Integrating Multiple Soft Constraints for Planning Practical Paths

Jing Yang, Patrick Dymond and Michael Jenkin

Abstract— **Sampling-based algorithms are a common approach to high-dimensional real-world path planning problems. Unfortunately the solutions found using such planners are often not practical in that they do not take into account soft application-specific constraints. This paper formulates the practicality of paths based on the notion of soft constraints found in the Planning Domain Definition Language 3 (PDDL3) [21] and a range of optimization strategies are developed targeted towards user-preferred qualities by integrating soft constraints in the pre-processing, planning and post-processing phases of the sampling-based path planners. An auction-based resource allocation approach coordinates competing optimization strategies. This approach uses an adaptive bidding strategy for each optimizer and in each round the optimizer with the best predicted performance is selected. This general coordination system allows for flexibility in both the number and types of the optimizers used. Experimental validation demonstrates the effectiveness of the approach.**

I. INTRODUCTION

Path planning is a fundamental problem for nearly all the robotic systems. The basic robot path planning problem involves finding a path for a robot to get from 'here' to 'there' while avoiding obstacles in a static environment. It has been proven that the basic path planning problem is PSPACE-complete in the dimensionality of the degrees of freedom (DOFs) possessed by the robot [1]. Since it can be difficult to plan a path for robots with many DOFs, early methods for high DOF robots aimed at finding any solution to the planning problem within a reasonable time. One shortcoming of basic sampling-based planning approaches is that they can obtain highly 'non-optimal' solutions since they rely upon randomization to explore the search space. Although basic sampling-based planning algorithms may find a valid solution, that solution may not be practical in that it does not meet soft constraints that exist within the problem domain.

The need to properly represent and use soft constraints is particularly important for redundant DOF robots such as tentacle devices (see Fig. 1). For these devices the high number of DOFs provide the capability to deal with complex environments and to produce solutions that are not only correct but also optimize other requirements of the problem space. The high number of DOF's coupled with the stochastic nature of the planning algorithm often leads to motion paths that involve the robot "flailing about" as it gets from the start to the goal state. One way of reducing these unwanted motions and taking soft constraints into account is to use an appropriate controller that takes the path identified by the path planner as input and only integrates the soft constraints while executing the path [2], [3]. There are many issues with this approach. Perhaps most critically the paths produced may be infeasible for a real robot. For example, following the path produced may require that the robot move extremely slowly in order to minimize the influence of dynamics and other physical constraints. These controllers are also system specific, and it can be very hard to develop a good 'general' controllers or to know which controller to use for a given task.

Rather than incorporating soft constraints as a secondary "refining" process a more general approach is to augment sampling-based path planning with mechanisms that generate paths that are not only correct but that also optimize the soft constraints. Such augmentation can take place at different points in the path planning process, including the pre-processing (i.e. sampling), planning, and post-processing phases of the sampling-based path planners. Given a range of optimization strategies it is difficult to determine which optimizer is most suitable at a certain time for a certain problem. This work describes an auction-based approach that enables multiple optimizers to compete in a 'market' for computational resources during randomized path planning. The optimizer with the best predicted performance is selected in each optimization round.

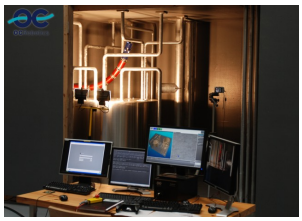
II. RELATED WORK

A. Sampling-based Path Planning

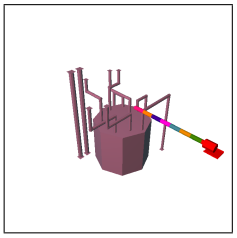
Sampling-based planners generate a number of discrete sample points in configuration space and test motions between these points and the start and goal states. Such planners usually represent motions as a graph as in the PRM [4], or as a tree as in the RRT [5]. Early variants of sampling-based planners provided no performance guarantee. More recently Karaman and Frazzoli [6] proposed the sampling-based algorithms PRM*, RRG and RRT*, which always converge to an optimal solution that minimizes the length of the path. PRM* is a variant of PRM with a variable connection radius that scales as the number of samples, while standard PRM uses a constant radius value to select neighbors to connect. RRG and RRT* incrementally build a connected roadmap, augmenting the RRT algorithm with connections within a sphere scaling the radius with the number of samples. RRT* has been shown to return significantly shorter paths than RRT given a specified number of samples when planning.

Sampling-based path planners that address the problem of path quality can be divided into three broad categories based on where these issues are integrated within the algorithm: pre-processing, post-processing, and customized learning.

