$(q_{\text{rand}}, s_{\text{near}})$
8 $D \leftarrow$ `TransitionTest`$(\mathcal{T}, s_{\text{near}}, q_{\text{new}}, c)$
9 **if** `NotEmpty`(D) **then**
10 `AddNode`$(\mathcal{T}, s_{\text{near}}, q_{\text{new}}, D)$

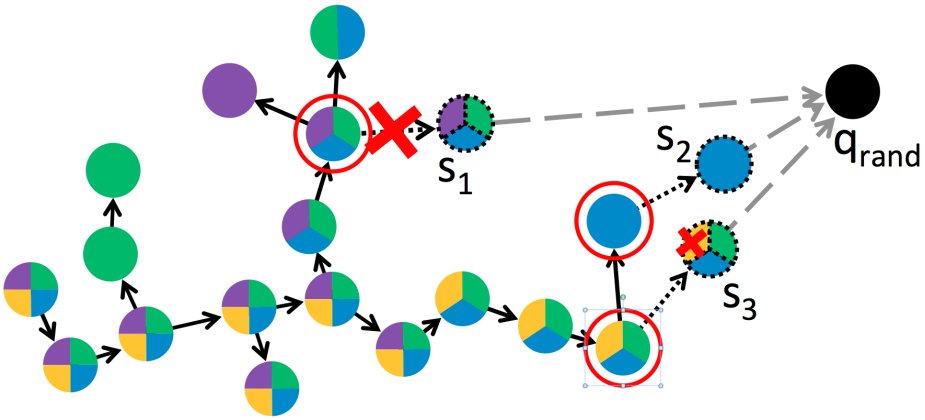

Fig. 1. An expansion operation for SDAP. Designs are encoded as colors within each node. The nodes being expanded are circled in red. The expansion towards node s_1 fails for all 3 designs, The expansion to s_2 succeeds, and the expansion to s_3 succeeds for 2 of the 3 designs.

d, the node in *Neighbors* closest to q_{rand} is identified. The temperature $T(d)$ is adjusted based on the success or failure of the extension operation from this node for design d.

The pseudocode for the transition test function is shown in Algorithm 2. The *Neighbors* set is processed in ascending order of the distance of each node from q_{rand} (line 6 of Algorithm 1). For the node being expanded (s_{near}), each design d has its cost evaluated (line 4). Transitions to lower cost nodes are always accepted (line 8). Transitions to higher cost nodes are subjected to probabilistic acceptance (line 10). The set V (lines 12 and 18) tracks designs which have had

Algorithm 2: TransitionTest($\mathcal{T}, s_{\text{near}}, q_{\text{new}}, c$)

input : the input tree $\mathcal{T}$; vector of temperatures T
parent node s_{near}; new node q_{new} ; the cost function c
temperature adjustment rate T_{rate}; Boltzmann constant K
internal: set of designs V with adjusted temperatures in this iteration
output : vector of designs D that pass the transition test

```
S ← φ
for d ∈ Designs(s_near) do
    if CollisionTest(q_new,d) == False then
        c_near = c(Config(s_near),d); c_new = c(q_new,d)
        success ← false
        Δc = c_new - c_near
        if Δc < 0 then
            success ← true
        else
            if exp(−Δc / (K · T(d))) > UniformRand()) then
                success ← true
            if d ∉ V then
                if success then
                    T(d) ← T(d) / 2^((Δc) / energyRange(T,d))
                else
                    T(d) ← T(d) · 2^T_rate
        if success then D ← D ∪ d
        V ← V ∪ d
return (D)
```

their temperature adjusted during this iteration. The function returns the set D of designs that pass the transition test.

3.3 Theoretical Analysis

In this section we provide some theoretical analysis of SDAP algorithm's completeness and path optimality. A theoretical analysis of the complexity of SDAP with respect to the brute-force approach is difficult, since both are stochastic processes. In this work, we instead provide empirical results in Section 4 that clearly show SDAP's efficiency versus an exhaustive search of paths for all possible designs.

