Sampling-Based Planning Algorithms for Multi-Objective Missions

Md Mahbubur Rahman[1], Leonardo Bobadilla[1], Brian Rapp[2]

***Abstract*— Multiple objective navigation is commonly found in practice, where a path for an autonomous vehicle needs to be generated to simultaneously optimize a number of different objectives such as distance, safety, and visibility. Objectives can be weighted to solve a single objective optimization problem but appropriate weights may not be known a priori. In this paper, we formulate a series of missions for a group of vehicles that need to keep connectivity among themselves, surveil a group of targets, and minimize path lengths. These problems are solved by extending optimal sampling-based algorithms (RRT*) to support multiple objectives, non-additive costs and cooperative conditions. We present several increasingly complicated missions in obstacle-filled environments to illustrate and compare our ideas with existing methods.**

I. Introduction

Motion planning for an autonomous system, at its simplest, involves finding a trajectory that avoids obstacles and respects differential constraints [1], [13]. An objective function, such as travel time or length, will then be optimized [10]. However, for many real-world applications, multiple objectives may be relevant. It might be desirable to conserve fuel, provide a comfortable ride, and avoid locations with a high risk for accidents. Trade-offs are likely to be involved when optimizing such disparate objectives. The objectives can be combined using a weighted function, but the appropriate weights may not be known a priori. Additionally, simple multi-objective combinatorial problems such as a 2-criteria shortest path are proven to be NP-Complete [4].

In some contexts like military operations, visibility becomes an important aspect of the objective. One typically wishes to maintain visibility with friendly units or targets of observation, while avoiding visibility by potential enemies. One such sample environment is shown in Figure 1 where a vehicle needs to monitor two blue units, avoid one enemy firing range and minimize path length from the purple starting location to the yellow goal area. Unless a weighted objective function can be specified for such a mission, it may be necessary to sample the problem space to obtain estimates for the weight of each objective. Integrating this process into the motion planning algorithm makes it possible to attempt an optimization of all the objectives simultaneously. This paper presents such a method and analyzes its applicability to certain multiple-objective motion planning problems.

Our work is motivated by the problem posed in [17] that requires one to determine the positions of a group of *units* that need to perform surveillance over a group of *targets* while simultaneously minimizing exposure to *enemy* units. In [17],

[1]M. Rahman and L. Bobadilla are with the School of Computing and Information Sciences, Florida International University, Miami, FL, 33199, USA. {mrahm025}@fiu.edu bobadilla@cs.fiu.edu

[2]B. Rapp is with the United States Army Research Lab. brian.m.rapp2.civ@mail.mil

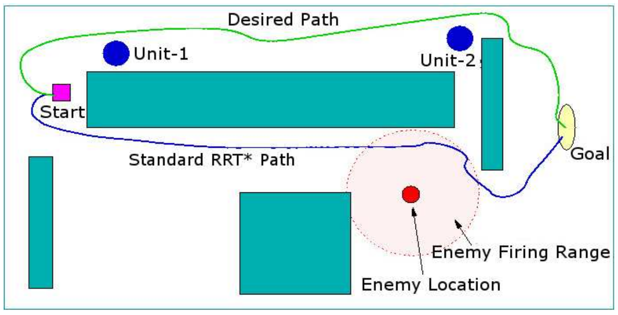

Fig. 1. A Dubins vehicle is assigned to observe two blue circular units while avoiding obstacles and an enemy unit throughout its path from the start to the goal location. Multi-criteria optimization is required to find the shortest path that resembles the green trajectory. Existing sampling-based path planning may give an incorrect path like the blue one.

the units and targets are static. The problem is formulated as a multi-objective correlated geometric optimization problem and it is solved through Markov chain Monte Carlo methods. Our goal is to extend this family of problems by allowing the units, targets, and other teammates to move in an environment with obstacles while also attempting to optimize other variables such as completion time, clearance from obstacles, and communications. We will frame this family of problems as multi-objective optimal path planning problems.

The contributions of this paper are the following: 1) We formulate a series of increasingly hard missions that require a group of vehicles to monitor targets while keeping network connectivity with other teammates; 2) We present extensions to optimal sampling-based motion planning algorithms (such as RRT* [10]) to consider multiple objectives, non-cooperative, and cooperative scenarios; and 3) We present several case studies to illustrate the application of our ideas.

The rest of the paper is organized as follows: Section II presents a discussion about existing solutions along with their usefulness and shortcomings in the context of our problem. Section III presents the preliminaries and the problem formulation. Section IV introduces algorithms to solve the class of multi-objective path planning problems. We then analyze the complexity and behavior of the proposed methods in Section V. Section VI presents illustrative case studies of several field missions. Finally, conclusions and directions for further research are discussed in Section VII.

II. Related Work

Multi-objective optimization has been studied widely for many years in different domains. A solution for a multi-objective problem can be based on scalarization of objectives where the objectives are weighted according to their priority and added to form a single scalar value [18]. Related to our ideas are the methods proposed in [18] for multi-criteria shortest path computations that compute a number of possible paths from source to goal and then choose a Pareto efficient path [19]. Although our methods are initially motivated by the techniques presented in [18], we propose a solution that works by incrementally building and rapidly exploring random trees, as in RRT* [11], [10]. We are also focused on generating a single path optimizing all of the objectives.

