Path Planning for a Tethered Robot using Multi-Heuristic A* with Topology-based Heuristics

Soonkyum Kim[1] and Maxim Likhachev[2]

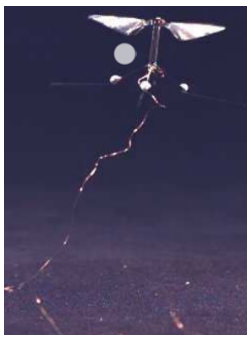
***Abstract*— In this paper, we solve the path planning problem for a tethered mobile robot, which is connected to a fixed base by a cable of length L. The reachable space of the robot is restricted by the length of the cable and obstacles. The reachable space of the tethered robot can be computed by considering the topology class of the cable. However, it is computationally too expensive to compute this space a-priori. Instead, in this paper, we show how we can plan using a recently-developed variant of A* search, called Multi-Heuristic A*. Normally, the Multi-Heuristic A* algorithm takes in a fixed set of heuristic functions. In our problem, however, the heuristics represent length of paths to the goal along different topology classes, and there can be too many of them and not all the topology classes are useful. To deal with this, we adapt Multi-Heuristic A* to work with a dynamically generated set of heuristic functions. It starts out as a normal weighted A*. Whenever the search gets trapped in a local minimum, we find the proper topology class of the path to escape from it and add the corresponding new heuristic function into the set of heuristic functions considered by the search. We present experimental analysis comparing our approach with weighted A* on planning for a tethered robot in simulation.**

I. INTRODUCTION

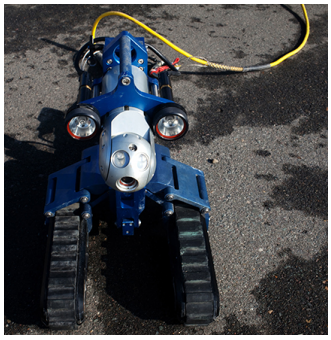
In extreme environments such as disaster areas, wireless signal may not be strong enough for an operator to communicate to a robot [16], [19]. For example, this was the case when robots were deployed in the reactor building after the disaster of Fukushima due to the radioactive environment [8]. In such cases, using power and communication cables to tether the robot can effectively solve such problems [5]. Tethering also helps in deploying robot in environments with limited accessibility. For example, Nassiraei et al. designed a tethered sewer pipe inspection robot to work instead of a human operator to decrease the cost and to speed-up the inspection [17]. Also, tethered mobile robot can be used to maintain and construct highways [12].

Tethering also used whenever size limits or actuation power make it impossible for a robot to carry its own power supply. For example, a number of prototypes of micro robots use external power source to actuate the motors or sensors as shown in Figure 1(a) [6] and Figure 1(b). These robots are tethered.

While tethering solves communication and power problem for mobile robots, it also causes challenges in control and planning. In general, the cable is stiff and has finite length, which restricts the workspace of the mobile robot around the fixed base point. Also, due to obstacles, certain robot poses become reachable only under specific cable configurations (the topology class of the cable).



(a) A picture of a flapping-wing microrobot prototype taken from (Chirarattananon, Ma, and Wood 2013).



(b) A tethered tunnel inspecting robot taken from http://www.wired.com/2011/12/robot-tunnels/all/

Fig. 1. Examples of tethered robots

In this work, we consider the path planning problem for a tethered mobile robot. The work space of the tethered robot is restricted by the cable of maximum length L which connects the robot to a fixed base. This workspace can be calculated by considering the homotopy class of the cable [14], which is computationally too expensive for online planning. Instead of relying on this preprocessing, we consider the topology class of the path and cable to guide a heuristic search in the form of heuristics. Recently-developed Multi-Heuristic A* allows to deal with numerous inadmissible heuristic functions while guaranteeing the suboptimality bound [2]. As there could be too many possible topology classes and corresponding heuristic functions and not all of them are useful during the search, we propose a Topology-based Multi-Heuristic A*, which starts as a normal weighted A* [18] but shifts to Multi-Heuristic A* by adding new heuristic functions to escape from local minima.

II. RELATED WORK

There has been active research on path planning for tethered mobile robots. Some early work has considered a tangle-free path planning of a group of tethered mobiles robots in obstacle-free environments [11]. In contrast, our approach can handle arbitrary obstacles.

Finding the shortest path for a tethered mobile robot was studied in [23], [24]. The authors triangulated the environment by using all the edges of polygonal obstacles as

[1]Soonkyum Kim is a Postdoctorial Fellow of Robotics Institute, Carnegie Mellon University, 5000 Forbes Avenue, Pittsburgh, Pennsylvania, 15213, USA kim.soonkyum@gmail.com

[2]Maxim Likhachev is a Research Assistant Professor of Robotics Institute, Carnegie Mellon University, 5000 Forbes Avenue, Pittsburgh, Pennsylvania, 15213, USA maxim@cs.cmu.edu

edges of triangles. Then the authors built the visibility graph to find the shortest path. However, the suggested algorithm suffers from the computational complexity associated with the number of vertex of obstacles. In addition, our approach is not constrained to polygonal obstacles.

Homotopy class of the cable was considered in [13] to solve a similar problem. However, the authors did not use the *homotopy invariant* but considered the path length to the node to distinguish paths in different homotopy classes. This representation of homotopy class is based on the metric information of the path and can lose some important information about the paths. This loss of information can result in failure to distinguish paths in different homotopy classes and may lead to finding solutions that are infeasible. For example, they will not be able to differentiate between two paths of the same length even if one passes an obstacle on the right and the other one on the left. In our work, we do consider the homotopy invariant and build an augmented graph to handle the topology information of path and cable. As a result, the solutions found by our approach are guaranteed to be feasible.

