Costmap planning in high dimensional configuration spaces

Romain Iehl, Juan Cortés, Thierry Simeon

Costmap planning in high dimensional configuration spaces

Romain Iehl[1,2], Juan Cortés[1,2], Thierry Siméon[1,2]
[1]CNRS ; LAAS ; 7 avenue du colonel Roche, F-31077 Toulouse, France
[2]Université de Toulouse; UPS, INSA, INP, ISAE ; LAAS ; F-31077 Toulouse, France
{riehl, jcortes, nic}@laas.fr

***Abstract*— For many applications, path planning algorithms are expected to compute not only feasible paths, but good-quality solutions with respect to a cost function defined over the configuration space. Although several algorithms have been proposed recently for computing good-quality paths, their practical applicability is mostly limited to low-dimensional problems. This paper extends the applicability of one of such algorithms, T-RRT, to higher-dimensional problems. To this end, we propose to introduce ideas from the ML-RRT algorithm, which can efficiently solve high-dimensional path planning problems by relying on a hierarchical partitioning of the configuration space parameters. Simulation results show the good performance of the new costmap planner, MLT-RRT, for solving problems involving up to several hundreds of degrees of freedom.**

I. Introduction

Sampling-based path planners are general, efficient, and easy to implement algorithms (see [1], [2] for a survey). These planners have been successfully applied to diverse problems in robotics and in other domains such as manufacturing, computer animation, and computational biology. In particular, variants of the Rapidly-exploring Random Tree (RRT) algorithm [3] have been shown efficient for solving path planning problems in all these domains.

Despite the many advantages of sampling-based planners, they suffer from some limitations. One of them is that the computed paths do not satisfy any optimality or quality properties, they are simply feasible paths. However, in many application domains, it can be important to compute good-quality paths with respect to a continuous cost criterion. In recent works, this notion of quality has been integrated into RRT-like planners [4], [5], [6], [7]. Nevertheless, these methods are often restricted to a specific application domain, or can be difficult to apply to high-dimensionality problems. The recently developed Transition-based RRT planner (T-RRT) [8] is aimed at generating low-cost paths according to a general cost function defined on the configuration space. The main idea behind T-RRT is to include a state transition test that avoids the expansion of the tree toward high-cost regions. The algorithm has been shown successful on a variety of moderately high-dimensional problems [8], [9]. Another limitation of sampling-based planners concerns dimensionality. Although such methods can in theory solve problems in any dimension, there are practical computational limits. In particular, the uniform sampling scheme of the standard PRM [10] and RRT [1] planners is not suited for planning in very high dimensions. Several variants have been developed, which are able to handle a much higher dimensionality in specific contexts [11], [12], [13].

Fig. 1. Multi-robot costmap planning problem : initial and goal configurations of the active arm (red) and solution path that minimizes end-effector's displacements of the two passive arms (blue) hindering the path to the goal.

This paper presents a method aimed to face these two limitations, thus enabling the computation of good-quality paths in high-dimensional problems. The proposed method extends the T-RRT algorithm by introducing ideas from the Manhattan-like RRT algorithm (ML-RRT) [11]. The principle of ML-RRT is to introduce two types of configuration parameters, labeled active and passive, and to generate their motions in a decoupled manner. The partition of the configuration parameters into active and passive parameters corresponds to their role in the planning problem. The active parameters are essential for the planning query, while the passive parameters only need to change when they hinder the motion of parts controlled by active parameters. ML-RRT has been applied to disassembly path planning [11], and in the domain of computational biology, to the study of ligand induced conformational changes in proteins [14].

The MLT-RRT algorithm proposed in this paper combines the underlying principles of T-RRT and ML-RRT at a low level. It can efficiently solve high-dimensional costmap planning problems involving multi-robot systems, for planning paths of a given set of active robots, together with the necessary motions of the passive ones considered as movable objects that possibly hinder reaching the desired goal, and while minimizing a cost function defined over the configuration space of the whole multi-robot system. Figure 1 illustrates an example of such problem solved by MLT-RRT where the objective is to compute a path for the active arm (red) while minimizing the end-effector's motions of the two other passive arms (blue).

After a brief overview of the T-RRT algorithm (Section II),

the proposed MLT-RRT extension is presented in Section III. Next, Section IV describes an improvement to the basic T-RRT algorithm that also applies to the new variant. Finally, in Section V, MLT-RRT is applied to different types of problems involving a cost function: disassembly path planning with clearance constraints, multi-robot path planning with distance and task constraints, and protein-ligand accessibility problems. The performance of the method is analyzed and compared to the basic T-RRT and ML-RRT algorithms, as well as to the RRT* planner [7].