Our ideas are also connected with the method described in [20] that modifies the RRT* algorithm [11] in order to adopt multiple criteria during expansion. In contrast with this approach our algorithm is able to produce a single path in terms of multiple costs rather than a number of Pareto optimal paths. Additionally, the weights in the *Tchebycheff* method and the weighted sum method [22] used in [20] can be difficult to tune as different objectives have different costs. Instead, our work solves the multi-criteria optimization problem by normalizing the objectives using the *Utopian* optimal vector [22].

Closely related to our work is [16] where the scalarization of objectives was also used to get a single objective function. This work starts with a known graph and uses A^* search. A modified Bellman-Ford method is used in [3] to assign a normalized label to each node in order to find a multi-criteria shortest path. Another solution that prioritizes one objective over another is presented in [5]. This type of hierarchization biases the path mostly towards the top priority objective. Also, this solution is limited to a 2D grid and cannot be applied in a higher dimensional configuration space.

Another stream of research proposes visibility-based solutions to monitor a number of units in an environment. A modified Traveling Salesman (TSP) algorithm was used by [15] where the problem was solved without optimizing multiple criteria.

III. Preliminaries

Consider an environment $\mathcal{W} = \mathbb{R}^2$ where a mission is taking place. Let $\mathcal{O} = \{O_1, O_2, \ldots, O_\zeta\}$ be the set of obstacles which are modeled as polygons. The collision-free space is defined as $E = \mathcal{W} \setminus \mathcal{O}$.

Let $\mathcal{A} = \{A_1, A_2, \ldots, A_k\}$ be a number of vehicles which are deployed, with configuration spaces $\mathcal{C}_1, \mathcal{C}_2, \ldots, \mathcal{C}_k$, respectively. The servicing vehicles move inside the bounded environment E as car-like robots. Therefore the configuration of each vehicle A_i is defined as $\mathcal{C}_i = E \times S^1$. These vehicles are like Dubins cars [2], and a given vehicle A_i must satisfy differential constraints and dynamics defined as $\dot{x}_i = u_s^i \cos\theta, \dot{y}_i = u_s^i \sin\theta$, and $\dot{\theta}_i = \frac{u_s^i}{L^i} \tan u_\phi^i$, where u_s^i is the forward speed and u_ϕ^i is the steering angle of the vehicle [6]. There are a number of mobile units deployed in E that can move freely in the world and are modeled as point robots without rotation. Accordingly, the configuration for a mobile unit is defined as, $B_i = (x, y) \in E$.

Let X be the state space for one servicing vehicle and for simplicity assume $X = \mathcal{C}_i$. There are n cost functions $l_1, l_2, \ldots, l_n$ where $l_i : X \to \mathbb{R}^{\geq 0}$. Each state $x \in X$ of the robot is associated with multiple objective costs. Accordingly, a vector valued function $L : X \to \mathbb{R}^n$ assigns n cost labels to a particular state x and is defined as,

$$L(x) = (l_1(x),\ l_2(x),\ \ldots,\ l_n(x)) \text{ where } x \in X \quad (1)$$

Let $X_{obs} = \{x \in X : x \cap O \neq \emptyset \ where \ O \in \mathcal{O}\}$ be the obstacle state space. The collision-free state space is then $X_{free} = X \setminus X_{obs}$. We define the initial configuration state of the vehicle as, $x_I \in X_{free}$, and a set of goal states as $X_G \subset X_{free}$.

Let σ be an obstacle-free feasible trajectory that starts from x_I and leads a vehicle to its goal region X_G.

Problem 1: Generation of multi-cost optimal paths for servicing vehicles.
Given an initial configuration x_I and a set of goal states X_G, find a collision-free continuous trajectory, $\sigma : [0, t] \to X_{free}$ for some time t, such that $\sigma(0) = x_I$ and $\sigma(t) \subset X_G$ attempts to optimize L.

We are also interested in finding paths in cooperative environments. For these scenarios, we will have a number of vehicles deployed inside the environment which have a common objective and perform cooperative behavior. An example of such behavior is maintaining visibility for communication.

Problem 2: Generation of multi-objective optimal paths for cooperative robots.
Given a set of friendly robot vehicles $A_1, A_2, \ldots, A_k$, with a common cost l_c, find a set of collision-free continuous trajectories, $\sigma_1, \sigma_2, \ldots, \sigma_k$, that solve Problem 1 and best optimize $l_c : X_1 \times X_2 \times \cdots \times X_k \to \mathbb{R}^{\geq 0}$.

IV. Methods

Our ideas for multi-objective optimal motion planning are modifications to the algorithms proposed by Karaman and Frazzoli [10], [8]. Algorithm 1 presents a procedure that computes a trajectory σ from x_I to X_G based on the sampling algorithm RRT* [10]. In our modified version, we start a tree structure $\mathcal{T}$ (line 1) and continue to expand it by sampling random states.