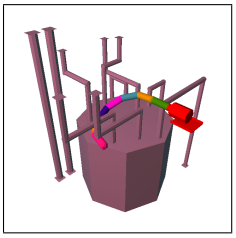
J. Yang, P. Dymond and M. Jenkin are with Department of Electrical Engineering and Computer Science, York University, Toronto, Canada {jyang, dymond, jenkin}@cse.yorku.ca



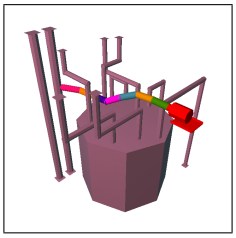
(a) Tentacle robot in hotcell



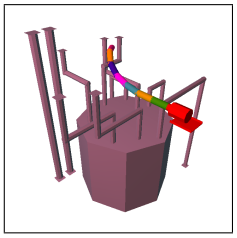
(b) c_1



(c) c_2



(d) c_3



(e) c_4

Fig. 1: (a) Tentacle robot in a hotcell in a mock environment (©OC Robotics and used with permission). (b-e) Four configurations of the tentacle robot considered in the experiments reported here, labelled as c_1, c_2, c_3 and c_4.

a) Pre-processing approaches.: Pre-processing approaches consider the specific preferences of desired paths in the pre-processing phase, i.e. during the roadmap construction phase before a query is made. Because of its probabilistic nature, the PRM roadmap often contains nodes and edges that lack practical usage or are redundant. Aiming at finding shorter paths with higher clearance, Nieuwenhuisen and Overmars [7] add nodes and edges to create "useful" cycles, which provide short paths and alternative paths in different homotopy classes. Based on this work, another PRM variant by Geraerts [8] retracts nodes and edges to the medial axis to generate high clearance paths.

b) Post-processing approaches.: Given a path found by the sampling-based path planner, post-processing approaches modify the path in accordance with the required practicality preference by adding new nodes, smoothing the path, eliminating unnecessary loops or detours, etc. Path pruning and shortcut heuristics are common post-processing techniques for creating shorter and smoother paths [9], [8]. Retraction algorithms add clearance to a given path [10]. Post-processing algorithms may take multiple paths as input rather than just a single one. For example, the path merging algorithm described in [11] computes a path with improved quality by hybridizing high-quality sub-paths. The algorithm considers the generalized formulation of path quality measures rather than specific requirements. Path refining approaches, mentioned earlier, fall into the post-processing category.

c) Customized learning.: Although post-processing algorithms have shown some success in improving the path quality and can be used by all the path planners, the final path depends on the original paths, i.e. they cannot find alternative routes that deviate considerably from the original ones. To address this problem, customized learning algorithms integrate the requirement for path quality in the learning phase. For example, Kim et al. [12] use an augmented version of Dijkstra's algorithm to extract a path from a roadmap on criteria other than path length. The approach of initially finding an approximate solution is utilized by the Fuzzy PRM [13], Lazy PRM [14], IRC (Iterative Relaxation of Constraints) [15] and C-PRM (Customizable PRM) [16] algorithms where the roadmap nodes and edges are not validated, or are only partially validated, during roadmap construction. During the learning phase, the path is searched by strengthening the constraints (obstacle collision, path length or other specified preferences) iteratively. These methods are designed to decrease the roadmap construction costs, while only slightly increasing the query costs.

B. Soft Constraints

Many of the problems with planning practical paths are related to having to deal with both hard and soft constraints within a common framework. In path planning the hard constraint is the need to obtain a collision free path and must be satisfied by any solution to the problem. The associated soft constraints include the desire to obtain paths with good clearance, short length and the like. Soft constraints may be violated by a solution but serve as guides to *encourage* or *influence* the planner to find good solutions. The integration of soft constraints in robot path planning is still limited. One common way of taking soft constraints into account is to use an appropriate controller that takes the path identified by the path planner as input and while implementing the desired path optimizes these other soft constraints (see [2], [3]). In [17] cost functions used by the controller are formulated by summing together the weighted soft constraints, which address problems associated with singularities, joint velocity demands, joint limits and collision with workspace obstacles. To optimize the given soft constraints, the Gradient Projection Method (GPM) modifies the manipulator configuration by controlling the amount of self-motion added to the joint velocity vector.

Many formalisms for describing soft constraints have been proposed in the literature (e.g., [18], [19], [20], [21]). The PDDL3 formalism is a commonly used representation because of its simplicity and generality. In [22] PDDL3 was applied to the representation of soft constraints within a motion planning framework. Using PDDL3, soft constraints are encoded into a cost function which the planner tries to minimize as it plans. This formalism is capable of representing complex preferences over planning trajectories and therefore can be applied to a broad class of problems.