Probabilistic Completeness: SDAP's probabilistic completeness directly derives from that of RRT [19], which is inherited by T-RRT under the condition to guarantee a strictly positive probability of passing the transition test as explained in [9]. Since SDAP maintains this property by incorporating temperatures in the

transition test for each given design $d \in \mathcal{D}$, it also ensures the positive transition probability and that each $\mathcal{C}^d_{free}$ will be completely sampled, thus maintaining the probabilistic completeness of the algorithm.

Path Optimality: The current SDAP implementation is based on T-RRT, which has been empirically shown to compute paths that tend to minimize cost with respect to the MW criterion [9], but without theoretical guarantee of optimality. Using anytime variants of T-RRT (AT-RRT or T-RRT*) [18] would provide asymptotic convergence guarantee. Implementing these within SDAP remains as future work.

4 Empirical Analysis and Results

As a proof of concept, we apply SDAP to a set of academic problems. SDAP is implemented as an adaptation of the Multi-T-RRT algorithm [22], with two trees growing from the initial and goal configurations. The search stops when the algorithm is able to join the two trees. For each problem, SDAP is compared against a naive approach consisting of multiple independent runs of Multi-T-RRT on each designs $d \in \mathcal{D}$.

4.1 Test System Description

The test system is a 2D articulated mechanism with a fixed geometry surrounded with fixed obstacles. The bodies $A_1..A_n$ are circles with radius R. The first body A_1 is a fixed base. The other bodies $A_2..A_n$ are articulated by a rotational joint centered on the previous rigid body that can move in the interval $[0, 2\pi)$. A configuration q is described by a vector of $n-1$ angles corresponding to the value of each rotational joint. The features $f_1..f_n$ assigned to each body are electrostatic charges in $\mathcal{F} = \{-1, 0, 1\}$ (i.e. $m = 3$). The design vector d contains n charges $f_1..f_n$ associated to each rigid body $A_1..A_n$ of the mechanism. In the following, d can be written as a string, with each charge $(-1, 0, 1)$ corresponding to N, U, and P respectively. For example, the design NPUN corresponds to the vector $d = (-1, 1, 0, -1)$. Obstacles $O_1..O_k$ are circles of radius R and have electrostatic charges with predefined values $g_i \in \mathcal{F}$.

The cost function is inspired by a simple expression of the potential energy of a molecular system. It contains two terms, one corresponding to the Lennard-Jones potential and the other to the electrostatic potential. It is defined as:

$$c(q, d) = LJ(q, d) + ES(q, d) \tag{3}$$

with:

$$LJ(q,d) = \sum_{i=1}^{|\mathcal{A}|-2} \left[\sum_{j=i+2}^{|\mathcal{A}|} \left(\frac{2 \cdot R}{\|A_i A_j\|} \right)^{12} - \left(\frac{2 \cdot R}{\|A_i A_j\|} \right)^{6} \right] + \sum_{i=1}^{|\mathcal{A}|} \left[\sum_{j=1}^{|\mathcal{O}|} \left(\frac{2 \cdot R}{\|A_i O_j\|} \right)^{12} - \left(\frac{2 \cdot R}{\|A_i O_j\|} \right)^{6} \right] \quad (4)$$

$$ES(q,d) = \sum_{i=1}^{|\mathcal{A}|-2} \left[\sum_{j=i+2}^{|\mathcal{A}|} \left(\frac{f_i \cdot f_j}{\|A_i A_j\|} \right) \right] + \sum_{i=1}^{|\mathcal{A}|} \left[\sum_{j=1}^{|\mathcal{O}|} \left(\frac{f_i \cdot g_j}{\|A_i O_j\|} \right) \right] \quad (5)$$

where $\|X_i X_j\|$ represents the Euclidean distance between the centers of the bodies/obstacles X_i and X_j.

SDAP is empirically tested using a 4 body and a 10 body scenarios described below. The objective is to find the path-design pair (P^*, d^*) that minimizes c_P.