Algorithm 1 MultiObjectiveRRTStar(x_{init})

```
 1: $\mathcal{T}.init(x_{init})$
 2: for $i \leftarrow 1$ to $K$ do
 3:     $x_{rand} \leftarrow RandomConfig()$
 4:     $x_{nearest} \leftarrow NearestNode(\mathcal{T}, x_{rand})$
 5:     $x_{new} \leftarrow Steer(x_{rand}, x_{nearest})$
 6:     if $ObstacleFree(x_{new}, x_{nearest}, \mathcal{O})$ then
 7:         $x_{opt} \leftarrow ChooseParent(x_{nearest}, x_{new}, \mathcal{T})$
 8:         $L \leftarrow (l^a(x_{opt}) + c(x_{opt}, x_{new}), \frac{l^{na}(x_{opt})+l^{na}(x_{new})}{2})$
 9:         $\mathcal{T} \leftarrow InsertNode(x_{opt}, x_{new}, \mathcal{T})$
10:         $\mathcal{T} \leftarrow ReWire(\mathcal{T}, x_{new})$
11:     end if
12: end for
13: $return\ \mathcal{T}$
```

In addition to the state cost $L(x)$, we assign edge cost, $(c_1, c_2, \ldots, c_n)$ to an edge $\tilde{x}_{ij}$ that connects two successive states x_i and x_j. Here, a cost function c_i for a particular objective i is defined as, $c_i : X \times X \to \mathbb{R}^{\geq 0}$.

Additive/Non-Additive Cost: There are two types of costs assigned to the nodes in a tree. An additive cost l^a is cumulative and depends on the parent's cost and the arc cost that connects it to its parent. On the other hand a non-additive cost l^{na} is independent and calculated based on the state (e.g. visibility, safety, clearance).

The primitives of the Algorithm 1 are similar to other sampling-based motion planning algorithms [9], [12], [14]:

Sampling: A tree is initialized in line 1 of Algorithm 1. In line 3 the method $RandomConfig()$ samples a random configuration $x_{rand} \in X_{free}$.

Nearest Node: In line 4, $x_{nearest} = NearestNode(\mathcal{T}, x_{rand})$ returns the node $x_{nearest}$ of the tree $\mathcal{T}$ that is nearest to the sampled node x_{rand} in terms of a distance metric.

Steer: The method $Steer(x_1, x_2)$ is used to solve control inputs u_s and u_ϕ and produces x_{new} from x_{rand} for a dynamic control system.

Collision Checking: Method $Obstaclefree()$ checks whether a path from x_{new} to $x_{nearest}$ avoids all the obstacles $\mathcal{O}$ and is therefore collision-free.

We completely modified the $ChooseParent()$ and $ReWire()$ methods of standard RRT* to support multiple objectives.

A. Choosing a Parent

Algorithm 2 is used to select the best parent x_{opt} of the newly sampled node x_{new} in terms of a multi-objective optimization. We choose a set of candidate nodes, X_{near} based on a nearness metric (e.g. point distance in Euclidean space) using method $NearestNeighbours()$ in line 1.

Domination and Non-Domination: A node $x_j \in X_{near}$ is dominated by another node $x_i \in X_{near}$ (defined as $x_i \prec x_j$) if and only if all the n costs of x_i are better than those of x_j. Accordingly the cost vector $L(x_j)$ is dominated by cost vector $L(x_i)$ (denoted by $L_i \prec L_j$).

$$x_i \prec x_j \;\Leftrightarrow\; \forall k, l_k(x_i) \le l_k(x_j);\; where\ 1 \le k \le n. \quad (2)$$

Therefore the set of dominated nodes D_X is defined as,

$$D_X = \{x_j \in X_{near} | \exists i\ x_i \prec x_j\}. \quad (3)$$

We only consider the nodes that are not dominated by another node. Accordingly the set of non-dominated nodes P_X comprises the *Pareto frontier*,

$$P_X = X_{near} \setminus D_X. \quad (4)$$

We must select a node from the *Pareto* set P_X that best optimizes all the costs. Therefore we first calculate the minimum cost tuple L^*,

$$L^* = (\min_{1\le i\le |P_X|} l_1(x_i), \dots, \min_{1\le i\le |P_X|} l_n(x_i)) \quad (5)$$

Next, we calculate the arc costs, $C = (c_1, c_2, \dots, c_n)$ associated with an arc, $\tilde{x}_{j,new}$, which connects the new node x_{new} to a nearest non-dominated node $x_j \in P_X$. The minimum costs c_i^* designated to an arc for each objective i are calculated and the optimum cost tuple for arc costs is:

$$C^* = (\min_{1\le i\le |P_X|} c_1(x_i, x_{new}), \dots, \min_{1\le i\le |P_X|} c_n(x_i, x_{new})) \quad (6)$$