Motion planning for a tethered axel rover on steep terrain was studied in [1]. They also consider the homotopy class of the path of round trip path to the goal to minimize the possibility of entanglement of the cable. This work was later extended to an online motion planning algorithm to deal with partially known environment [21]. Also, finding the coverage area is another form of planning problem. In [20], the authors solved the planning problem for a tethered mobile robot to explore and cover the partially known environment and return to the base point, which also works as the starting point. In our work, we consider arbitrary starting position and cable configurations.

The shortest path of a tethered mobile robot can be generated by relying on pre-computation of the reachable space, which is computationally expensive [14]. We avoid this pre-computation and propose a novel algorithm to consider the topology class of the cable and robot path to focus the search. This method results in fast computation of bounded suboptimal paths.

III. The Preliminaries

In this section, we discuss the topology of curves, which could be paths or cables. Since the notion of topology was already introduced in various literature [7], [9], [10], [22], [3], we will briefly discuss the topology required in our paper. Throughout this paper, we follow the same notations as [14].

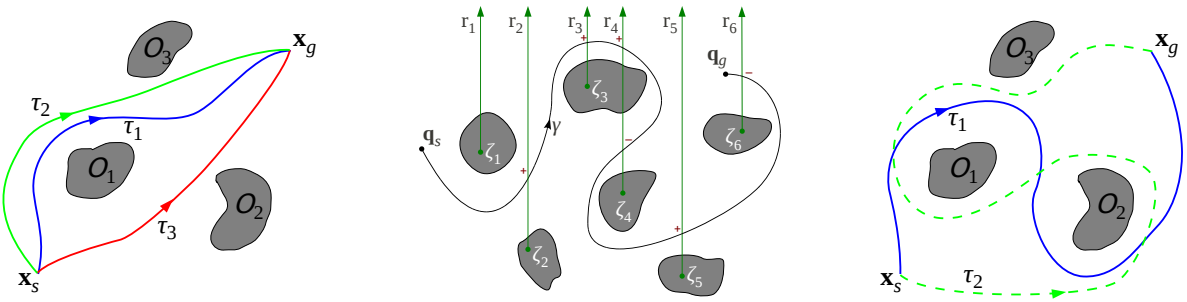
We consider the workspace or search space, $W \subset \Re^2$ as a simply connected and bounded region on plane. We have m arbitrary-shaped obstacles of $\mathcal{O} = \{O_1, O_2, \ldots, O_m\}$. We consider a path or cable in the workspace as a curve on the plane. Two curves, connecting the same start and end points, are *homotopic* if and only if one can continuously deform to the other without intersecting any obstacles. We call the set of curves that are homotopic *homotopy class*. For example, the two curves, τ_1 and τ_2 in Figure 2(a) are homotopic and in the same homotopy class. However, τ_3 is not homotopic to these two curves as it cannot deform to τ_1 or τ_2 because of the obstacle, O_1.

Homotopy invariant is a function of curves that will return the same value for all the curves in the same homotopy class and the different values for pairs of curves that lie in different homotopy classes. In general, it is not easy to find or design such a function. In a 2-dimensional plane, however, one can find a relatively simple and easy homotopy invariant. To do this, we choose reference points inside of each obstacle, $\zeta_i = (x_{r,i}, y_{r,i}) \in O_i$. Then we set the reference ray, r_i, for each obstacle to start from the corresponding reference point in the direction of $[0, 1]^T$. These reference rays serve to compute the homotopy invariant, *word*, of the given curve. We can then represent the homotopy class of the given curve, $h(\cdot)$, by building a *word* to trace the curve while consecutively appending a *letter* corresponding to the reference ray that the curve crosses. For example, if the curve crosses the reference ray r_i from left to right, we add "r_i." When the curve crosses the reference ray from right to left, we add "r_i^{-1}." Figure 2(b) shows an example (borrowed from [14]), the homotopy invariant of the curve γ is $h(\gamma) =$ "$r_2 r_3 r_5 r_6^{-1}$." Note that "$r_4 r_4^{-1}$" gets evaluated to an empty word, "".

Homology is a similar concept to homotopy but slightly different. Two curves, connecting the same start and end points, are *homologous* if and only if they form a boundary of a 2-dimensional region that does not contain or intersect any obstacles. If two curves are homotopic, they are homologous. However, the inverse is not necessarily true. Figure 2(c) shows an example that the two curves, τ_1 and τ_2, are homologous but not homotopic. The homology invariant of the curves was defined as H-signature of the curve [3]. The H-signature is not unique and can be designed in several ways including Cauchy-Integration [3], [4]. In this work, we use the H-signature introduced in [15] because this H-signature is designed to count the number of crossing with reference rays, $+1$ for left-to-right and -1 for right-to-left. This function can be calculated from the homotopy invariant, *word*, defined above by counting the number of the corresponding letters appearing in the word. For example, the H-signature of the curve γ in Figure 2(b) is $H(\gamma) = [0, 1, 1, 0, 1, -1]^T$, where i^{th} components correspond to obstacle O_i.

IV. Problem Formulation

In this paper, we solve the problem of planning a path for a tethered mobile robot from the given initial position and cable configuration to the goal while satisfying the cable length constraint. We set the maximum cable length as L. We represent the problem as searching for a path in a homotopy invariant augmented graph. The vertex of the augmented graph, $(\mathrm{q}, \mathfrak{w})$, includes the position of the robot and homotopy class of the cable, respectively. Also, to decrease the computation load, we add a variable, called cable length l, which is initialized as the length of initial cable configuration at the start vertex and is updated at every generated state based on its predecessor's l and the transition. Each vertex is feasible if the position of the robot



(a) Example where τ_1 and τ_2 are homotopic. τ_1 and τ_3 are not homotopic because of obstacle O_1.