Small 4 Body System: The first system consists of four bodies and five obstacles as shown in Figure 2. q_{init} and q_{goal} correspond to fully stretched configurations, to the left and to the right, represented with solid and dashed outlines respectively. The design space consists of $3^4 = 81$ possible combinations and the configuration space is 3 dimensional. This scenario favors designs with a negatively charged end-effector. The uncharged obstacles at the top and bottom of the workspace create a narrow passage that all solutions must pass through. Figure 3 shows a projection of the configuration-space costmap for two designs along with a solution path. One design has a negatively charged end-effector (UUUN) and one has a positively charged end-effector (UUUP). Angles 1 and 2 are projected onto the x and y axis respectively, with angle 3 being set to minimize the cost function. For the UUUN design, the costmap is highly favorable to the desired motion, starting at a high cost and proceeding downhill to a low cost area. The costmap associated with the UUUP design shows a non-favorable motion between the two states.

Larger 10 Body System: A larger system with 10 bodies and six obstacles is shown in Figure 2. The design space $\mathcal{D}$ contains $3^{10} = 59049$ possibilities, which cannot be exhaustively explored within a reasonable computing time, and is also challenging for SDAP because of memory issues (see discussions in Section 5). For that reason, two simplified versions of this scenario are constructed. The first one fixes the design for the first seven bodies $A_1..A_7$ as UUUUUUU. The remaining bodies (A_8, A_9, and A_{10}) can be designed, resulting in a design space of $3^3 = 27$ designs. The second version expands the design space to the last 4 bodies ($A_7..A_{10}$), resulting in a design space of $3^4 = 81$ designs. It both cases, the configuration space is 9 dimensional. Both versions of the 10-body system are constrained with the same obstacles. They were chosen so that designs with strongly positively or negatively charged end-effectors will be trapped at local minima resulting from attractive or repulsive forces generated by the bottom obstacles.

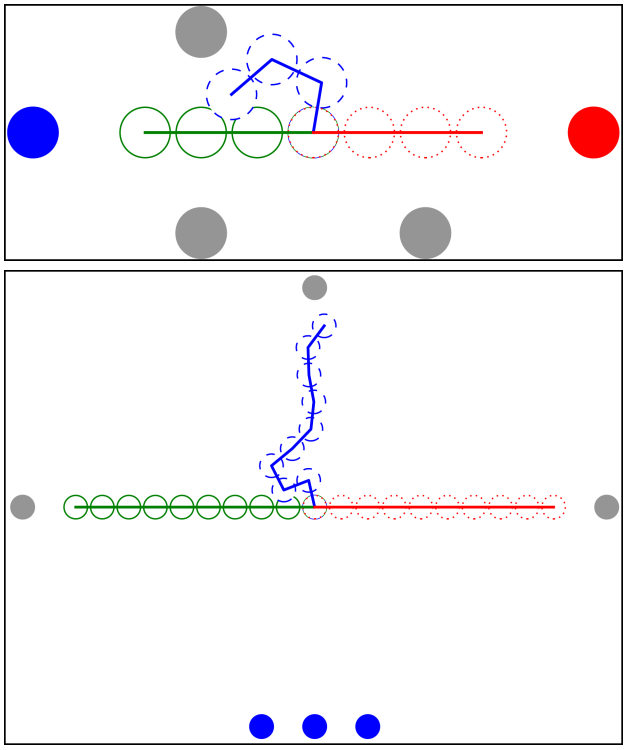

Fig. 2. A 4-body (top) and 10-body (bottom) scenario. Obstacles with positive charges are shown in solid red, negative in solid blue, neutral in gray. The initial state is shown in green with a solid line, a transition state shown in blue, and the goal state is shown in red with a dashed line.

4.2 Benchmark Results

We compare SDAP to a naive approach (solving individual problems for each design $d \in \mathcal{D}$) using the same Multi-T-RRT implementation. In other words, we compare one run of the SDAP algorithm against $|\mathcal{D}|$ runs of a single-design path search. Mutliple runs are performed (100 for the 4-body scenario, 50 for the 10-body scenario with 3 designed bodies, and 20 for the 10-body scenario with 4 designed bodies) to avoid statistical variance inherent with stochastic methods. The single-design explorations can spend time trying to escape local minima associated with the costmaps of unfavorable designs, causing very long execution times and high-cost paths. As we are not interested in finding a solution path for every possible design but only for the designs with low-cost paths, a timeout is enforced for the single-design explorations of 300 seconds for the 4-body scenario and 1,200 seconds for the 10-body scenarios. The SDAP algorithm was considered unsuccessful if it did not find a solution within 2,400 seconds.