II. Basic T-RRT algorithm

T-RRT [8] extends the Rapidly-exploring Random Tree (RRT) algorithm by incorporating a stochastic transition test that is used to accept or reject new expansions of the search tree, based on the cost variation associated with the expansion. The pseudo-code of T-RRT (Algorithm 1) is similar to that of RRT, with the addition of the `TransitionTest` and `MinExpandControl` functions.

The `TransitionTest` function is based on the Metropolis criterion from Monte Carlo algorithms [15]. Its role is to bias the exploration towards low-cost regions. This function is described in Algorithm 2, the principle is as follows. When performing an expansion from a configuration q_i to a configuration q_j, `TransitionTest` is called with the following arguments: the costs c_i and c_j of the configurations, and the distance d_{ij} between them. If the expansion is downhill (i.e. $c_j < c_i$), it is always accepted. However if it is an uphill expansion, it is accepted with a probability p that decreases exponentially with the slope $(c_j - c_i)/d_{ij}$. This probability is further modified by an adaptive *temperature* parameter, T, that changes along with the iterations: after each accepted uphill extension, T is divided by α to avoid over-exploring higher cost regions; while after $nFail_{max}$ rejections, T is multiplied by α, to allow for easier exploration. This automatic tuning allows T-RRT to balance the exploration between the Voronoi bias of RRT and a bias towards low-cost regions. The $nFail_{max}$ parameter determines the trade-off between computation time and quality of the path. As explained in [8], it can be set a low value (e.g. 10) for a greedy search, or a high value (e.g. 100) for a computationally more intensive but better quality exploration.

The role of the `MinExpandControl` function (see [8]) is to avoid an undesirable behavior of the algorithm, which can spend a significant time over-exploring easy regions of the costspace. This will be further explained in Section IV, along with an improvement introduced in this work.

III. MLT-RRT

MLT-RRT extends the applicability of the T-RRT algorithm to higher-dimensional problems by introducing ideas from the Manhattan-like RRT (ML-RRT) variant [14]. ML-RRT relies on a decomposition of the configuration space parameters into two sets, labeled active and passive: $\mathcal{C} = \mathcal{C}^{act} \times \mathcal{C}^{pas}$. The mobile parts of the system controlled by active or passive parameters are called active parts or passive parts, respectively. The active parameters are treated directly

Algorithm 1: Transition-based RRT

```
input :
        the configuration space C;
        the cost function c : C → ℜ;
        the root q_init and the goal q_goal;
output: the tree T;
begin
    T ← InitTree(q_init);
    while not StopCondition(T, q_goal) do
        q_rand ← SampleConf(C) ;
        q_near ← NearestNeighbor(q_rand, T);
        q_new ← Extend(T, q_rand, q_near);
        if q_new ≠ NULL
        and
        TransitionTest(c(q_near), c(q_new), d_near−new)
        and MinExpandControl(T, q_near, q_rand) then
            AddNewNode(T,q_new);
            AddNewEdge(T,q_near, q_new);
```

Algorithm 2: TransitionTest(c_i, c_j, d_{ij})

```
begin
    nFail = GetCurrentNFail();
    if c_j < c_i then
        return True;
    p = exp(−(c_j − c_i)/d_ij / T);
    if Rand(0,1) < p then
        T = T/α;
        nFail = 0;
        return True;
    else
        if nFail > nFail_max then
            T = T * α;
            nFail = 0;
        else
            nFail = nFail + 1;
        return False;
```

at each iteration of the exploration process, while the passive parameters are treated only when their associated passive parts hinder the movement of active parts.

In the proposed algorithm, sketched in Algorithm 3, the underlying principles of T-RRT and ML-RRT are combined at a low level. The algorithm, like ML-RRT, explores the active and passive parameters of the configuration space in a decoupled manner. However, passive parameters to be moved are not only selected based on collisions between moving parts, but also on their importance within the cost function.

The passive part selection is carried out within the `ExpandAndSelect` function, described in Algorithm 4. This function performs a standard RRT expansion but when a collision is encountered along the path from q_{near} to q_{rand}, `SelectColliding` identifies the list of passive parameters L_c^{pas} involved in the collision (see Fig. 2). Similarly, the uphill expansions rejected by the `TransitionTest` function because of a too high cost increase, are also checked by the `SelectByCost` function for a cost-based selection of