A weighting variable $\alpha_i = [0, 1]$ is assigned to each objective i to control the effect of the objective cost on a path to be planned where $\sum_{1\le i\le n} \alpha_i = 1$. This weighting variable is different from that used in scalarization methods [20], [22] and is only used to define priority. Our algorithm is capable of running without α unlike the methods described in [20], [22]. Accordingly, we select a node $x_j \in P_X$ as the parent (line 7) that yields the minimum *normalized* costs to come to x_{new},

$$\underset{x_j\in P_X}{\operatorname{argmin}} \sum_{i=1}^{n} \alpha_i \left[\frac{l_i(x_j)}{l_i^*} + \frac{c_i(x_j, x_{new})}{c_i^*}\right] \quad (7)$$

Algorithm 2 ChooseParent($x_{min} \leftarrow x_{nearest}, x_{new}, \mathcal{T}$)

```
1:  $X_{near} \leftarrow NearestNeighbours(x_{new}, \mathcal{T}, \mathcal{O})$
2:  $D_X \leftarrow \{x_i \in X_{near} | \exists j\ \forall k\ l_k(x_i) \le l_k(x_j)\}$
3:  $P_X \leftarrow X_{near} \setminus D_X$
4:  $L^* \leftarrow (l_1^*,\ l_2^*,\ \dots,\ l_n^*)$
5:  $C^* \leftarrow (c_1^*,\ c_2^*,\ \dots,\ c_n^*)$
6:  for $x \in P_X$ do
7:      if $\sum_i \left[\frac{l_i(x)}{l_i^*} + \frac{c_i(x,x_{new})}{c_i^*}\right] < \sum_i \left[\frac{l_i(x_{min})}{l_i^*} + \frac{c_i(x_{min},x_{new})}{c_i^*}\right]$ then
8:          $x_{min} \leftarrow x$
9:          $L(x_{min}) \leftarrow L(x)$
10:     end if
11: end for
12: $return\ x_{min}$
```

B. Updating the Tree

Different Costs: In line 8 of Algorithm 1, we assign a cost label L defined in (1) to the current node x_{new} once a parent x_{opt} is selected for that node. Suppose we have k additive costs and $n-k$ non-additive costs. *Additive* costs $l^a(x_{new})$ like *distance* are allocated by combining the cost of a parent and the arc cost to resembles the total cost from the root node.

$$l_i^a(x_{new}) = l_i^a(x_{opt}) + c_i(x_{opt}, x_{new}); \forall\ i,\ 1 \le i \le k. \quad (8)$$

Non-additive costs l^{na} (such as *visibility*) do not propagate, and require in-place computation. The parent's cost and the current node's cost (computed in-place) are averaged to assign the cost $l^{na}(x_{new})$.

$$l_j^{na}(x_{new}) = \frac{l_j^{na}(x_{opt}) + l_j^{na}(x_{new})}{2}; \forall\ j,\ k+1 \le j \le n. \quad (9)$$

Node Insertion: Once we know the parent x_{opt} of the newly sampled node x_{new}, we add it to the tree $\mathcal{T}$ along with the corresponding edge $\tilde{x}_{opt,new}$ and arc cost $c(\tilde{x})$. The modified costs $L(x_{new})$ for the involved node are also updated in this step (line 9 in Algorithm 1).

Algorithm 3 ReWire($\mathcal{T}, x_{new}$)

```
1: $X_{near} \leftarrow NearestNeighbours(x_{new}, \mathcal{T}, \mathcal{O})$
2: for $x \in X_{near}$ do
3:     if $\bigwedge_{i=1}^{k}(l_i^a(x_{new}) + c(x_{new}, x) \leq l_i^a(x))$ and $\bigwedge_{j=k+1}^{n} l_j^{na}(x_{new}) \leq l_j^{na}(x.parent)$ then
4:         $x.parent \leftarrow x_{new}$
5:         $L(x) \leftarrow (l^a(x_{new}) + c(x_{new}, x), \frac{l^{na}(x_{new})+l^{na}(x)}{2})$
6:         $\mathcal{T}.update(x)$
7:     end if
8: end for
9: $return\ \mathcal{T}$
```

C. Refining Connections

Algorithm 3 is used to refine the existing connections in the neighborhood of a newly connected vertex x_{new}. This procedure makes x_{new} the parent of the neighboring nodes $x \in X_{near}$ if this yields better costs compared to the costs incurred through the current parent. Method $NearestNeighbours()$ in line 1 computes the nearest node set X_{near}. A neighbor $x \in X_{near}$ is connected through the newly added node x_{new} if the following two conditions are satisfied:

$$\forall\ i, 1 \leq i \leq k;\ l_i^a(x_{new}) + c(x_{new}, x) \leq l_i^a(x) \tag{10}$$

$$\forall\ j, k+1 \leq j \leq n;\ l_j^{na}(x_{new}) \leq l_j^{na}(x.parent) \tag{11}$$

This implies that we make x_{new} the parent of the neighboring node x if a) the connecting cost reduces the existing cost $l^a(x)$ as shown in (10) and b) x_{new} provides a better cost than $x.parent$ in terms of non-additive costs $l^{na}(x)$. For example, if a node's parent has better visibility or safety than x_{new}, then making x_{new} the new parent is not desirable. Algorithm 3 terminates by updating the cost vector $L(x)$ and tree $\mathcal{T}$ in lines 5 and 6 respectively.