All the runs were performed in a single threaded process on a Intel(R) Xeon(R) CPU E5-2650 0 @ 2.00GHz processor with 32GB of memory.

Small Scenario Results: The runtimes for the small scenario are shown at the top in Figure 4. For the single-design approach, the sum of the 81 runs to

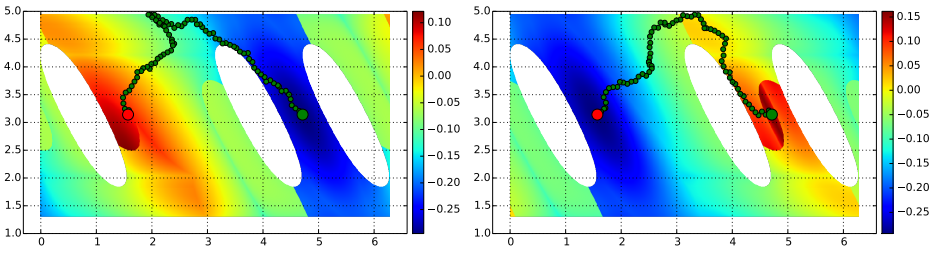

Fig. 3. Configuration-space costmap of the 4-body system projected onto the first two DOFs of the system expressed in radians. The initial configuration is indicated by the red dot on the left, and the goal by the green dot on the right. The plot on the left is for the UUUP design, the one to the right for UUUN. Each cell's cost is computed by finding the value of the 3rd angle that minimizes the cost.

cover $\mathcal{D}$ are plotted versus the SDAP runtime. The figure shows that SDAP is twice faster than the single-design approach. Although the variance in execution time for SDAP seems much larger than for the naive approach, recall that each complete run of the latter involves 81 runs of the Multi-T-RRT algorithm, which attenuates the overall variance. However, the computing time variance for a specific design can be much larger. Figure 5 (top) compares the solutions found by the two methods. SDAP successfully identifies the designs corresponding to the lowest-cost paths. Recall that the current implementation of SDAP terminates when one valid path is found. Asymptotic convergence to the global optimum could be guaranteed by implementing an anytime variant of the algorithm such as AT-RRT [18] (this remains as future work). The high density of nodes created by the SDAP algorithm (19,784 nodes on average compared to 698 for each single-design search) could be well exploited to improve the path cost by incrementally adding cycles.

Larger scenario results: The runtimes for both versions of the larger scenario are shown in Figure 4 (middle and bottom). In both cases, the difference in computing time between the two approaches increases significantly compared to the 4-body scenario. In the 3-designed-body version, SDAP is 26 times faster than the single-design search on average. In the 4-designed-body version, SDAP is 46 times faster. Note that, while the cardinality of design space $\mathcal{D}$ is multiplied by 3 between the two versions of the large scenario, the execution time of SDAP is only multiplied by 2.5 on average. The variance of the execution times is now lower for SDAP compared to the naive approach. The reason is that the performance of Multi-T-RRT highly depends on the roughness of the configuration-space costmap. In a smooth costmap, Multi-T-RRT will be quite fast with a low variance, whereas the time required to find a solution in a rugged costmap will be higher and will have a larger variance. SDAP's computing time is only dependent on the difficulty to find the best designs, which typically have a smoother costmap, whereas the single-design search has to find a solution for