C. Resource allocation

In sampling-based path planning, the problem of optimizing soft constraints involves optimizing over a collection of possible constraints at various stages in the computation.

Given a set of optimization methods associated with different optimization costs, it is necessary to determine how to distribute a constrained optimization budget among them.

Automated resource allocation is a key problem in the area of multiagent systems, in which participating agents interact in both allocating the resources each agent's activities require and in evaluating the results from these activities [23]. Because each agent has limited competence and awareness of the decisions produced by others, some sort of coordination is required to maximize the performance of the overall system. One common solution to resource allocation among multiple agents is to apply well known results and insights from auction theory (e.g., [24], [25]) to represent the task and its solution. In an auction, a set of items is offered by an auctioneer in an announcement phase, and the participants can make an offer for these items by submitting bids to the auctioneer. Once all bids are received or a prespecified deadline has passed, the auction is then cleared in the winner determination phase by the auctioneer who decides which items to award and to whom. In practical applications, the items for sale are typically tasks, roles, or resources. The bid prices reflect each agent's costs or utilities associated with completing a task, satisfying a role, or utilizing a resource [26].

III. PATH PLANNING WITH SOFT CONSTRAINTS

Informally speaking, given a robot with a description of its kinematics and dynamics, a description of the environment, an initial state, and a goal state, a solution to the path planning with soft constraints problem seeks to find a sequence of control inputs so as to drive the robot from its initial state to a goal state while obeying the hard constraints, e.g. not colliding with the surrounding obstacles and staying within joint limits, while optimizing the soft constraints, e.g. maintaining appropriate distance from the obstacles, minimizing rotations of joints, minimizing the path length and so on. This more general version of the problem is formalized below, building upon the traditional definition of path planning and PDDL3 to represent the soft constraints. See [22] for more details on the approach.

Following PDDL3 [21], the syntax for soft constraints includes two parts: (i) the identification of the soft constraints; and (ii) a description of how the satisfaction or level of satisfaction affects the quality of the resulting path. Each soft constraint is associated with a violation penalty weight such that paths that satisfy different subsets of soft constraints can be compared.

The core of sampling-based path planners involves searching for a path by sampling and testing motions connecting the samples. Thus a path is comprised of a sequence of nodes connected using local planners. It is appropriate to define soft constraint cost functions that operate on this underlying representation, such that they can be easily integrated within different stages in the sampling-based path planning. At the node generation (sampling) level, the sampled configurations have associated soft constraints. At the edge generation (node connection for roadmap-based planners or tree extension for tree-based planners) level, the local planner associates a soft-constraint with the transition between nodes. During the graph search and post-processing phase soft constraints can also be specified so that among several paths solutions some will be favored among others. Therefore, based on where they are applied in sampling-based planners potential soft constraints are divided into three categories: node-level, edge-level, and global-path-level soft constraints. This structure is used to characterize different soft constraints.

Node-level soft constraint $cost_s^v$. Node-level soft constraints capture the practicality of a configuration of the robot. Such constraints can help in selecting "good" nodes or eliminating "bad" ones in the roadmap or tree. The cost function $cost_s^v$ may take on any value in $[0, 1]$ with 0 corresponding to a state that meets fully the corresponding soft constraint and 1 corresponding to a state that fully violates the corresponding soft constraint.

Edge-level soft constraint. Edge-level soft constraints capture the practicality of a transition from one configuration to another, i.e. an edge connecting local edges between nodes in the roadmap or tree. These soft constraints are used to determine which pairs are preferable when building the roadmap or tree. Sampling-based path planners usually attempt to connect pairs of nodes that are close to each other because the probability that two neighbor configurations can be connected is relatively high. However, short and correct edges may not be practical for the robot to execute. Consider an edge connecting two close feasible configurations (c, c') and an edge-level soft constraint s, the cost of the edge $cost_s^e : \mathcal{C}_{free} \times \mathcal{C}_{free} \rightarrow [0, 1]$, i.e. $cost_s^e(c, c') \in [0, 1]$ needs to be computed for each pair of configurations.

Global-path-level soft constraint. Global-path-level soft constraints are defined over the global path from the initial to the goal configuration rather than just the partial elements (nodes, edges) of the path. These constraints are used to select the practical one among a set of paths, typically in a post-processing phase. Such constraints are typically used to measure consumption of critical resources and other parameters, such as energy consumption and path length. Given a global-path-level soft constraint s, the cost function of the path $cost_s^p : \mathbb{P} \rightarrow [0, 1]$.