(b) ζ_i are representative points inside the obstacles, $O_1, O_2, \cdots, O_m$ (in that order), and r_i are rays emanating from the respective points. The homotopy invariant of this curve γ is $h(\gamma) =$ "$r_2 r_3 r_5 r_6^{-1}$".

(c) Example where two curves (τ_1 and τ_2) are homologous but not homotopic.

Fig. 2. We consider two topology classes of curves, homotopy class and homology class.

is obstacle free and if this configuration is *reachable* [14]. In other words, the given position is reachable if there is at least one feasible cable configuration, satisfying cable length constraint and is obstacle free, under the given homotopy class, $\mathfrak{w}$. It is a sufficient condition of the reachability that the dependent variable l is less than the maximum cable length, $l \leq L$. So, whenever l is longer than the cable length, we run the $l = curveShorten(s, \mathcal{O})$, described below, which returns the length of the shorten path connecting the given state s and the fixed base from the given state s and a set of obstacles $\mathcal{O}$. Then we update the value of l. If the updated value of l is smaller than the maximum cable length L, this vertex is feasible. Even when the updated value of l is greater than L, there can be a curve that is homotopic to the extracted path and its length is shorter than L. So this method is a conservative way to find or build a feasible and reachable space of the tethered mobile robot. The following is the brief algorithm for the $CurveShorten$ function that was also used in [14]. (Here, we keep track of which obstacles, $\mathcal{O}_c$, are blocking the visibility to remove unnecessary computation in the later part of the planner.)

Algorithm *CurveShorten*:

1) **curveShorten**$(s, \mathcal{O} = \{O_1, O_2, \ldots, O_m\})$
2) $P = [v_0 = q_b, v_1, \ldots, v_n = s] = extractPathTo(s)$;
3) $l = 0$, $\mathcal{O}_c = \emptyset$;
4) $i = 0, \quad j = 0$;
5) **while** $j \leq n$
6) **if** $j < n$ **AND** $\overline{\mathrm{q}_i\mathrm{q}_{j+1}} \cap \mathcal{O} = \emptyset$
7) $j = j + 1$;
8) **else**
9) $l = l + \|\overline{\mathrm{q}_i\mathrm{q}_j}\|$;
10) $i = j$;
11) add $\forall k$ into $\mathcal{O}_c$ s.t. $\overline{\mathrm{q}_i\mathrm{q}_{j+1}} \cap O_k \neq \emptyset$
12) $l = l + \|\overline{\mathrm{q}_i\mathrm{q}_j}\|$;
13) **return** $(l, \mathcal{O}_c)$

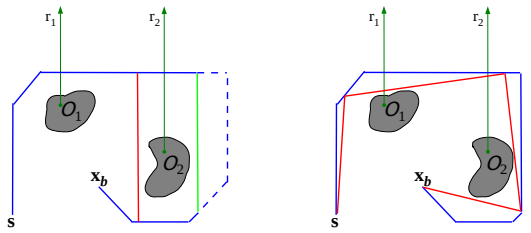
In the above algorithm $extractPathTo(s)$ generates the sequence of vertices from the state to the initial configuration by backtracking the path from the given vertex, s, to the initial configuration of the robot and appending the initial cable configuration as a sequence of connected states. Then this sequence lies in the same homotopy class with the cable configuration of the given vertex s. To speed up and improve the performance of the $curveShorten()$, we trace all the vertices on this sequence to find a pair of vertices that have the same x value and the same homotopy invariants. If the path segment connecting these vertices is not a straight vertical line and if such vertical straight line segment is feasible, we can replace the path segment with this vertical straight line because the process does not change the homotopy class of the path and the replaced segment is definitely shorter than the original segment. The green vertical line of the Figure 3(a) shows an example of such replacement. However, the red vertical line is obstacle free but cannot replace the original path as it will change the topology class of the path. By doing so, we run the $curveShorten()$ with slightly different and shorter initial path than the one in [14]. This additional procedure helps to avoid finding locally optimal solutions. The red curve in Figure 3(b) shows an example of the shortened path achieved by the above $curveShorten()$ from the given initial path of the blue curve.

V. Topology-based Multi-Heuristic A*

A. Heuristic functions considering topology

To search the graph we use a heuristic search. As such, it relies heavily on having a good heuristic function that for any state s estimates the cost of reaching a goal state from s.

In this work, we minimize the travelling distance of the robot on a plane. The most common admissible heuristic in 2D search is the Euclidean distance to the goal. In this work, we choose the Euclidean distance as the admissible heuristic of the search. However, the search could be trapped in local minima, which are usually caused by obstacles that block the movement of the cable while exploring the search



(a) Example of $extractPathTo(s)$. The blue curve is the path extracted by backtracking form s to the base. The vertical green curve can replace the dashed part of the blue curve because this replacement does not change the topology class of the path. However, the vertical red curve is not acceptable as it will change the topology class of the path.

(b) Example of $curveShorten$. The blue curve is the initial sequence of states achieved from $extractPathTo(s)$. The red curve is the line segments, $\overline{q_i q_j}$, in line 9 and 12 of the Algorithm $CurveShorten$. So the length of this red curve is the value of $curveShorten()$, which is shorter than the length of the blue curve.

Fig. 3. Example of Algorithm $curveShorten$.

space guided by a heuristic function which does not consider the cable length constraint and obstacles. If there are some additional heuristic functions to help escaping from these local minima, they can decrease the computation. To this end, we consider the topology-based additional heuristic functions to calculate the length of the shortest path under specific *homology classes*. In the following section, we explain how these heuristic functions are being used.