Algorithm 3: Construct_MLT-RRT

input :
the configuration space C;
the root q_{init} and the goal q_{goal};
the partition $\{L^{act}, L^{pas}\}$;
output: the tree $\mathcal{T}$;
begin
  $\mathcal{T} \leftarrow$ `InitTree`(q_{init});
  while not `StopCondition`$(\mathcal{T}, q_{goal})$ **do**
    `// Active expansion`
    $q^{act}_{rand} \leftarrow$ `SampleConf`$(\mathcal{C}, L^{act})$;
    $q_{near} \leftarrow$ `BestNeighbor`$(q_{rand}, \mathcal{T})$;
    $(q_{new}, L_c^{pas'}) \leftarrow$ `ExpandAndSelect`$(q_{near}, q^{act}_{rand})$;
    if $q_{new} \neq null$ **then**
      `AddNewNode`$(\mathcal{T}, q_{new})$;
      `AddNewEdge`$(\mathcal{T}, q_{near}, q_{new})$;
    `// Passive expansion`
    while $L_c^{pas} \neq \emptyset$ **do**
      $q^{pas}_{rand} \leftarrow$ `PerturbConf`(q_{near}, L_c^{pas});
      $(q_{new}, L_c^{pas'}) \leftarrow$ `ExpandAndSelect`$(q_{near}, q^{pas}_{rand})$;
      if $q_{new} \neq null$ **then**
        `AddNewNode`$(\mathcal{T}, q_{new})$;
        `AddNewEdge`$(\mathcal{T}, q_{near}, q_{new})$;
      $L_c^{pas} \leftarrow L_c^{pas'} \setminus L_c^{pas}$;

Algorithm 4: ExpandAndSelect(q_{near}, q_{rand})

begin
  $q_{new} \leftarrow$ `Expand`(q_{near}, q_{rand});
  if $q_{new} \notin C_{free}$ **then**
    $L_c^{pas} \leftarrow$ `SelectColliding`(q_{new});
    return $(Null, L_c^{pas})$
  else if `TransitionTest`$(q_{near}, q_{new}) = false$ **then**
    $L_c^{pas} \leftarrow$ `SelectByCost`$(q_{near}, q_{new}) \cap L^{pas}$;
    return $(Null, L_c^{pas})$;
  else
    return $(q_{new}, \emptyset)$

the passive parameters L_c^{pas} that significantly contribute to the cost increase. When `ExpandAndSelect` returns a non empty L_c^{pas} list of (blocking or high-cost) passive parts, a new expansion attempt is performed in order to move the identified parts. The manhattan-like expansion mechanism is iterated until a successfull expansion is performed or a maximum number of steps is reached.

Cost-based selection: The role of `SelectByCost` is to identify the list of passive parameters (if any) that are involved in the cost increment between the configurations q_{near} and q_{rand}. This requires, for a given configuration q and a cost function f, to accurately determine the individual cost contribution of parameters subsets q^k associated to the active subchain L^{act} and to the set $L^{pas,k}$ of kinematically independant subchains formed by the passive parts. This leads us to consider a decomposition of the cost function as a sum of elementary terms: $f(q) = \sum f_k(q^k)$, where each term f_k assigns a cost to the q^k subset of configuration

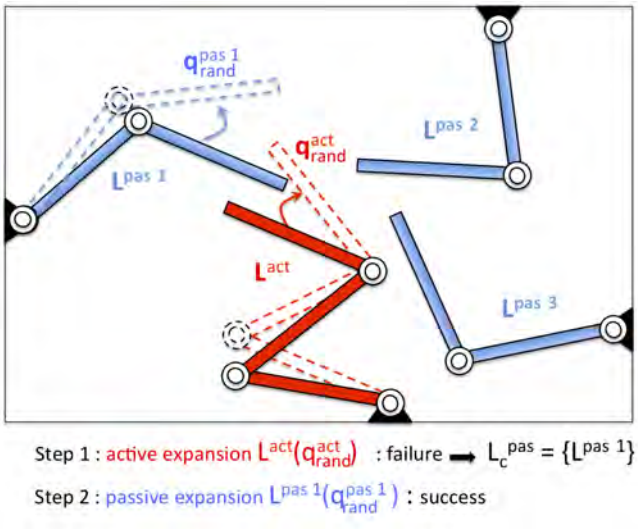

Fig. 2. Manhattan-like exploration of the active (red) and passive (blue) configuration parameters : only the passive parameters of parts hindering the motion of the active parameters are considered

space parameters. Each f_k function is itself decomposed into a term depending only on q^k and a sum of pairwise costs also depending on the other parameter subsets q^l ($l \neq k$). Thus, each elementary cost is expressed as : $f_k(q^k) = f'_k(q^k) + \sum_l f''_{k,l}(q^k, q^l)$. Using this decomposition, the `SelectByCost` function described in Algorithm 5 returns the list of all parameters involved in terms that contribute more than a minimum $threshold$ value to the cost increment. Empirically, consistently good results have been obtained by using $threshold = 0.1 * \max_{k=1}^{n}(f_k(q^k_{rand}) - f_k(q^k_{near}))$, thus limiting the selection to parameters with a significant contribution, relatively to the most contributing term.