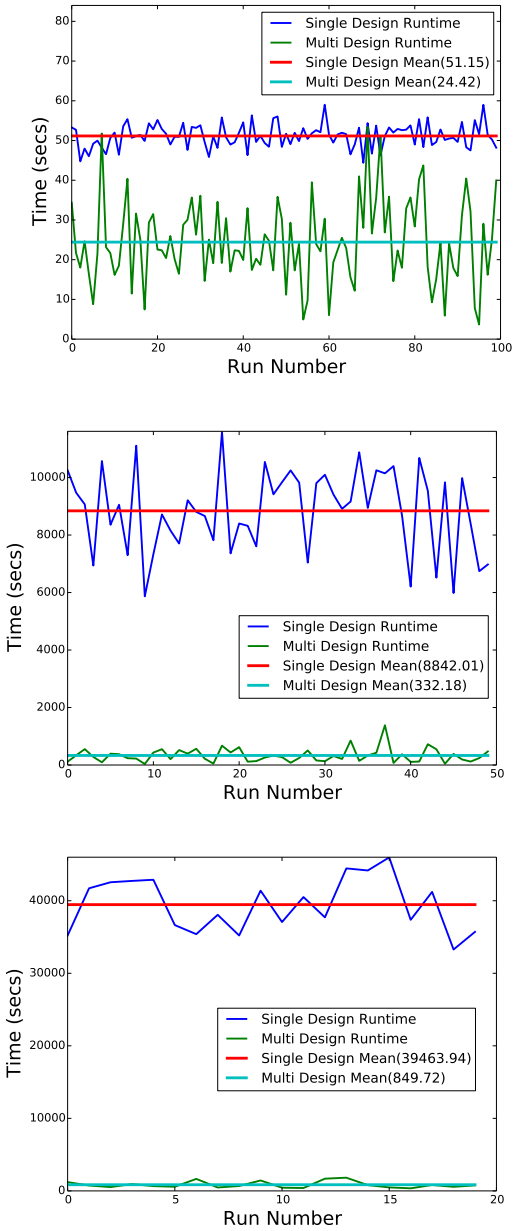

Fig. 4. Run times comparisons for the 4-body scenario (top), 10-body scenario with 3 designed bodies (middle) and 4 designed bodies (bottom). SDAP (green line) is compared against an exhaustive search using single-design explorations (blue line).