The heuristic function we design calculates the length of the shortest path from the given position, q_c, to the goal position, q_g, while considering the change of the H-signature value corresponding to the given obstacle, O_i. There are three different types of heuristic functions: not changing the value of H-signature, increasing the value of H-signature and decreasing the value of H-signature. However, it is not trivial to compute such functions as they also depend on the size and shape of the obstacles. We therefore designed a heuristic function which finds the under-estimated length of the path by considering the corresponding obstacle as a vertical line segment and ignoring other obstacles. For each obstacle, we define $y_{min,i} = min_{(x,y)\in O_i}\ y$ s.t. $x = x_{r,i}$ and $y_{max,i} = max_{(x,y)\in O_i}\ y$ s.t. $x = x_{r,i}$. We then consider O_i and a vertical line segment connecting $(x_{r,i}, y_{min,i})$ and $(x_{r,i}, y_{max,i})$, which is the longest vertical line inside of O_i containing ζ_i. Then we can calculate the length of the path from given position to the goal to change or maintain the homology class corresponding to obstacle. The following algorithm shows how to calculate such distance by using symmetry of the problem.

Algorithm ***heuristic function considering homology class*:**

1) **procedure distNotToChange**($q_c = (x_c, y_c)$, $q_g = (x_g, y_g), O_i$)
2) **if** $x_c > x_g$
3) return distNotToChange(q_g, q_c, O_i)
4) **if** $x_c < x_i$ and $x_i < x_g$
5) **if** $(y_g - y_c)(x_i - x_c) > (y_{min,i} - y_c)(x_g - x_c)$
6) return dist$(x_c, y_c, x_i, y_{min,i})$ +dist$(x_i, y_{min,i}, x_g, y_g)$
7) return dist(x_c, y_c, x_g, y_g)
8) **procedure distIncrease**($q_c = (x_c, y_c)$, $q_g = (x_g, y_g), O_i$)
9) **if** $x_c > x_g$
10) return distDecrease(q_g, q_c, O_i)
11) **if** $x_i < x_c$
12) return dist$(x_c, y_c, x_i, y_{min,i})$+$(y_{max,i} - y_{min,i})$ +dist$(x_i, y_{max,i}, x_g, y_g)$
13) **if** $x_i < x_g$
14) **if** $(y_g - y_c)(x_i - x_c) > (y_{max,i} - y_c)(x_g - x_c)$
15) return dist(x_c, y_c, x_g, y_g)
16) **else**
17) return dist$(x_c, y_c, x_i, y_{max,i})$ +dist$(x_i, y_{max,i}, x_g, y_g)$
18) return dist$(x_c, y_c, x_i, y_{max,i})$+$(y_{max,i} - y_{min,i})$ +dist$(x_i, y_{min,i}, x_g, y_g)$
19) **procedure distDecrease**($q_c = (x_c, y_c)$, $q_g = (x_g, y_g), O_i$)
20) **if** $x_c > x_g$
21) return distIncrease(q_g, q_c, O_i)
22) **if** $x_i < x_c$
23) return dist$(x_c, y_c, x_i, y_{max,i})$+$(y_{max,i} - y_{min,i})$ +dist$(x_i, y_{min,i}, x_g, y_g)$
24) **if** $x_i < x_g$
25) return dist$(x_c, y_c, x_i, y_{min,i})$+$2(y_{max,i} - y_{min,i})$ +dist$(x_i, y_{max,i}, x_g, y_g)$
26) return dist$(x_c, y_c, x_i, y_{min,i})$+$(y_{max,i} - y_{min,i})$ +dist$(x_i, y_{max,i}, x_g, y_g)$
27) **procedure dist**(x_1, y_1, x_2, y_2)
28) return $\sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$

In our algorithm, we will be adding new heuristic functions whenever the search is trapped in the local minimum. As part of this process, we will be figuring out which heuristic function helps to escape from this local minimum. We start with the empty set of additional heuristic functions, $H_{add} = \emptyset$. Whenever we expand vertices in a local minimum, we find a critical obstacle by assuming that this local minimum is caused by the cable length constraint. The following description of the algorithm shown in *incHeuristics(s)* shows how it works for a vertex s. First, we find the obstacles that block the visibility, which causes the cable to detour. Then, we find the most dominant obstacle by running and comparing the result of $curveShorten()$ with different sets of obstacles. Then we add the desired homology class of the path to pass the chosen obstacle in opposite direction.

Algorithm *add new heuristic function*:

1) **procedure incHeuristic**(s)
2) $(l_c, \mathcal{O}_c) = curveShorten(s, \overline{\mathcal{O}})$;
3) find O_k minimizes $l_k = curveShorten(s, \overline{\mathcal{O}} \setminus \{O_k\})$
4) **if** $l_k < l_c$
5) $w = H_k(\mathfrak{w}, k)$;
6) **Case based on** w
7) **case** $w > 0$
8) $w_d = w - 1$;
9) **case** $w < 0$
10) $w_d = w + 1$;
11) **case** $w = 0$
12) **if** s is on left side of goal
13) $w_d = 1$;
14) **else**
15) $w_d = -1$;
16) add (O_k, w_d) into H_{add}
17) **procedure** $\mathbf{H}_k$**(**$\mathfrak{w}$**,** k**)**
18) return (# of r_k in $\mathfrak{w}$ - # of r_k^{-1} in $\mathfrak{w}$)

Here $H_k()$ calculates the H-signature of the given vertex corresponding to the given obstacle and w_d is the desired H-signature to avoid the local minimum due to the corresponding obstacle. By doing so, the corresponding heuristic function for each desired homology class is calculated for the given vertex. If there is a heuristic function corresponding to desired homology class of (O_k, w_d) and a given vertex $s = (\mathrm{q}, \mathfrak{w})$, the heuristic function to the goal, q_g is

$$h(s) = \begin{cases} distNotToChange(\mathrm{q}, \mathrm{q}_g, O_k) & \text{if } w_s = w_d \\ distIncrease(\mathrm{q}, \mathrm{q}_g, O_k) & \text{if } w_s < w_d \\ distDecrease(\mathrm{q}, \mathrm{q}_g, O_k) & \text{if } w_s > w_d \end{cases}$$

where $w_s = H_k(\mathfrak{w}, k)$ is the value of H-signature of the given vertex, q, corresponding to the obstacle O_k.