Discussion on the cost function decomposition: While T-RRT operates on an arbitrary, user defined cost function $f(q)$, the need for a decomposition as a sum of simpler terms imposes an additional constraint. However, this is not a restrictive assumption since cost functions can generally be defined in such a way that the configuration parameters can be decoupled into subsets with pairwise contributions to the cost. For example, if we consider a clearance-based cost to maximize the distance between the active and passive parts of a multi-robot system, such a function depends on the pairwise distance between parts. Similarly, in a molecular context, the energy of the molecular system can be defined as a sum of pairwise atomic interactions. Consequently, a simple decomposition is readily available in these examples. Note that in situations for which such decomposition is not possible, the elementary cost contributions of q^k subsets could also be approximated using the jacobian $\delta f / \delta q$.

IV. Improved Temperature Tuning

This section presents an improvement made to the adaptive temperature tuning method of T-RRT (see Algorithm 2), and which also benefits to MLT-RRT. The motivation is to provide a better way to balance exploration versus refinement during the tree construction.

Algorithm 5: SelectByCost(q_i, q_j)

```
begin
    L_cost ← ∅;
    for (k in 1..n) do
        if f_k(q_j^k) − f_k(q_i^k) > threshold then
            L_cost ← L_cost ∪ (q^k)
return L_cost
```

Exploration versus Refinement: As explained in [8], the adaptive temperature tuning in T-RRT may yield an undesirable behavior of the algorithm in some situations. Indeed, when the tree has already covered a low cost region, and expansion toward other favorable region requires to go through higher cost configurations, most of new nodes only contribute to refine the covered portion of the space. Such a slowing down of the T-RRT exploration is due to a stabilization of the temperature parameter.

In [8], the method for ensuring a minimal exploration rate, implemented in the `MinExpandControl` function, uses a rejection criterion that is based on the distance between the sampled configurations q_{rand}, and their nearest neighbor q_{near}: when this distance is less than the expansion step size δ, the new node from the expansion of q_{near} is considered to be a refinement node. If half of the nodes in the tree are refinement nodes, the addition of new refinement nodes is rejected. This rejection criterion was shown to improve the performance of T-RRT. However, this method does not scale well to higher dimensions, due to the low probability of sampling within the neighborhood of an existing node. The following paragraph presents a modification that provides a better control of the tree expansion.

Cost-dependent temperature tuning: The proposed improvement of T-RRT is to change the temperature reduction strategy in the transition test by replacing the line $T = T/\alpha$ in Algorithm 2 by : $T = T/(\alpha^{(c_j - c_i)/normalization\_value})$, where $normalization\_value$ is a parameter explained below. With this change, the reduction of the temperature after a successful uphill expansion depends on the local steepness of the costmap. An *easy* transition, which will contribute slightly to the overall cost of the solution path, will have a small effect on the temperature, and therefore, on the speed of the exploration process. However, when accepting a *steep* transition, the temperature parameter is reduced significantly, ensuring that no other steep transition will be accepted without first performing an extensive search in the neighborhood of the search tree. The effect of this modification is illustrated in Figure 3. This modification provides an implicit expansion control mechanism and a significant speed-up of the exploration process. Note that the proper setting of the $normalization\_value$ parameter is important. A too low value will result in an unnecessarily aggressive temperature reduction, which will lead to poor performance. Contrarily, a too high value will yield rapid exploration but degraded quality of the solution path. Setting $normalization\_value$ to one tenth of the cost variations along the path has been found empirically to yield good results in general. However, such variations cannot be known beforehand. Thus, the strategy used is to initialize $normalization\_value$ to a small, conservative value, and to reset it at each iteration to $0.1 * cost\_delta$, where $cost\_delta$ is the current cost difference between the configuration with the highest cost and the configuration with the lowest cost in the search tree. The efficiency of this new temperature tuning method is shown in the results presented in Section V.

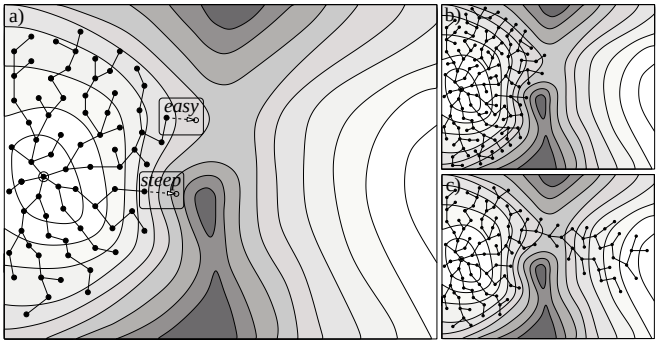

Fig. 3. Illustration of the effect of the new temperature tuning method on the behavior of T-RRT. a) Intermediate stage of the T-RRT construction. The tree is covering a basin of the costmap. b) With the original temperature tuning strategy of T-RRT, many iterations are nedded to escape the bassin since every uphill expansion implies a constant temperature reduction. The tree will tend to cover very densely the basin before finding the saddle area. c) With the new strategy, the temperature decrease after an *easy* transition is minor compared to a *steep* transition, which favors exploration of the saddle area versus refinement in the basin.