Costs related to the transition through a particular node. Although edge related costs are well captured by the edge cost above, such costs do not capture a cost of the transition between two edges that share a node. For instance, it may not be desirable for a robot to make significant changes in trajectory as it passes through a node from one edge to another.

The soft constraint cost of the path is the combination of these and possibly other categories of soft constraints. Various combination mechanisms for the various soft constraints are possible. For example, a max norm would be useful if the soft cost is dependent upon the worst soft cost of the path. A simple sum of the constraints along the path may not be ideal as the stochastic nature of the sampling and edge linking process will produce non-uniform sampling in either configuration or Cartesian space. In the work described

Algorithm 1 Dynamic optimization

```
for t = 1, ..., T do
    for all O_i ∈ 𝒪 do
        U_i ← O_i.PredictUtility
    end for
    O_w ← O_i with highest utility U_i
    O_w.Optimize
    O_w.CorrectUtility
end for
```

here we choose a distance averaged sum of node/edge costs so to not bias the soft constraint cost due to the underlying sampling. Each soft cost has an associated importance weight (see [21]).

Path planning with soft constraints. Given a path planning problem with robot $\mathcal{A}$, workspace $\mathcal{W}$, an initial configuration c_{init}, a goal configuration c_{goal}, a set of soft constraints $\mathcal{SC}$ with corresponding importance weights and a cost function $cost$, generate a feasible path $\mathcal{P}$ such that $cost(\mathcal{P})$ is minimized. Report failure if no feasible path can be found.

Each of the categories of soft constraints have a corresponding optimization within the PRM algorithm. Soft constraints defined at the vertex level can be optimized during the node generation phase. Soft constraints defined at the edge level can be optimized during the edge linking phase, while optimizations defined at the path level can be optimized at the post processing phase. Optimization strategies in each of these phases are well understood (see [27], [22] for examples). Given a set of weighted soft constraint penalty terms and a group of corresponding optimization strategies, they key question becomes which optimizations should be performed and in what order?

IV. COORDINATING MULTIPLE OPTIMIZERS: AN AUCTION-BASED APPROACH

Given a specific set of soft constraints, there typically exists more than one optimization method that can be applied at a given time. Given a constrained optimization budget, how should the various optimization methods be applied to the problem? Among a set of optimizers, only one can be applied to a given solution at a time. Once an optimizer is applied, the solution may change and thus so does the problem that is to be optimized in the next iteration. This optimization process can be viewed as a multi-agent sequential decision-making problem [28], where the global objective is to maximize benefit (soft constraint path cost reduction) accumulated over time while minimizing resource consumption (computational effort). Addressing this resource allocation issue can be decomposed into two main issues: which optimizer should be given the resources to run at a given time; and how to ensure that the system can adapt to deal with uncertainties related to the relative performance of a given optimizer.

A. Multiple-round sequential optimizations

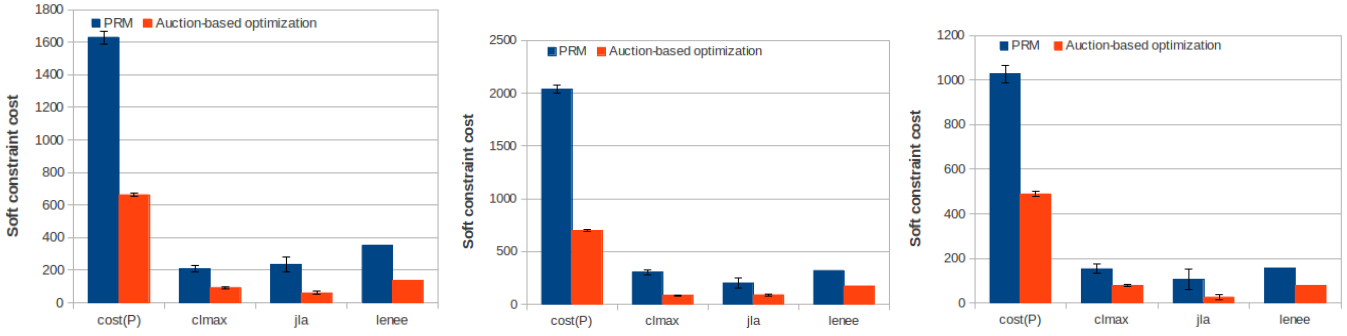
It is clearly intractable to try all possible optimizers in all possible sequences. Nor is it practical to "try" each optimizer at a given state in the optimization sequence and then accept in a greedy manner the locally best optimization strategy. Instead, let us assume that each potential optimizer has an associated utility oracle, that scores the expected enhancement – measured in terms of the soft cost function – of applying this optimization to the current state of the optimization process. Assuming that this oracle is accurate and has negligible computational cost, then a straightforward greedy algorithm can be used to schedule the sequencing of optimizers.