B. Algorithm

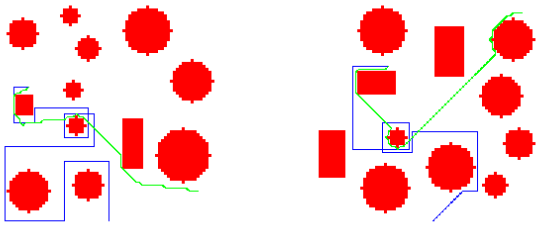
The Topology-based Multi-Heuristic A* is a variation of Shared Multi-Heuristic A* in [2], which explores the search space guided by fixed number of heuristic functions. The difference between the proposed Topology-based Multi-Heuristic A* and Shared Multi-Heuristic A* is that the former starts out by following the normal weighted A* search with a single admissible heuristic function, which is the Euclidean distance to the goal in this work, and whenever the search is trapped in a local minimum, it adds additional heuristic functions, which are not required to be admissible, to escape from this local minimum. These additional heuristic functions are achieved by considering the topology class of the path to the goal and current cable configuration as described in the previous section. If we have more than one heuristic function, the search switches to Shared Multi-Heuristic A* [2]. Given m obstacles, there are $3m+1$ candidate heuristic functions, one for the admissible heuristic function and three heuristic functions per obstacle considering three different homology classes. However, it is not clear which functions are the best in guiding the search. Therefore, as described in the previous section, we only add necessary heuristic functions whenever we need them to solve the given path planning problem.

The following is the pseudo-code of the suggested Topology-based Multi-Heuristic A*. This search algorithm is fundamentally the same as Shared Multi-Heuristic A*. However, the search shifts from weighted A* to Shared Multi-Heuristic A* when it first adds additional heuristic function; when n becomes bigger than 0 (line 49). Whenever expanding the selected vertex, we check if it is a local minimum (line 42). This check basically compares if the priorities of all successor states are larger than the priority of state in question. Then we find and add a proper heuristic function by using $incHeuristic(s)$ function as described in the previous section. By doing so, we add and consider only few heuristic functions during the search on as-needed basis.

Topology-based Multi-Heuristic A* guarantees bounded optimality. If the algorithm terminates at line 10 while running as a weighted A*, the solution will be the same as in weighted A* and bounded by w_1. If the algorithm shifted to Shared Multi-Heuristic A* and the search terminates at line 17 or 22, then the optimality of the solution is bounded by $w_1 w_2$ same as in Shared Multi-Heuristic A*.

Algorithm *Topology-based Multi-Heuristic A:**

1) **procedure TbMHA*()**
2) $g(s_{goal}) = \infty,\ bp(s_{start}) = bp(s_{goal}) =$ null
3) $g(s_{start}) = 0$;
4) $OPEN_0 = \emptyset$;
5) $OPEN_0$.insert$(s_{start}, key(s_{goal}, 0))$;
6) $n = 0$
7) **while** $OPEN_0$ not empty
8) **if** $n = 0$
9) **if** $bp(s_{goal})$
10) terminate and return path pointed by $bp(s_{goal})$;
11) $s = OPEN_0.Top()$;
12) $expand(s, 0)$;
13) **else**
14) **for** $i = 1$ to n
15) **if** $OPEN_i.Minkey() \leq w_2 OPEN_0.Minkey()$
16) **if** $g(s_{goal}) \leq OPEN_i.Minkey()$
17) terminate and return path pointed by $bp(s_{goal})$;
18) $s = OPEN_i.Top()$;
19) $expand(s, i)$;
20) **else**
21) **if** $g(s_{goal}) \leq OPEN_0.Minkey()$
22) terminate and return path pointed by $bp(s_{goal})$;
23) $s = OPEN_0.Top()$;
24) $expand(s, 0)$;
25) **procedure key(s,i)**
26) return $g(s) + w_1 h_i(s)$;
27) **procedure expand(s,i)**
28) **for** $i = 0, \ldots, n$
29) $OPEN_i.remove(s)$;
30) **if** s is goal and $g(s) < g(s_{goal})$
31) $s_{goal} = s$;
32) **for each** $s' \in Succ(s)$
33) **if** s' was never visited
34) $g(s') = \infty;\ bf(s') =$ null;
35) **if** $g(s') > g(s) + c(s, s')$
36) **if** s' has not been expanded in the *anchor* search
37) insert/update s' in $OPEN_0$ with $key(s', 0)$
38) **if** s' has not been expanded in any *inadmissible* search
39) **for** $i = 1$ to n
40) **if** $key(s', i) \leq w_2 key(s', 0)$
41) insert/update s' in $OPEN_i$ with $key(s', i)$
42) **if** $key(s, 0) \geq key(s', 0)$ for each $s' \in Succ(s)$
43) $m = incHeuristic(s)$;
44) **for** $i = 1$ to m
45) $OPEN_{n+i} = \emptyset$;
46) **for each** $s' \in OPEN_0$
47) **if** $key(s', n+i) \leq w_2 key(s', 0)$
48) $OPEN_{n+i}$.insert$(s', key(s', n+i))$;
49) $n = n + m$;



(a) (b)

Fig. 4. Example paths of the simulations. The blue curves are the initial cable configurations and the green curves are the paths. The fixed base point is the end of blue curve at the bottom center of the map.