V. Results

The MLT-RRT algorithm has been implemented in the path planning software Move3D [17]. The algorithm is applied to two motion planning problems issued from robotics (Figs. 4 and 1), and to a third example that showcases the application in the domain of computational biology (Fig. 8). On the first two examples, MLT-RRT is compared, in terms of computation time and quality of the solution paths, to RRT, ML-RRT and T-RRT (Sections V-A, V-B) as well as to RRT* [7] (Section V-C). In all cases, MLT-RRT (and T-RRT) runs were performed with the $nFail_{max}$ parameter set to 100 to favor high-quality of the solution paths over performance in computation time.

A. Academic disassembly problem

This set of examples from [11] is useful for empirical performance analysis. The first example, 2d-simple (Fig. 4.a), represents a simple mobile object to be extracted from an articulated object formed by a static body and 3 mobile sticks (7 dofs in total). The second and third examples, 2d-medium (Fig. 4.b) and 2d-hard (Fig. 4.c), are more difficult versions involving a longer static object with 6 and 12 mobile sticks (10 and 16 dofs) respectively. The mobile object is considered active, while the movable sticks are treated passively. The cost function, aimed to maximize clearance, is defined as $1/d_{min}$, where d_{min} is the minimal distance between the active object, and the static and passive parts.

	2d-simple (7 dofs)			2d-medium (10 dofs)			2d-hard (16 dofs)		
	nb of nodes	time (seconds)	clearance	nb of nodes	time (seconds)	clearance	nb of nodes	time (seconds)	clearance
RRT	1742	10.57	0	—	—	—	—	—	—
ML-RRT	294	0.29	0	609	1.2	0	1070	2.58	0
T-RRT	5524	703	0.2	—	—	—	—	—	—
MLT-RRT	311	0.77	0.45	1077	5.8	0.4	1639	22.4	0.4

TABLE I

DISASSEMBLY: NUMERICAL RESULTS

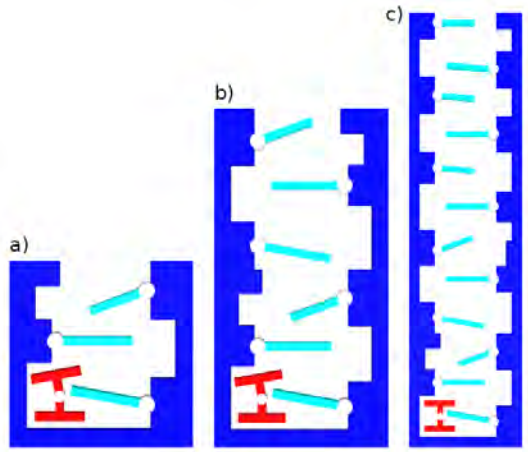

Fig. 4. Three variants (single, medium, hard) of the 2d disassembly problem

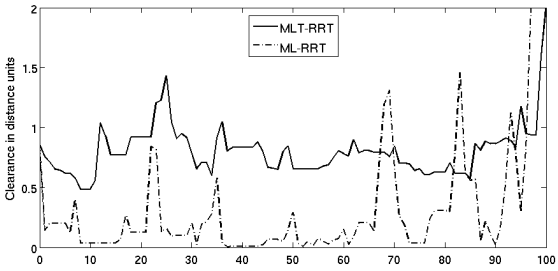

Fig. 5. Clearance along the solution path for the 2d-medium problem.

Table I shows the simulation results (averaged over 5 runs of the algorithms) for this set of problems. The table contains the number of nodes in the search tree, the computation time (in seconds), and the minimum clearance obtained along the solution path, in internal distance units. Fig. 5 compares the clearance along two paths for the 2d-medium problem, computed with the ML-RRT and the MLT-RRT algorithms.

The RRT and ML-RRT planners show better performance in terms of computation time than their costmap planning counterparts, T-RRT and MLT-RRT. This is expected, since they are solving a much easier collision-free planning problem, with no concern for the cost function. Note however that the computation time ratios between ML-RRT and MLT-RRT are very reasonable: about 1:3 on 2d-simple, up to 1:9 on the 2d-hard problem. The results for the RRT and ML-RRT algorithms are consistent with those presented in [11]. The basic RRT planner failed to solve the 2d-medium and 2d-hard problems after several hours of computation. It can solve the 2d-simple problem in reasonable time, but still required an extensive search tree (1742 nodes). ML-RRT displays much better performance and scalability with the increase in difficulty over the set of problems, thanks to the decoupling of motions.