D. Cooperative Path Generation for Multiple Robots

A cooperative scenario is developed when multiple friendly vehicles want to communicate while optimizing their own objectives. We propose Algorithm 4 where two cooperative trees $\mathcal{T}_u$ and $\mathcal{T}_v$ expand in parallel while affecting each other. Lines $4-10$ from Algorithm 1 are used for their expansion (called in lines 4 and 6 of Algorithm 4). A function $l_c : X \times X \to \{0, 1\}$ is defined that checks whether the two newly sampled vertices x_u, x_v of the two trees cooperate (in line 8). A reward function $\omega_k : X \times X \to \mathbb{R}^{\geq 0}$ is defined that helps to decrease the costs for objective k if the two nodes cooperate (lines $9-10$). Otherwise, a penalty function $\rho_k : X \times X \to \mathbb{R}^{\geq 0}$ is used to increase each of the costs (lines $12-13$). In the cases of non-cooperative planning scenarios, the reward and penalty functions are swapped. Finally, $PropagateCost()$ is used in lines $15-16$ to pass the effect of the updated cost down towards all the nodes throughout the child chain.

V. Analysis

Running Time Analysis: The main modification in our proposed model is implemented in Algorithms 2 and 3 compared to the standard RRT* [10], [8]. In order to take care of n objective costs, both of them run n times more than the standard RRT* algorithm. Therefore the running time of our multi-objective RRT* algorithm is $O(n \cdot RRT^*)$, which is a constant multiple of the running time of the standard RRT*.

Algorithm 4 RRTStarCooperative(x_u^{init}, x_v^{init})

```
1: $i \leftarrow 0$
2: while $i \leq K$ do
3:     $x_u \leftarrow RandomConfig()$
4:     $\mathcal{T}_u \leftarrow MultiRRT^*\_Expansion(\mathcal{T}_u, x_u)$
5:     $x_v \leftarrow RandomConfig()$
6:     $\mathcal{T}_v \leftarrow MultiRRT^*\_Expansion(\mathcal{T}_v, x_v)$
7:     if $x_u \neq NULL$ and $x_v \neq NULL$ then
8:         if $l_c(x_u, x_v)$ then
9:             $\forall k, 1 \leq n, l_k(x_u) \leftarrow l_k(x_u) - \omega_k(x_u, x_v)$
10:            $\forall k, 1 \leq n, l_k(x_v) \leftarrow l_k(x_v) - \omega_k(x_u, x_v)$
11:        else
12:            $\forall k, 1 \leq n, l_k(x_u) \leftarrow l_k(x_u) + \rho_k(x_u, x_v)$
13:            $\forall k, 1 \leq n, l_k(x_v) \leftarrow l_k(x_v) + \rho_k(x_u, x_v)$
14:        end if
15:        $PropagateCost(x_u, \mathcal{T}_u)$
16:        $PropagateCost(x_v, \mathcal{T}_v)$
17:    end if
18: end while
```

Proposition 5.1: $ChooseParent()$ selects a non-dominated optimal parent.

Proof: (sketch) It is trivial that we select a non-dominated node as a parent as we select it from the non-dominated set P_X according to line 6 of Algorithm 2.

We now prove the optimality by contradiction. Let $ChooseParent()$ select x_p as a parent of a node x_{new} which does not provide the optimal costs and let there be a parent x'_p that provides the optimal costs such that: $\sum_{i=1}^{n}\left[\frac{l_i(x'_p)}{l_i^*} + \frac{c_i(x'_p, x_{new})}{c_i^*}\right] \leq \sum_{i=1}^{n}\left[\frac{l_i(x_p)}{l_i^*} + \frac{c_i(x_p, x_{new})}{c_i^*}\right]$ This contradicts definition (7), where we select the node as a parent which minimizes the above cost. Therefore, x_p is chosen over x'_p implies that x_p and x'_p cannot be different. This essentially proves that the method $ChooseParent()$ selects the optimal non-dominated parent in terms of multi-objective cost. ■

Proposition 5.2: $ReWire()$ computes the optimal parent in terms of multiple cost vector L in a particular tree $\mathcal{T}$.

Proof: (sketch) This is trivial from the conditions in (10) and (11) where the current parent of a node $x \in X_{near}$ is changed to the newly sampled node x_{new} if and only if x_{new} is better in all n cost metrics l_k where $1 \leq k \leq n$. See line 3 of Algorithm 3. ■

Proposition 5.3: A solution path σ is a non-dominated solution for a particular MultiObjectiveRRT* tree $\mathcal{T}$.