VI. Simulation Result

To demonstrate the performance of the suggested Topology-based Multi-Heuristic A*(TbMHA*) for path planning for a tethered mobile robot, we performed various simulations and compared the performance with weighted A*. (As the suggested TbMHA* guarantees bounded suboptimality, we choose weighted A*, instead of A*, to compare the performance, which also guarantees bounded suboptimality) Since the simulation result depends on the weighting parameters for both algorithms, we used the weighting value of $w = 10$ for the weighted A* and we chose $w_1 = \frac{10}{3}$ and $w_2 = 3$ for the Topology-based Multi-Heuristic A*. Then, both algorithms had the optimality bound of $w_1 w_2 = 10$. Both planning algorithm adapt travelling distance as cost, and weighted A* uses Euclidean distance to the goal as heuristic function, h. TbMHA* adapts the heuristic function considering homology class described in the previous section.

We built two simple 100×100 grid maps with 12 and 11 obstacles (Figure 4(a) and 4(b) respectively). And we choose the geometric center of each obstacle as its reference point. Using those, we ran several simulations with different cable length constraints and goal positions. The plots in Figure 4 show example paths generated by TbMHA*. We ran 18(three different cable lengths and three different goals for each two maps) simulations and compared the computation time and the number of expansions during the search. We compared evaluation parameters as the ratios of expansion and computation time of TbMHA* with respect to those of weighted A*.

$$r_e = \frac{\text{\# of expansions by TbMHA*}}{\text{\# of expansions by weighted A*}}$$
$$r_t = \frac{\text{computation time of TbMHA*}}{\text{computation time of weighted A*}}$$

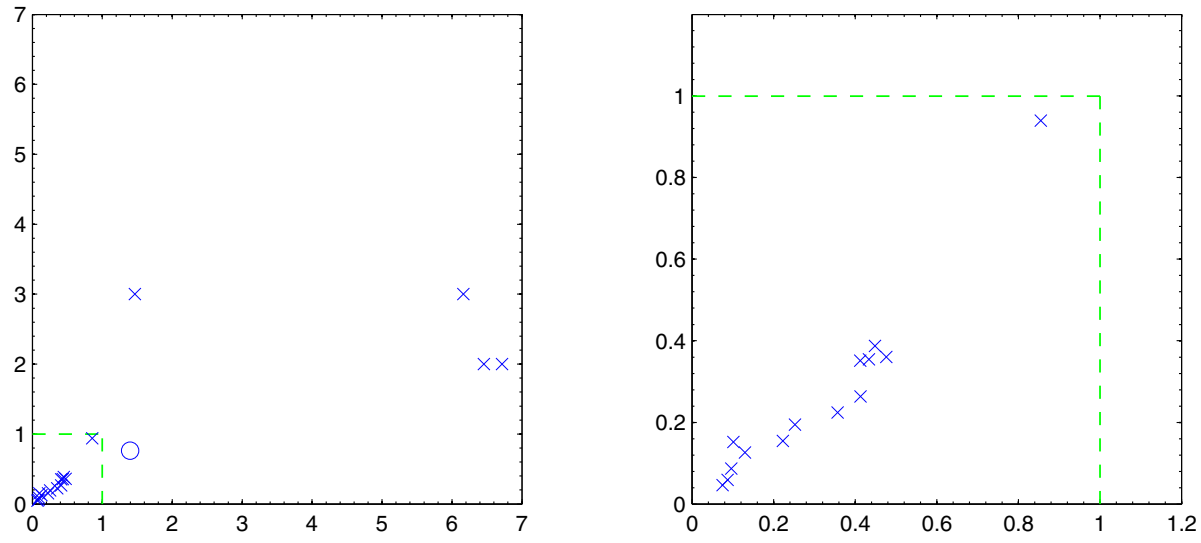
We plot these evaluation parameters in Figure 5. The x-axis represents the r_e and y-axis represents r_t. The $\times$ marks are the evaluation of each simulation and the circle is the mean of r_e and r_t for all 18 simulations. The inside of the dotted lines is the region where both parameters are less than 1, in other words, where the suggested Topology-based Multi-Heuristic A* works faster than weighted A* with less expansions. The plot in Figure 5(a) shows all 18 simulations. The mean values of the evaluation parameters are $\bar{r}_e = 1.4$ and $\bar{r}_t = 0.76$. According to these results, TbMHA* can find the path faster but needs more expansions. However, there are only four simulations, lying outside of the dotted line, where the weighted A* works better. All these simulations are the cases where we set the largest value of maximum cable length. So the cable is long enough that the cable constraint does not effect the resulting paths. All those four cases correspond to cases where the shortest path for the robot without the tether is the same as the path for tethered robot.

The Figure 5(b) is zoom-in on the lower left portion of Figure 5(a). The cases shown in Figure 5(b) are the simulations where cable length constraint did make the problem harder. If we consider only these 14 cases in Figure 5(b), the average evaluation parameters are $\bar{r}_e = 0.3116$ and $\bar{r}_t = 0.2643$. According to this result, TbMHA* finds the paths faster with less expansions than weighted A*.

To show the performance of TbMHA* on large problem, we considered an example of 300×300 grid map with 53 obstacles. (As the computational complexity increases exponentially with the number of obstacles, we increase the number of obstacles instead of increasing the size of the grid.) The Figure 6 shows the resulting path to the goal. Figure 7 shows the snapshots of the path. The blue curves are one of candidates of cable configurations at each instance which is achieved by $curveShorten()$ with path history appending initial cable configuration. At each instant the length of the blue curve is less than the maximum cable length. So all the configurations on this path are feasible and reachable. To find a path of this example, the weighted A* took $55240ses > 15hr$ while expanding about five million vertices. However, the Topology-based Multi-Heuristic A* solves the same problem in $3100sec < 52min$ while expanding only $580,000$ vertices. The evaluation parameter for this example was $r_t = 0.056$ and $r_e = 0.117$.