Results also show the higher performance of MLT-RRT compared to the original T-RRT costmap planner that could only solve the 2d-simple case and required a significant computation time (703 seconds). The MLT-RRT algorithm solves all 3 examples in a short time (1.2, 5.8 and 22.4 seconds) and results show its good scalability to high dimensional problems. Additionally, for the 2d-simple example, the clearance of the MLT-RRT solution path is higher than for T-RRT. This can be explained by the fact that MLT-RRT performs a more exhaustive sampling of the relevant parameters, and thus find more favorable configurations.

B. Robotic arms

The second test exemple is the multi-arm planning scenario of Figure 1, where an active robotic arm, in red, evolves in an environment obstructed by an obstacle and two other robotic arms, considered passive, in blue. Each arm has 7 dofs, for a total of 21 dofs. The objective is to solve the planning query involving the active arm, while minimizing the movement of the end effectors of the other arms. However, it is geometrically impossible to extract the active arm without moving the passive arms significantly. To this end, we use a cost function that is a composite of a clearance term, representing a safety distance between the robotic arms, and a penalty proportional to the displacement of the end effectors of the passive arms. The Table II shows the tree size and computation time for T-RRT and MLT-RRT, averaged over 5 planning queries, and Fig. 6 shows the cost along a solution path. The computation times between both algorithms are similar, slightly in favor of MLT-RRT. However, the quality of the generated paths is significantly better when using the MLT-RRT algorithm. This is due to the nature of the planning problem: on this example, it is relatively easy to find a geometric solution, where the three arms move in a mostly upwards movement, allowing extraction of the active arm. However, the cost function adds an important constraint by restricting the movement of the

	robotic arms	
	nb of nodes	time (seconds)
T-RRT	1390	27.7
MLT-RRT	733	19.7

TABLE II

ROBOTIC ARMS: NUMERICAL RESULTS

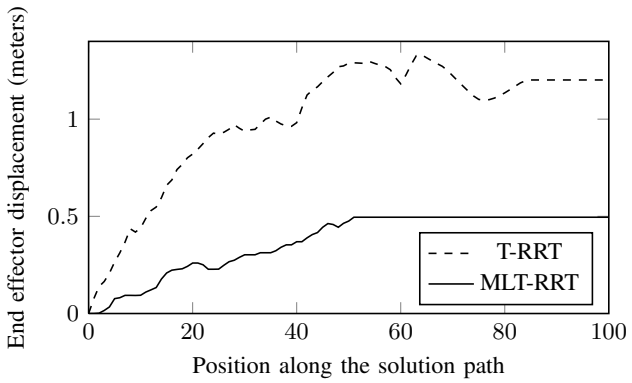

Fig. 6. Robotic arms: end effectors displacement.

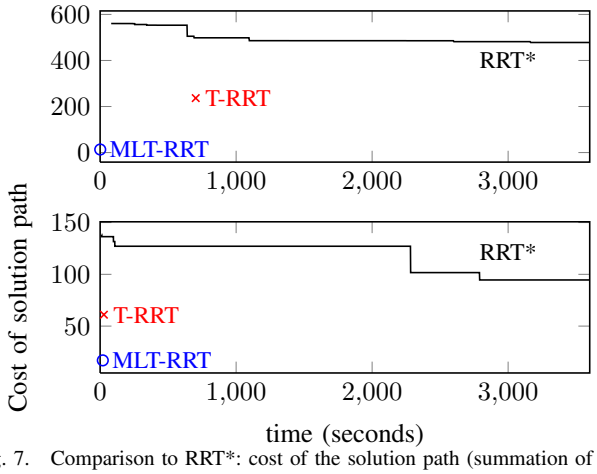

Fig. 7. Comparison to RRT*: cost of the solution path (summation of the configuration costs along the path) as a function of time, on the 2d simple example (top) and the robotic arms example (bottom).

end effectors. Consequently, the decoupled motion of parts does not provide a significant benefit in a geometrical sense, however, it is necessary in order to find low cost paths. This is illustrated in Fig. 6, that shows the sum of the end effector displacements for the two passive arms, along the solution path. The maximum value for the MLT-RRT path corresponds to a total displacement of 49cm, whereas the T-RRT path contains an important displacement of 1m33cm.

C. Comparison with RRT*

RRT* [7] is a recently published RRT variant with asymptotic optimality guarantees. RRT* builds an exploration tree in a similar fashion to RRT, with the main difference that the tree is incrementally rewired to improve the quality of the paths originating from the start configuration. Results presented in related work (as well as our own experiments) show that RRT* quickly converges towards the optimal solution when applied to low-dimensional problems. However, the results presented below show that the method performs poorly on higher-dimensional and constrained problems, such as the ones addressed in this paper. Indeed, optimizing the initial path becomes a very expensive process.