A dynamic optimization algorithm that coordinates multiple optimizers along these lines is outlined in Algorithm 1. Auctions take place among the set of optimizers $\mathcal{O} = \{O_1, O_2, ..., O_n\}$ for T rounds. In each round, optimizers submit their predicted utility for a unit of optimization and the optimizer with the highest expected utility is declared the winner. After the winner O_w performs its unit-cost optimization on the current solution, the optimizers correct their expected utility U_i based on the new information acquired. The key problem of course is estimating the expected utility of each optimizer, given the current system state, and properly recalibrating these utility estimates throughout the path finding process. The modularity of this approach allows for flexibility in both the number and types of the optimizers available, including the pre-processing, post-processing and customized-learning techniques.

B. Auction mechanism

Because the auction acts to minimize the practical path cost within budget, the utility function must closely correspond to actual progress toward this goal. There are many ways to define the utility function. One way to measure the utility of an optimizer is to measure the path soft cost improvement the optimizer achieves. It is desirable that this measure be a selective measure as we can expect substantial changes in scale of the absolute cost and improvement during the optimization process. In this work, a utility function that is equal to the percentage of the path cost reduction from the old path to the new path obtained by the optimizer, i.e. $U = (cost(P_{old}) - cost(P_{new}))/cost(P_{old})$ is used. The oracle does not have access to this utility function, of course, but rather produces an inexpensive "best guess" as to how the investment of a unit of optimization effort is expected to improve the solution.

The process of optimizing and updating the utility functions must be efficient. The n optimizers that participate in the auction at time t places a bid. Let $\hat{U}(t) = (\hat{U}_1(t), \hat{U}_2(t), ..., \hat{U}_n(t))'$ be the $n \times 1$ vector containing the estimated utility of the true utility $U(t)$ of the optimizers. Suppose an optimizer O_w wins the auction and is chosen to execute, its utility distribution $U_w(t)$ is partly revealed via the observed cost of the new path, whereas the true utility of all other optimizers remain hidden because they did not execute at time t. We now have $U_w(t)$ for the winning optimizer O_w and $\hat{U}_i$ for all of the optimizers. How should



(a) $c_1 \rightarrow c_2$ (b) $c_2 \rightarrow c_3$ (c) $c_3 \rightarrow c_4$

Fig. 2: Soft constraint cost comparison between basic PRM path planning and auction-based path optimization with soft constraints for the poses shown in Fig 1. The plot shows the average solution soft constraints costs with standard deviations. All values are taken over 20 independent runs.

we estimate $\hat{U}_i$ at $t+1$ efficiently? We seek a straightforward way of estimating $\hat{U}_i(t+1)$ given the sequence of measured utility values $U_{w_t}(t)$ the winning optimizers. Let us make the strong assumption that the error between $\hat{U}_{w_i}(t)$ and $U_{w_i}(t)$ is $N(0, \sigma_i)$ and that we can treat the optimization of $\hat{U}(t)$ as a recursive least squares estimation process. Following the notation of a Kalman filters [29] we define the state vector by U which evolves following a linear plant model.

$$\hat{U}(t) = F(t)\hat{U}(t-1) + B(t)d(t-1) + w(t) \qquad (1)$$

where $F(t)$ is the transition model (here $F(t) = I$), $B(t)$ is the control input model which is applied to the control vector $d(t-1)$. In our case we assume that $B(t) = I$ and that $d(t)$ is a term to capture the expected reduction in utility as the optimization process proceeds. Specifically $d(t)$ represents the drift rate that is expected to be negative, which can be constant or time-varying. $w(t)$ is the process noise which is assumed to be drawn from a zero mean multivariate normal distribution with covariance $Q(t)$.

At time t an observation is made of the true utility of the winning optimizer according to

$$z(t) = H(t)x(t) + v(t) \qquad (2)$$

Here $H(t)$ is a matrix with one non-zero diagonal element corresponding to the chosen optimizer. $v(t)$ is the observation noise. $v(t)$ is assumed to be zero mean Gaussian with covariance $R(t)$. Theoretically $R(t)$ is zero, but here is implemented as a small non-zero value to model numerical errors. Under this assumption we can follow standard recursive least-squares mechanisms (e.g., a Kalman [29] or Particle [30] filter) to implement the optimization process. Here we choose to utilize a Kalman updating process.