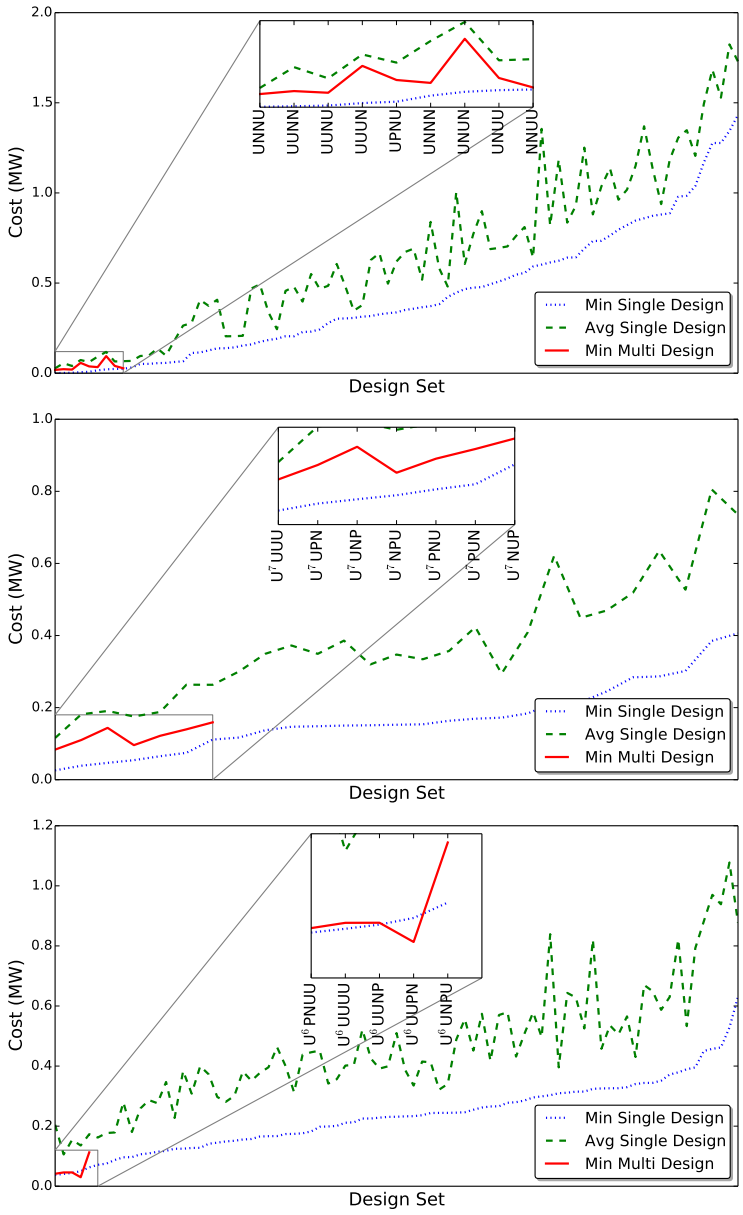

Fig. 5. The best cost paths for single-design runs (dotted blue and dashed green lines) and SDAP (red line) for the small scenario (top) and larger 3-designed-body (middle) and 4-designed-body (bottom). SDAP solutions shown only for discovered paths (does not exhaustively search). SDAP discovers the same low-cost designs as the exhaustive single-design searches.

every design, including those with a very rugged costmap. The 4-body scenarios is relatively simple, and thus even for very bad designs, a solution was found in close-to-constant time. But for the 10-body scenario, the problem is more complex, and the 27 (resp. 81) runs are not enough to attenuate a very high variance.

The single-design search reached the timeout 4 times over the 27 runs on average for the 3-designed-body version of the 10-body scenario, and 19 times over the 81 runs on average for the 4-designed-body version of the problem. The SDAP algorithm always found a solution before the timeout.

Figure 5 (middle and bottom) compares the solutions found by the two searches. Once again, SDAP successfully identifies the designs that yield the best path cost.

5 Conclusion

In this paper, we have presented an original formulation of a challenging problem combining system design and path planning, and have proposed a new approach to solve it building on sampling-based algorithms. The current implementation of SDAP is still preliminary, but it already shows significant gains in efficiency and accuracy compared to a brute-force approach.

The actual motivation of this work concerns computational biology. We are currently applying SDAP to help understand the effect of mutations in antibodies, which is a preamble to protein design (publication in preparation).

Several applications of SDAP in robotics can also be envisioned. In addition to the design of some robot's features to optimize its motion in a given workspace, it would also be possible to apply the proposed method to optimize the workspace layout for a given robot. One can also imagine applications for helping to the design of modular self-reconfigurable robots.

For future work, in addition to the implementation of a variant of SDAP with asymptotic optimality guarantees based on AT-RRT, we aim to introduce further improvements. The exploration of very-high-dimensional configuration and design spaces implies computer memory issues (the resulting tree/graphs are very large). A solution to this problem would be to introduce pruning stages during the exploration, as is done in the SST* algorithm [24]. Larger design spaces will also require SDAP to employ more sophisticated filters, such as those that incorporate statistical learning, to limit the design search and control the size of the search tree.

Acknowledgment

This work has been partially supported by the French National Research Agency (ANR) under project ProtiCAD (project number ANR-12-MONU-0015).