Proof: (sketch) It is a necessary condition that a subpath of an optimal path generated by RRT* is also optimal. From propositions 5.1 and 5.2, we guarantee that the local sub-solutions are non-dominated. This implies that once a tree $\mathcal{T}$ is generated, the corresponding path σ is a non-dominated path. ■

VI. Case Studies

We developed an implementation of multiple cost vectors on top of the MIT SMP library [7] that was originally devel-

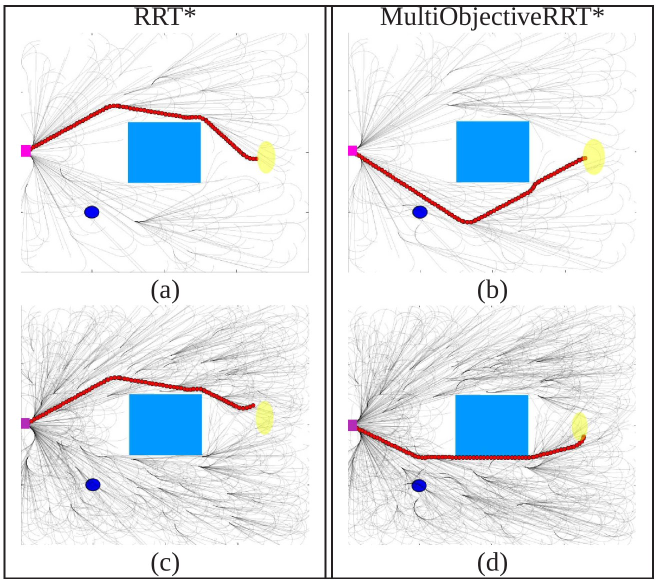

Fig. 2. Trajectory finding for a car-like vehicle while monitoring the blue circular landmark: (a) Standard RRT* tree and trajectory after 500 iterations. The purple rectangle is the initial position and the yellow region is the goal. (b) Our MultiObjectiveRRT* tree and trajectory after 500 iterations. (c) Standard RRT* tree after 3000 iterations. (d) Our MultiObjectiveRRT* tree after 3000 iterations.

oped by the authors of RRT*. We modeled a problem based on Figure 1 where a number of point robots $B_1, B_2, \ldots, B_m$ must be served through visible light communication [21] by a number of serving vehicles (Dubins cars) $A_1, A_2, \ldots, A_k$. An optimal trajectory σ maintains visibility while optimizing the traveling distance from the starting location (purple) to the goal location (yellow). Additionally, there might be a number of friendly units we want to observe and a number of enemy units we want to avoid.

A. Case Study 1: Single Unit Visibility and Patrolling

In Figure 2, we present a case where a single vehicle is present to serve the blue unit while avoiding an obstacle $\mathcal{O}$ at the middle of the map. An optimal trajectory is required to minimize the traveling distance and maximize LoS visibility to the unit.

Figure 2(a) and (c) are the resulting tree and trajectory (red) that are computed by the standard RRT* [10] algorithm after 500 and 3000 iterations respectively. We then apply our algorithm and the results are shown in Figure 2(b) and (d). Clearly the red trajectory σ of Figure 2(b) is better than Figure 2(a) as it is more visible to the unit. Similarly the path converges to optimality in terms of both length and visibility as shown in Figure 2(d) after 3000 iterations while the standard RRT* in Figure 2(c) only optimizes the length.

B. Case Study 2: Two Vehicles, Two Units

In Figure 3, we present a more complex case where two blue units B_1 and B_2 need to be monitored by two Dubins vehicles A_1 and A_2 while reaching their respective goal regions. Figure3(a), (c) and (e) are the results from our algorithm and Figure 3(b), (d) and (f) are the outcomes of the weighted sum and *Tchebycheff* methods used in the state of art [22], [20] solutions.

The path for A_1 (green) first covers unit B_1 and turns towards the unit B_2 to optimize visibility and path length after 500 iterations in Figure 3(a). Similarly A_2's path (red) makes a turn to maximize visibility and then follows the optimal distant path.

The weighed sum and Tchebycheff [22], [20] methods both use scalarization of objectives and show similar characteristics as shown in Figure 3(b). In Table I, we provide a numerical comparison of our method with these prevalent techniques. We found that these methods frequently become biased towards a particular objective. A_1's trajectory optimizes path length while A_2's trajectory is mostly out of the visibility range of unit B_2.

In Figure 3(c), we present the result after 2000 iterations which achieves good compromisation between length and visibility. However, the weighted sum method generates longer trajectory (red one) with a slightly increased visibility as presented in Table I.