The above simulation result shows that the proposed Topology-based Multi-Heuristic A* works much more efficiently and faster than weighted A* to find the path of a tethered mobile robot in large and more complex environments. Also, the suggested TbMHA* has more benefits in computation time compared to number of expansions. The reason is that the proportion of nodes that require a call to $curveShorten()$, which is by far the most expensive procedure, is much higher in weighted A* than in TbMHA*. The reason for this is $curveShorten()$ runs when the value of l of the expanded node is greater than the cable length constraint L. This happens when nodes are around the local minima caused by cable length constraints. TbMHA* avoids or escapes these local minima by adapting proper heuristic function based on the topology class of cable and paths. So, the simulation results show that TbMHA* tends to expand less nodes around local minima or escapes from local minima faster than weighted A*.

We can achieve this result by considering the topology class of the cable and path to add proper heuristic functions



(a) (b)

Fig. 5. Comparing the computation load of the suggested Topology-based Multi-Heuristic A* to weighted A*. The x-axis is the ratio of expanded vertices, and y-axis is the ratio of computation time. Each $\times$ represents the evaluation parameters of each simulation and the circle is the average.

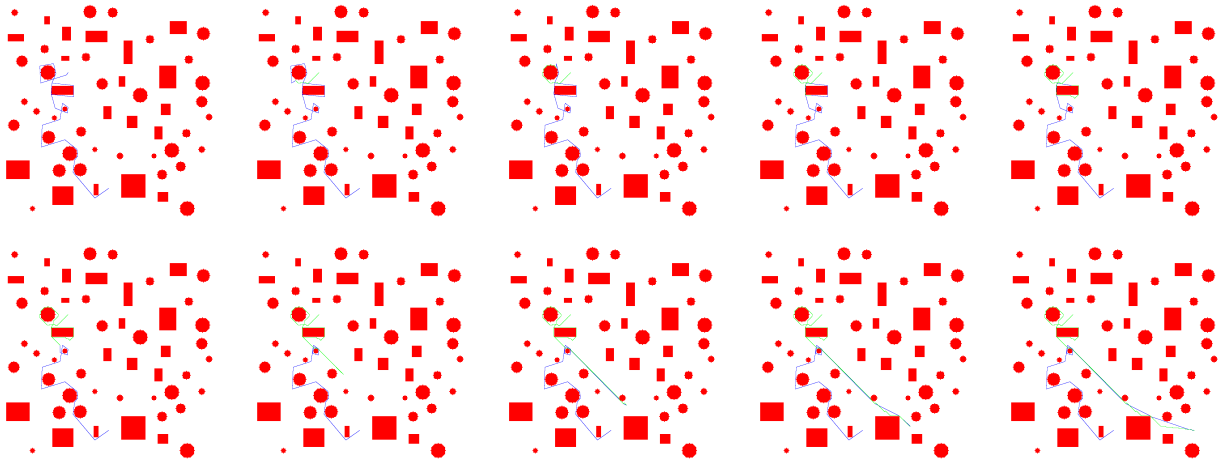


Fig. 7. Snapshots of the path in the example of Figure 6 from left to right, top to bottom. The blue curves are the expected cable configuration achieved by $curveShorten()$. At each instant, the length of the blue curve is shorter than the maximum cable length. The green curve is the path history of the robot.

to guide the search in the right direction to the goal. There are some exceptional examples where TbMHA* does not work better than weighted A*. If the cable is long enough, the problem is a simple path planning problem that doesn't require reasoning over tether.

VII. Conclusion

In this paper we described the algorithm to solve the path planning problem for a tethered mobile robot, which is connected to the fixed base by a flexible fixed-length cable. We formulate this path planning as a graph-search and propose Topology-based Multi-Heuristic A* which is a variation of Multi-Heuristic A*. The algorithm guarantees the suboptimality bound on the solution while decreasing the computation time dramatically. In our approach, we considered the topology class of the path and cable to find proper heuristic functions that help the search escape local minima. We demonstrated the performance of the proposed algorithm in simulations. One of the future directions is to find other constraints that cause local minima in planning for a tethered robot and design proper heuristic functions.

Acknowledgment

This research was sponsored by the ONR DR-IRIS MURI grant N00014-09-1-1052 and ONR ANTIDOTE MURI grant N00014-09-1-1031.

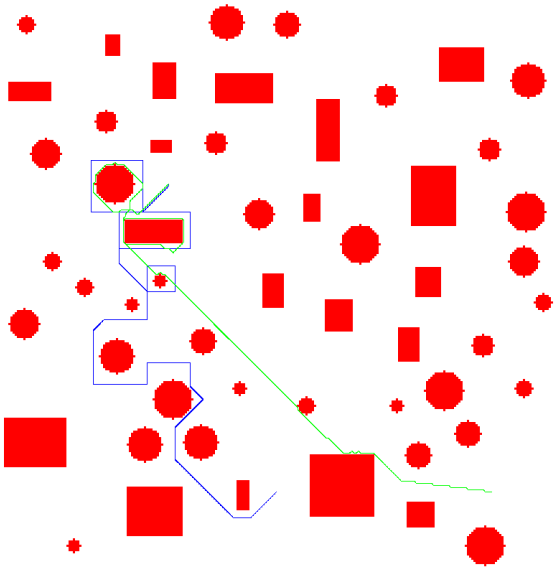


Fig. 6. Example path of the simulation in 300×300 grid map with 53 obstacles. The blue curves are the initial cable configuration and the green curve is the path. The fixed base point is the end of blue curve at the bottom center of the map.