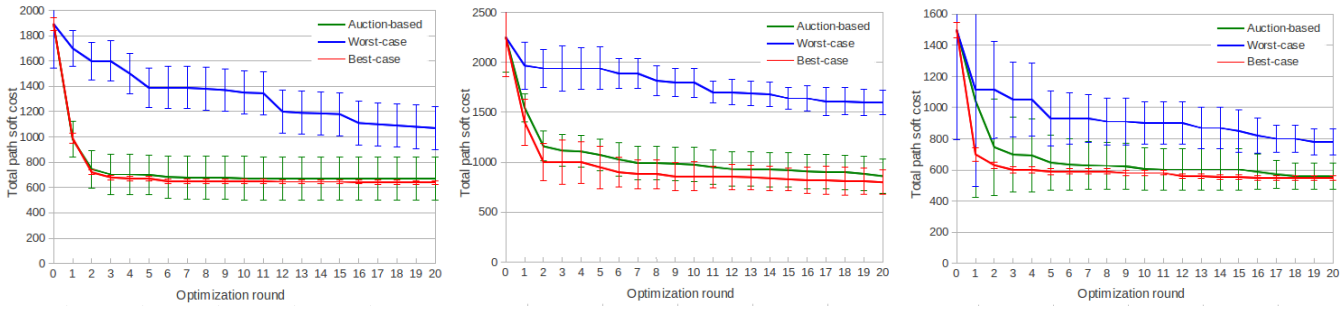
In practice it is likely that $F(t) = I$, which provides considerable computationally efficiencies in terms of the computation of the state covariance matrix – it may prove to be diagonal under reasonable assumptions – allowing the Kalman filter to be separated for each optimizer.

V. EXPERIMENTAL VALIDATION

This section applies the theoretical framework for planning practical paths within an optimization budget on a tentacle robots designed and built by OC Robotics, shown in Fig. 1. The robot has a mobile base that translates in one dimension and seven joints, each consisting of two DOFs - pitch and yaw. The robot has 15 DOFs in total. The environment is based on a hot cell mock-up constructed by OC Robotics, representing a dry processing cell containing pipework and vessels [31]. It is a representative of confined-space challenges found in contaminated environments. The algorithms were implemented within LaValle's Motion Strategy Library [32], and were run on a Mac running OS X with 3.06 GHz Intel Core 2 Duo processor and 2 GB memory.

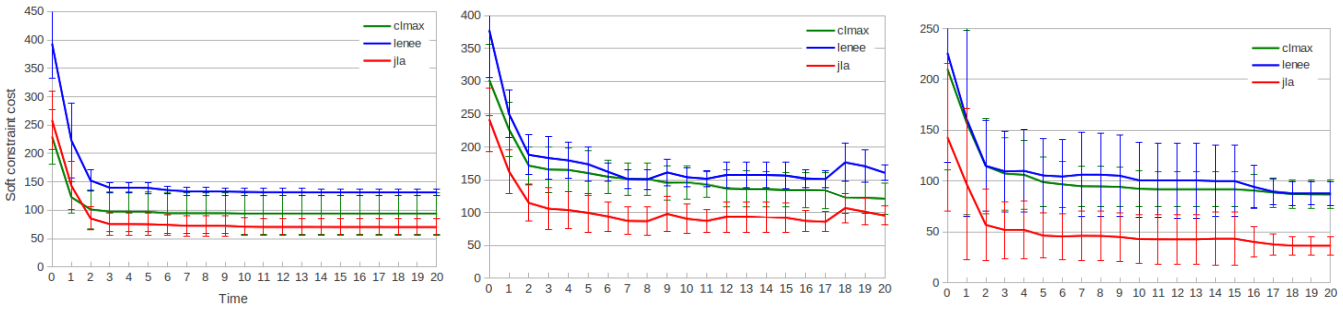
There are a number of different properties the tentacle robot should exhibit for this hot cell task, but safety is perhaps the most important issue. First, we choose two node-level soft constraints that are commonly associated with tentacle robots, including clearance maximization $cost^v_{clmax}$ and joint limit avoidance $cost^v_{jla}$. The robot is typically required to carry a tool by its end-effector, so it is desired that the robot's end-effector moves along a short path. We define a global-path-level soft constraint cost for shortening the distance travelled by the end-effector $cost^p_{lenee}$. In this example, as safety is the most important concern for nuclear tasks we assign the weights as $w_{clmax} = 5, w_{jla} = 1, w_{lenee} = 1$. Both node-level and global-path-level soft constraints are involved in this experiments, so optimization strategies designed for these two types of soft constraints are suitable here. We optimize $cost^v_{clmax}$ and $cost^v_{jla}$ in the sampling phase of PRM*, and $cost^p_{lenee}$ in the post-processing phase. These two optimizers are combined in the auction framework.