Finally we ran the methods for 5000 iterations as shown in Figures 3(e) and (f). Our method converged to a near optimal non-dominated solution. On the other hand, the Tchebycheff method generated the green trajectory with a slightly increased visibility (0.47 vs 0.53) and a longer path

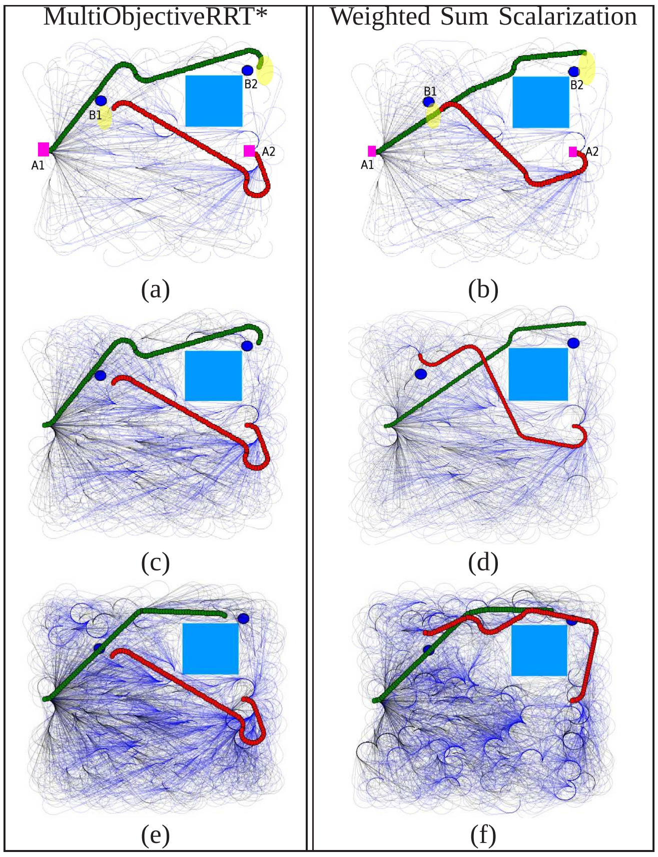

Fig. 3. Dubins car trajectory finding for two car-like robots. Vehicles start from two small rectangular positions (purple colored): (a) Our MultiObjectiveRRT* tree and trajectory after 500 iterations. (b) RRT* tree with weighted sum (scalarization) method after 500 iterations. (c) Our Multi RRT* at 2000 iterations.(d) Tchebycheff (scalarization) method after 2000 iterations. (e) Our Multi RRT* after 5000 iterations.(f) Tchebycheff (scalarization) method after 5000 iterations.

TABLE I
TRAJECTORY ANALYSIS IN TERMS OF MULTIPLE OBJECTIVES

	Iteration	Objective	Tchebycheff	Our Multi RRT*
$Vehicle\ 1$	500	Visibility	0.62	0.46
		Distance	98	111
	2000	Visibility	0.81	0.46
		Distance	106	111
	5000	Visibility	0.47	0.53
		Distance	91	88
$Vehicle\ 2$	500	Visibility	0.80	0.58
		Distance	83	96
	2000	Visibility	0.80	0.58
		Distance	103	96
	5000	Visibility	0.55	0.58
		Distance	117	96

(91 vs 88). The red trajectory generated by our method is very short compared to the trajectory generated by the Tchebycheff method (96 vs 117) with very good visibility cost (0.58).

C. Case Study 3: Cooperative Motion Planning

A case is presented with two vehicles A_1, A_2 and two units B_1, B_2 in Figure 4. A_1 is assigned to monitor B_1 and A_2 is assigned to monitor B_2. Additionally, an extra cooperative cost $l_c(x_1, x_2)$ is assigned that allows a reward ω to the costs of A_1 and A_2 when they are visible to each other. Otherwise it incurs a penalty ρ on the costs of the states x_1, x_2 (See lines $8-14$ of Algorithm 4). In Figure 4(a), we see that the paths of the two vehicles attract each other while keeping visibility to their respective blue units. In contrast, Figure 4(b) is the outcome of MultiObjectiveRRT* where the vehicles only keep proximity to their assigned blue units and finish the calculated paths without cooperation.

VII. CONCLUSIONS AND FUTURE WORK

In this paper, we formulated the problem of a group of units that need to monitor a group of targets in a contested environment as a multi-objective optimal motion planning problem. We presented modifications to optimal sampling-based planning algorithms to include multiple objectives and non-additive costs. Additionally, we proposed algorithms that can handle cooperative missions based on the ideas in [8]. We found that our proposed system can generate better paths than the weighted sum and Tchebycheff model [20] for certain types of motion planning problems. Several interesting directions are left for future work.

Multi-optimality problems in motion planning appear naturally in several practical domains. The modifications of Algorithm 1 should work for other motion planning problems. An immediate goal would be testing the performance of the multi-objective addition to RRT* on benchmark problems in manipulation of an articulated robot body to see its performance.

We also want to extend the possible set of multi-objective missions in contested environments. Simple extensions will include modeling moving units as unmanned aerial vehicles (UAVs) that want to maintain a connected visibility network. Following such units to cover them makes the problem more complex where a tuning among velocity, safety, and monitoring is required. We believe that the proposed system can be a useful aid to calculate a feasible solution in these complex scenarios.