Fig. 7 shows the quality and computation time of the solution paths computed by RRT*, T-RRT and MLT-RRT on the 2d-simple and the robotic arms examples. In agreement with its incremental nature, the results of RRT* are represented as a function of time : the black line represents the quality of the best solution path so far computed by RRT*. The solutions found by T-RRT and MLT-RRT are represented by a dot. As can be seen in the figure, for both exemples, RRT* quickly finds an initial solution, but is unable to generate better quality solutions than both T-RRT and MLT-RRT, even after many iterations. Note however, that RRT* may also benefit from the introduction of some level of biasing to speed up the expansion of the search tree as in MLT-RRT, but at the price of loosing its guarantee of asymptotically optimal solutions.

D. Ligand-protein interactions

The last example presents MLT-RRT applied to the domain of computational biology. As shown in previous work [14], path-planning algorithms operating on mechanistic molecular models can be applied as efficient methods to compute the access/exit pathway of a ligand to the active site in a protein, thus providing important information to understand protein-ligand interactions. We applied T-RRT and MLT-RRT to the ligand exit problem illustrated in Fig. 8. The ligand has 12 degrees of freedom (6 for the pose, and 6 internal torsion parameters). For MLT-RRT, the protein has been modeled with full sidechain flexibility, for a total of 627 torsional joints. For T-RRT, only the flexibility of the ligand is taken into account, as the algorithm cannot handle the additional dimensionality from the flexible sidechains (considered as passive chains in MLT-RRT). Fig. 9 shows the benefit of the additional flexibility: the energy of the path computed by MLT-RRT is significantly improved, compared to that of T-RRT which has high energy barriers corresponding to unlikely conformations. For MLT-RRT, the possible rearrangements of the sidechains allows for the computation of a more realistic motion. Remarkably, the computation times are of the same order: about 3 minutes and 10 minutes for T-RRT and MLT-RRT respectively, despite the increase in dimensionality.

E. Effect of the cost-dependent temperature reduction

All the results above were obtained using the modification to the transition test presented in Section IV. For comparison purposes, Table III shows results on the various preceding problems, when this improvement is not used. The table shows that the proposed cost-dependent temperature tuning yields a notably improved performance, with speed-ups ranging from 3 to 5 on the 2d-medium, 2d-hard and robotic arms problems and negligible impact on the quality of the solution (path costs, not reported in the table, remain almost the same for the three problems). The effect is particularly important on the ligand exit example. When using the modified transition test, MLT-RRT found a solution path in only 533 seconds, but could not find one after 5 hours without the change. Indeed, the ligand exit example is the one that suffers the most from the exploration versus refinement problem. The volume of reachable space is large,

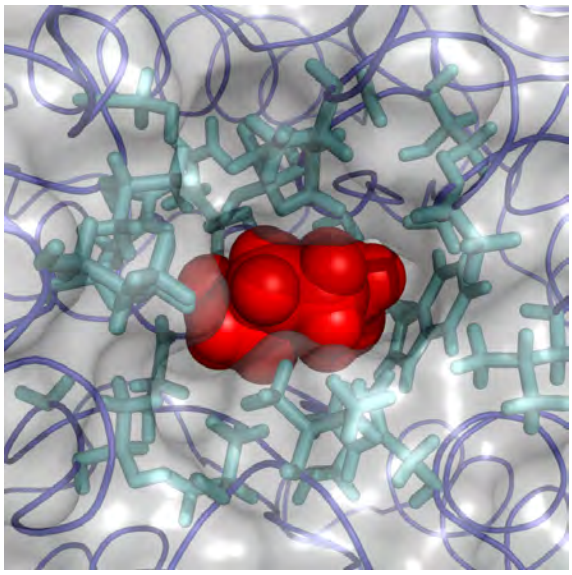

Fig. 8. Protein-ligand exit problem. The light blue elements are some of the flexible sidechains, handled passively by MLT-RRT. The problem involves 627 dofs.

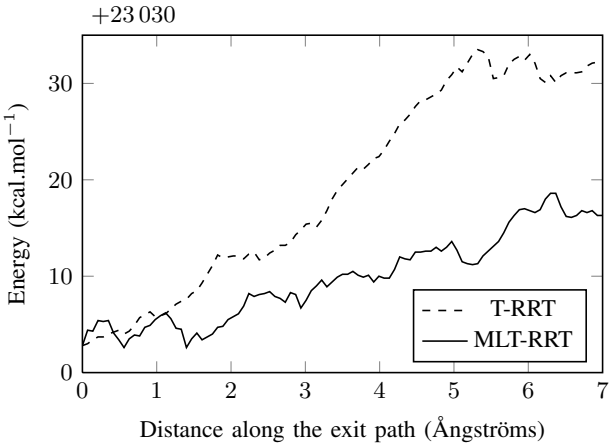

Fig. 9. Energy of ligand exit paths computed with T-RRT and MLT-RRT.

and the costspace has a complicated topology, with a very high number of small local minima in which the algorithm can spend a long time refining the space.