Figure 2 shows the soft constraint cost of the paths computed by the basic PRM algorithm and the auction-based optimization. The optimization strategy reduced the cost of the total soft constraint cost by at least 50%. Furthermore, the optimization strategy reduced the cost of each individual soft constraint by at least 50%, although the improvement for each soft constraint varies in the different scenarios.



(a) $c_1 \rightarrow c_2$ (b) $c_2 \rightarrow c_3$ (c) $c_3 \rightarrow c_4$

Fig. 3: Performance of the auction-based path optimization. Upper and lower bounds are shown. The plot shows the average solution soft constraints costs with standard deviation.



(a) $c_1 \rightarrow c_2$ (b) $c_2 \rightarrow c_3$ (c) $c_3 \rightarrow c_4$

Fig. 4: Effects of the auction-based path optimization with respect to time budget. The plot shows the individual soft constraints costs averages with standard deviation. All values are taken over 20 independent runs.

For example, to move the robot from c_2 to c_3 (shown in Figure 2(b)) the optimization is able to discover a path with relatively larger clearance, resulting $cost^v_{clmax}$ to decrease by about 70%. To move the robot from c_1 to c_2 and from c_3 to c_4 (shown in Figure 2(a) and (c)), the optimization is able to reduce $cost^v_{jla}$ by about 75%.

Figure 3 compares the performance of the auction-based approach with the best case and the worst case in each round of the optimization. The total combined soft constraint costs of the paths are compared. The best/worst case in each round is computed by trying all the participating optimizers before choosing the one that improve the path cost the most/least. This increased the computation time to $2T$, but provides an idea of how well our auction-based approach performs. These charts plot the average path cost with standard deviations in each round of the optimization.

Although the auction-based optimization aims at reducing the combined soft costs of the solution, it is interesting to investigate its effects on each individual soft constraint. Figure 4 plots the solution cost of each soft constraint achieved by the auction-based optimization given a specific planing budget. Note that it is not the case that each soft constraint cost is monotonically decreasing. This is due to the conflict between different soft constraints. For example, in Figure 4(b) at some time during the optimization $cost_{jla}$ and $cost_{lenee}$ increased while $cost_{clmax}$ decreased. This is a consequence of the safe clearance soft constraint having a higher importance weight than joint limit avoidance and length of the end-effector, i.e. $w_{clmax} > w_{jla}$ and $w_{clmax} > w_{lenee}$.

VI. SUMMARY

Path planning is an important but difficult problem in robot planning with high numbers of DOFs. Sampling-based path planning algorithms are successful in solving high-dimensional problems. However, their ability to find paths that meet certain soft constraints is still limited. This paper presents a framework within which plans can be developed for high DOF robotic systems within a time budget that meet the hard constraints of path feasibility and at the same time seek to reduce the cost of a collection of domain-specific soft constraints. This framework is intended to be robot independent, but in this paper the approach is grounded in the capabilities and tasks associated with tentacle robots.

ACKNOWLEDGEMENT

This work was supported by the Natural Sciences and Engineering Research Council (NSERC) through the NSERC Canadian Field Robotics Network (NCFRN).

REFERENCES

[1] J. F. Canny, *The Complexity of Robot Motion Planning*. Cambridge, MA, USA: MIT Press, 1988.

[2] J. Bruce and M. Veloso, "Real-Time Multi-Robot Motion Planning with Safe Dynamics," in *Multi-Robot Systems: From Swarms to Intelligent Automata*, vol. 3, 2005.

[3] M. Kobilarov and G. S. Sukhatme, "Near time-optimal constrained trajectory planning on outdoor terrain," in *Proceedings IEEE International Conference on Robotics & Automation (ICRA)*, April 2005, pp. 1833–1840.

[4] L. E. Kavraki, J.-C. Latombe, R. Motwani, and P. Raghavan, "Randomized query processing in robot path planning," *Journal of Computer and System Sciences*, vol. 57, no. 1, pp. 50–60, 1998.

[5] S. M. LaValle and J. J. Kuffner, "Rapidly-exploring random trees: Progress and prospects," in *Proceedings International Workshop on Algorithmic Foundations of Robotics (WAFR)*, 2000.

[6] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *International Journal of Robotics Research*, vol. 30, no. 7, pp. 846–894, 2011.

[7] D. Nieuwenhuisen and M. H. Overmars, "Useful cycles in probabilistic roadmap graphs," in *Proceedings IEEE International Conference on Robotics & Automation (ICRA)*, vol. 1, April-May 2004, pp. 446–452.

[8] R. Geraerts, "Sampling-based motion planning: Analysis and path quality," Ph.D. dissertation, Utrecht University, 2006.