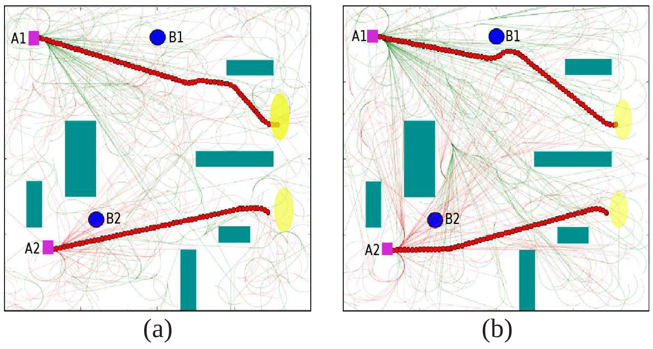

Fig. 4. (a) Cooperative path generation using MultiObjectiveRRT* Algorithm; (b) MultiObjectiveRRT* path generation without cooperation.

REFERENCES

[1] J. Barraquand and J. Latombe. On nonholonomic mobile robots and optimal maneuvering. In *Intelligent Control, 1989. Proceedings., IEEE International Symposium on*, pages 340–347. IEEE, 1989.
[2] L. E. Dubins. On curves of minimal length with a constraint on average curvature, and with prescribed initial and terminal positions and tangents. *American Journal of Mathematics*, 79(3):497–516, 1957.
[3] J. S. Dyer, P. C. Fishburn, R. E. Steuer, J. Wallenius, and S. Zionts. Multiple criteria decision making, multiattribute utility theory: the next ten years. *Management science*, 38(5):645–654, 1992.
[4] M. Ehrgott. Approximation algorithms for combinatorial multicriteria optimization problems. *International Transactions in Operational Research*, 7(1):5–31, 2000.
[5] K. Fujimura. Path planning with multiple objectives. *Robotics & Automation Magazine, IEEE*, 3(1):33–38, 1996.
[6] P. R. Giordano and M. Vendittelli. Shortest paths to obstacles for a polygonal dubins car. *Robotics, IEEE Transactions on*, 25(5):1184–1191, 2009.
[7] S. Karaman and E. Frazzoli. sampling-based motion planning library for dynamical systems. *https://svn.csail.mit.edu/smp*.
[8] S. Karaman and E. Frazzoli. Incremental sampling-based algorithms for a class of pursuit-evasion games. In *Algorithmic Foundations of Robotics IX*, pages 71–87. Springer, 2010.
[9] S. Karaman and E. Frazzoli. Incremental sampling-based algorithms for optimal motion planning. *Robotics Science and Systems*, 2010.
[10] S. Karaman and E. Frazzoli. Sampling-based algorithms for optimal motion planning. *The International Journal of Robotics Research*, 30(7):846–894, 2011.
[11] S. Karaman, M. R Walter, A. Perez, E. Frazzoli, and S. Teller. Anytime motion planning using the rrt*. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 1478–1483. IEEE, 2011.
[12] J. Kuffner and S. M. LaValle. Rrt-connect: An efficient approach to single-query path planning. In *Robotics and Automation, 2000. Proceedings. ICRA'00. IEEE International Conference on*, volume 2, pages 995–1001. IEEE, 2000.
[13] S. M. LaValle. *Planning Algorithms*. Cambridge University Press, Cambridge, U.K., 2006. Also available at http://planning.cs.uiuc.edu/.
[14] S. M. LaValle and S. A. Hutchinson. Optimal motion planning for multiple robots having independent goals. *Robotics and Automation, IEEE Transactions on*, 14(6):912–925, 1998.
[15] K. J. Obermeyer, P. Oberlin, and S. Darbha. Sampling-based path planning for a visual reconnaissance unmanned air vehicle. *Journal of Guidance, Control, and Dynamics*, 35(2):619–631, 2012.
[16] B. K. Oleiwi, R. Al-Jarrah, H. Roth, and Bahaa I. K. Multi objective optimization of trajectory planning of non-holonomic mobile robot in dynamic environment using enhanced ga by fuzzy motion control and a*. In *Neural Networks and Artificial Intelligence*, pages 34–49. Springer, 2014.
[17] D. A. Richie, J. A. Ross, S. J. Park, and D. R. Shires. A monte carlo method for multi-objective correlated geometric optimization. Technical report, DTIC Document, 2014.
[18] Z. Tarapata. Selected multicriteria shortest path problems: An analysis of complexity, models and adaptation of standard algorithms. *International Journal of Applied Mathematics and Computer Science*, 17(2):269–287, 2007.
[19] A. Warburton. Approximation of pareto optima in multiple-objective, shortest-path problems. *Operations Research*, 35(1):70–79, 1987.
[20] D. Yi, M. A. Goodrich, and K. D. Seppi. Morrf: sampling-based multi-objective motion planning. In *Proceedings of the 24th International Conference on Artificial Intelligence*, pages 1733–1739. AAAI Press, 2015.
[21] Z. Yu, R. J. Baxley, and G. T. Zhou. Multi-user miso broadcasting for indoor visible light communication. In *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*, pages 4849–4853. IEEE, 2013.
[22] Q. Zhang and H. Li. Moea/d: A multiobjective evolutionary algorithm based on decomposition. *Evolutionary Computation, IEEE Transactions on*, 11(6):712–731, 2007.