VI. Conclusion and future works

This paper has presented a new algorithm, called MLT-RRT, which extends T-RRT to high-dimensional problems by introducing the principle of decoupled motions from ML-RRT. Experiments on several types of problems, both in the contexts of robotics and computational biology, show the good performance of the algorithm.

A possible improvement of the method will consist in introducing multiple levels of passivity associated with different parts of the system according to their properties. This principle was already introduced in the context of computational biology to simulate protein motions induced by protein-ligand interactions [14], but needs to be generalized.

As future work, we also plan to further investigate the application of MLT-RRT to human-aware path planning in the presence of Human-Robot Interaction constraints. This application domain involves possibly highly-articulated multi-arm robots like humanoid torso and the presence of the human requires to account for various comfort and safety criteria [18].

	2d-medium	2d-hard	robotic arms	ligand exit
with	5.8s	22.4s	19.7s	533.2s
without	16.1s	50.4s	112.5s	>5h

TABLE III

Effect of the temperature reduction modification on the computation time, in seconds.

References

[1] S. LaValle and J. Kuffner, "Rapidly-exploring random trees : Progress and prospects," in *Algorithmic and Computational Robotics: New Directions*, B. Donald, K. Lynch, and D. Rus, Eds. Boston: A.K. Peters, 2001, pp. 293–308.

[2] H. Choset, K. Lynch, S. Hutchinson, G. Kantor, W. Burgard, L. Kavraki, and S. Thrun, *Principles of Robot Motion: Theory, Algorithms, and Implementations*. Cambridge: MIT Press, 2005.

[3] J. Kuffner and S. LaValle, "Rrt-connect: An efficient approach to single-query path planning," *Proc. IEEE Int. Conf. on Robotics and Automation*, pp. 995–1001, 2000.

[4] A. Ettlin and H. Bleuler, "Randomised rough-terrain robot motion planning," *Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, pp. 5798–5803, 2006.

[5] J. Lee, C. Pippin, and T. Balch, "Cost based planning with rrt in outdoor environments," *Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, pp. 684–689, 2008.

[6] C. Urmson and R. Simmons, "Approaches for heuristically biasing RRT growth," *Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, pp. 1178–1183, 2003.

[7] S. Karaman and E. Frazzoli, "Incremental sampling-based algorithms for optimal motion planning," *Int. Journal of Robotics Research*, 2010, submitted.

[8] L. Jaillet, J. Cortés, and T. Siméon, "Sampling-based path planning on configuration-space costmaps," *IEEE Transactions on Robotics*, vol. 26, 2010.

[9] D. Berenson, T. Siméon, and S. Srinivasa, "Addressing cost-space chasms for manipulation planning," *IEEE International Conference on Robotics and Automation*, 2011.

[10] L. 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(4), pp. 566–580, 1996.

[11] J. Cortés, L. Jaillet, and T. Siméon, "Disassembly path planning for complex articulated objects," *IEEE Transactions on Robotics and Automation*, pp. 475–481, 2008.

[12] A. Shkolnik and R. Tedrake, "Path planning in 1000+ dimensions using a task-space voronoi bias," *IEEE International Conference on Robotics and Automation*, 2009.

[13] X. Tang, S. L. Thomas, P. Coleman, and N. M. Amato, "Reachable distance space: Efficient sampling-based planning for spatially constrained systems," *The international journal of robotics research*, 2010.

[14] J. Cortés, D. T. Le, R. Iehl, and T. Siméon, "ligand-induced conformational changes in proteins," *Physical Chemistry Chemical Physics*, vol. 29, 2010.

[15] D. Frenkel and B. Smit, *Understanding Molecular Simulation: From Algorithms to Applications*. San Diego: Academic Press, 2002.

[16] A. Yershova, L. Jaillet, T. Siméon, and S. LaValle, "Dynamic-domain RRTs: Efficient exploration by controlling the sampling domain," *Proc. IEEE Int. Conf. on Robotics and Automation*, pp. 3867–3872, 2005.

[17] T. Siméon, J.-P. Laumond, and F. Lamiraux, "Move3D: A generic platform for path planning," *Proc. IEEE Int. Symp. on Assembly & Task Planning*, pp. 25–30, 2001.

[18] J. Mainprice, E. Sisbot, T. Siméon, and R. Alami, "Planning Human-aware motions using a sampling-based costmap planner," in *IEEE Int. Conf. Robot. And Autom.*, 2011.