[9] D. Hsu, "Randomized single-query motion planning in expansive spaces," Ph.D. dissertation, Stanford University, May 2000.

[10] R. Geraerts and M. Overmars, "On improving the clearance for robots in high-dimensional configuration spaces," in *Proceedings IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, August 2005, pp. 679–684.

[11] B. Raveh, A. Enosh, and D. Halperin, "A little more, a lot better: Improving path quality by a path merging algorithm," *IEEE Transactions on Robotics*, vol. 27, no. 2, pp. 365–371, 2011. [Online]. Available: http://arxiv.org/abs/1001.2391

[12] J. Kim, R. A. Pearce, and N. M. Amato, "Extracting optimal paths from roadmaps for motion planning," in *Proceedings IEEE International Conference on Robotics & Automation (ICRA)*, vol. 2, September 2003, pp. 2424–2429.

[13] C. Nielsen and L. E. Kavraki, "A two-level fuzzy PRM for manipulation planning," in *Proceedings IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, vol. 3. IEEE Press, November 2000, inproceedings, pp. 1716–1722.

[14] R. Bohlin and L. E. Kavraki, "Path planning using Lazy PRM," in *Proceedings IEEE International Conference on Robotics & Automation (ICRA)*, vol. 1, IEEE Press. San Fransisco, CA, USA: IEEE Press, April 2000, inproceedings, pp. 521–528.

[15] O. B. Bayazit, "Solving motion planning problems by iterative relaxation of constraints," Ph.D. dissertation, Texas A&M University, 2003.

[16] G. Song, S. Miller, and N. M. Amato, "Customizing PRM roadmaps at query time," in *Proceedings IEEE International Conference on Robotics & Automation (ICRA)*, 2001, pp. 1500–1505.

[17] R. V. Dubey, S. McGhee, and T. F. Chan, "Probability-based weighting of performance criteria for redundant manipulators," *Journal of Intelligent and Robotic Systems*, vol. 19, pp. 89–103, 1997.

[18] S. Bistarelli, U. Montanari, and F. Rossi, "Semiring-based constraint satisfaction and optimization," *Journal of the ACM*, vol. 44, no. 2, pp. 201–236, 1997.

[19] E. C. Kerrigan and J. M. Maciejowski, "Soft constraints and exact penalty functions in model predictive control," in *Proceedings UKACC International Conference (Control 2000)*, 2000.

[20] M. Garber and M. C. Lin, "Constraint-based motion planning using Voronoi diagrams," in *Proceedings International Workshop on Algorithmic Foundations of Robotics (WAFR)*, 2002.

[21] A. Gerevini and D. Long, "Plan constraints and preferences in PDDL3 - the language of the fifth international planning competition," University of Brescia, Tech. Rep., 2005.

[22] J. Yang, R. Codd-Downey, P. Dymond, J. Xu, and M. Jenkin, "Planning practical paths for tentacle robots," in *Proceedings 6th International Conference on Agents and Artificial Intelligence (ICAART)*, 2013.

[23] L. Hurwicz, "The design of mechanisms for resource allocation," *The American Economic Review*, vol. 63, no. 2, pp. 1–30, 1973.

[24] R. Cassady, *Auctions and Auctioneering*. California University Press, Berkeley, 1967.

[25] R. McAfee and J. McMillan, "Auctions and bidding," *Journal of Economic Literature*, no. 25, pp. 699–738, 1987.

[26] M. B. Dias, R. Zlot, N. Kalra, and A. Stentz, "Market-based multirobot coordination: A survey and analysis," in *Proceedings of the IEEE*, vol. 94, 2006, pp. 1257–1270.

[27] J. Yang, P. Dymond, and M. Jenkin, "Practicality-based probabilistic roadmaps method," in *Proceedings Canadian Conference on Computer and Robot Vision (CRV)*, 2011, pp. 102–108.

[28] R. Cavallo, "Social welfare maximization in dynamic strategic decision problems," Ph.D. dissertation, Harvard University, 2008.

[29] R. E. Kalman, "A new approach to linear filtering and prediction problems," *Transactions of the ASME–Journal of Basic Engineering*, pp. 35–45, 1960.

[30] A. Doucet, N. Freitas, and N. Gordon, *Sequential Monte Carlo Methods in Practice*. Springer-Verlag, 2001.

[31] "Nuclear decommissioning case study: Sellafield," 2014. [Online]. Available: http://www.ocrobotics.com/applications–solutions/nuclear/nuclear-case-study–sellafield/

[32] S. M. LaValle, *Planning Algorithms*. Cambridge, U.K.: Cambridge University Press, 2006, available at http://planning.cs.uiuc.edu.