References

1. C. Gosselin and J. Angeles, "A global performance index for the kinematic optimization of robotic manipulators," *Journal of Mechanical Design*, vol. 113, no. 3, pp. 220–226, 1991.
2. J.-P. Merlet, "Optimal design of robots," in *Robotics: Science and Systems*, 2005.
3. J. M. McCarthy and L. Joskowitz, "Kinematic synthesis," in *Formal Engineering Design Synthesis*, J. Cagan and E. Antonson, Eds. Cambridge Univ. Press., 2001.
4. E. S. Rudnick-Cohen, S. Azarm, and J. Herrmann, "Multi-objective design and path planning optimization of unmanned aerial vehicles," in *Proc. 16th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference, AIAA Aviation*, 2015.
5. L. E. Kavraki, P. Svestka, J.-C. Latombe, and M. Overmars, "Probabilistic roadmaps for path planning in high dimensional configuration spaces," *IEEE Transactions on Robotics and Automation*, vol. 12, no. 4, pp. 566–580, 1996.
6. S. M. LaValle, *Planning Algorithms*. New York: Cambridge University Press, 2006.
7. T. Siméon, J.-P. Laumond, J. Cortés, and A. Sahbani, "Manipulation planning with probabilistic roadmaps," *Int. J. Robot. Res.*, vol. 23(7), pp. 729–746, 2004.
8. K. Hauser and J.-C. Latombe, "Multi-modal motion planning in non-expansive spaces," *Int. J. Robot. Res.*, vol. 29, no. 7, pp. 897–915, 2010.
9. L. Jaillet, J. Cortés, and T. Siméon, "Sampling-based path planning on configuration-space costmaps," *IEEE Trans. Robotics*, vol. 26, no. 4, pp. 635–46, 2010.
10. M. Moll, D. Schwarz, and L. E. Kavraki, *Roadmap Methods for Protein Folding*. Humana Press, 2007.
11. I. Al-Bluwi, T. Siméon, and J. Cortés, "Motion planning algorithms for molecular simulations: A survey," *Comput. Sci. Rev.*, vol. 6, no. 4, pp. 125–43, 2012.
12. B. Gipson, D. Hsu, L. Kavraki, and J.-C. Latombe, "Computational models of protein kinematics and dynamics: Beyond simulation," *Ann. Rev. Analyt. Chem.*, vol. 5, pp. 273–91, 2012.
13. A. Shehu, "Probabilistic search and optimization for protein energy landscapes," in *Handbook of Computational Molecular Biology*, S. Aluru and A. Singh, Eds. Chapman & Hall/CRC Computer & Information Science Series, 2013.
14. A. E. Keating, *Methods in protein design*, ser. Methods in enzymology. Amsterdam: Academic Press/Elsevier, 2013, vol. 523.
15. B. R. Donald, *Algorithms in Structural Molecular Biology*. The MIT Press, 2011.
16. C. E. Tinberg, S. D. Khare, J. Dou, L. Doyle, J. W. Nelson, A. Schena, W. Jankowski, C. G. Kalodimos, K. Johnsson, B. L. Stoddard, and D. Baker, "Computational design of ligand-binding proteins with high affinity and selectivity," *Nature*, vol. 501, pp. 212–6, 2013.
17. B. E. Correia, J. T. Bates, R. J. Loomis, G. Baneyx, C. Carrico, J. G. Jardine, P. Rupert, C. Correnti, O. Kalyuzhniy, V. Vittal, M. J. Connell, E. Stevens, A. Schroeter, M. Chen, S. MacPherson, A. M. Serra, Y. Adachi, M. A. Holmes, Y. Li, R. E. Klevit, B. S. Graham, R. T. Wyatt, D. Baker, R. K. Strong, J. E. Crowe, P. R. Johnson, and W. R. Schief, "Proof of principle for epitope-focused vaccine design," *Nature*, 2014.
18. D. Devaurs, , T. Siméon, and J. Cortés, "Optimal path planning in complex cost spaces with sampling-based algorithms," *IEEE Transactions on Automation Science and Engineering*, vol. 13, no. 2, pp. 415–424, 2015.

19. S. LaValle and J. Kuffner, "Rapidly-exploring random trees: progress and prospects," in *Algorithmic and Computational Robotics: New Directions*, 2001, pp. 293–308.
20. S. Karaman and E. Frazzoli, "Sampling-based Algorithms for Optimal Motion Planning," *Int. J. Rob. Res.*, vol. 30, no. 7, pp. 846–894, Jun. 2011.
21. L. Janson, E. Schmerling, A. Clark, and M. Pavone, "Fast marching tree," *Int. J. Rob. Res.*, vol. 34, no. 7, pp. 883–921, 2015.
22. D. Devaurs, T. Siméon, and J. Cortés, "A multi-tree extension of the transition-based RRT: Application to ordering-and-pathfinding problems in continuous cost spaces," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2014, pp. 2991–2996.
23. N. Metropolis, A. Rosenbluth, M. Rosenbluth, A. Teller, and E.Teller, "Equation of state calculations by fast computing machines," *Journal of Chemical Physics*, vol. 21, pp. 1087–1092, 1953.
24. Y. Li, Z. Littlefield, and K. E. Bekris, "Asymptotically optimal sampling-based kinodynamic planning," *Int. J. Robot. Res.*, vol. 35, pp. 528–564, 2016.