REFERENCES

[1] P. Abad-Manterola, I.A.D. Nesnas, and J.W. Burdick. Motion planning on steep terrain for the tethered axel rover. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 4188–4195, May 2011.

[2] Sandip Aine, Siddharth Swaminathan, Venkatraman Narayanan, Victor Hwang, and Maxim Likhachev. Multi-heuristic a*. In *Proceedings of Robotics: Science and Systems*, Berkeley, USA, July 2014.

[3] Subhrajit Bhattacharya, Vijay Kumar, and Maxim Likhachev. Search-based path planning with homotopy class constraints. In *Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence*, Atlanta, Georgia, July 11 2010.

[4] Subhrajit Bhattacharya, Maxim Likhachev, and Vijay Kumar. Identification and representation of homotopy classes of trajectories for search-based path planning in 3d. In *Proceedings of Robotics: Science and Systems*, 27-30 June 2011.

[5] Stephen Cass. DARPA Unveils Atlas DRC Robot, Jul 2013. http://spectrum.ieee.org/automaton/robotics/humanoids/darpa-unveils-atlas-drc-robot.

[6] P. Chirarattananon, K.Y. Ma, and R.J. Wood. Adaptive control for takeoff, hovering, and landing of a robotic fly. In *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*, pages 3808–3815, Nov 2013.

[7] D. Grigoriev and A. Slissenko. Polytime algorithm for the shortest path in a homotopy class amidst semi-algebraic obstacles in the plane. In *ISSAC '98: Proceedings of the 1998 international symposium on Symbolic and algebraic computation*, pages 17–24, New York, NY, USA, 1998. ACM.

[8] Erico Guizzo. Fukushima Robot Operator Writes Tell-All Blog, Aug 2011. http://spectrum.ieee.org/automaton/robotics/industrial-robots/fukushima-robot-operator-diaries.

[9] Allen Hatcher. *Algebraic Topology*. Cambridge Univ. Press, 2001.

[10] J. Hershberger and J. Snoeyink. Computing minimum length paths of a given homotopy class. *Comput. Geom. Theory Appl*, 4:331–342, 1991.

[11] S. Hert and V. Lumelsky. The ties that bind: motion planning for multiple tethered robots. In *Robotics and Automation, 1994. Proceedings., 1994 IEEE International Conference on*, pages 2734–2741 vol.4, 1994.

[12] Daehie Hong, Steven A. Velinsky, and Kazuo Yamazaki. Tethered mobile robot for automating highway maintenance operations. *Robotics and Computer-Integrated Manufacturing*, 13(4):297 – 307, 1997.

[13] Takeo Igarashi and Mike Stilman. Homotopic path planning on manifolds for cabled mobile robots. In David Hsu, Volkan Isler, Jean-Claude Latombe, and MingC. Lin, editors, *Algorithmic Foundations of Robotics IX*, volume 68 of *Springer Tracts in Advanced Robotics*, pages 1–18. Springer Berlin Heidelberg, 2011.

[14] Soonkyum Kim, S. Bhattacharya, and V. Kumar. Path planning for a tethered mobile robot. In *Robotics and Automation (ICRA), 2014 IEEE International Conference on*, pages 1132–1139, May 2014.

[15] Soonkyum Kim, Subhrajit Bhattacharya, Hordur Heidarsson, Gaurav Sukhatme, and Vijay Kumar. A topological approach to using cables to separate and manipulate sets of objects. In *Proceedings of the Robotics: Science and System (RSS)*, Syndey, Australia, June 24-28 2013.

[16] Nathan Michael, Shaojie Shen, Kartik Mohta, Yash Mulgaonkar, Vijay Kumar, Keiji Nagatani, Yoshito Okada, Seiga Kiribayashi, Kazuki Otake, Kazuya Yoshida, Kazunori Ohno, Eijiro Takeuchi, and Satoshi Tadokoro. Collaborative mapping of an earthquake-damaged building via ground and aerial robots. *Journal of Field Robotics*, 29(5):832–841, 2012.

[17] A.A.F. Nassiraei, Y. Kawamura, A. Ahrary, Y. Mikuriya, and K. Ishii. Concept and design of a fully autonomous sewer pipe inspection mobile robot "kantaro". In *Robotics and Automation, 2007 IEEE International Conference on*, pages 136–143, April 2007.

[18] Ira Pohl. Heuristic search viewed as path finding in a graph. *Artificial Intelligence*, 1(34):193 – 204, 1970.

[19] K.S. Pratt, R.R. Murphy, J.L. Burke, J. Craighead, C. Griffin, and S. Stover. Use of tethered small unmanned aerial system at berkman plaza ii collapse. In *Safety, Security and Rescue Robotics, 2008. SSRR 2008. IEEE International Workshop on*, pages 134–139, 2008.

[20] Iddo Shnaps and Elon Rimon. Online coverage by a tethered autonomous mobile robot in planar unknown environments. In *Proceedings of Robotics: Science and Systems*, Berlin, Germany, June 2013.

[21] M.M. Tanner, J.W. Burdick, and I.A.D. Nesnas. Online motion planning for tethered robots in extreme terrain. In *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, pages 5557–5564, May 2013.

[22] Benjamn Tovar, Fred Cohen, and Steven M. LaValle. Sensor beams, obstacles, and possible paths. In *Workshop on the Algorithmic Foundations of Robotics*, pages 317–332, 2008.

[23] P.G. Xavier. Shortest path planning for a tethered robot or an anchored cable. In *Robotics and Automation, 1999. Proceedings. 1999 IEEE International Conference on*, volume 2, pages 1011–1017 vol.2, 1999.

[24] Ning Xu, Peter Brass, and Ivo Vigan. An improved algorithm in shortest path planning for a tethered robot. Technical report.