

# A New Event-Driven Cooperative Receding Horizon Controller for Multi-agent Systems in Uncertain Environments

Yasaman Khazaeni and Christos G. Cassandras  
 Division of Systems Engineering  
 and Center for Information and Systems Engineering  
 Boston University  
 Brookline, MA 02446  
 yas@bu.edu, cgc@bu.edu

**Abstract**—In previous work, a Cooperative Receding Horizon (CRH) controller was developed for solving cooperative multi-agent problems in uncertain environments. In this paper, we overcome several limitations of this controller, including potential instabilities in the agent trajectories and poor performance due to inaccurate estimation of a reward-to-go function. We propose an event-driven CRH controller to solve the maximum reward collection problem (MRCP) where multiple agents cooperate to maximize the total reward collected from a set of stationary targets in a given mission space. Rewards are non-increasing functions of time and the environment is uncertain with new targets detected by agents at random time instants. The controller sequentially solves optimization problems over a planning horizon and executes the control for a shorter action horizon, where both are defined by certain events associated with new information becoming available. In contrast to the earlier CRH controller, we reduce the originally infinite-dimensional feasible control set to a finite set at each time step. We prove some properties of this new controller and include simulation results showing its improved performance compared to the original one.

## I. INTRODUCTION

Cooperative control is used in systems where a set of control agents with limited sensing, communication and computational capabilities seeks to achieve objectives defined globally or individually [1],[2]. Uncertain environments further require the agents to respond to random events. Examples arise in UAV teams, cooperative classification, mobile agent coordination, rendez-vous problems, task assignment, persistent monitoring, coverage control and consensus problems; see [3],[4],[5],[6],[7],[8],[9],[10],[11] and references therein. Both centralized and decentralized control approaches are used; in the latter case, communication between the agents in order to make collaborative decisions plays a crucial role.

In this paper, we consider Maximum Reward Collection Problems (MRCP) where  $N$  agents are collecting time-dependent rewards associated with  $M$  targets in an uncertain environment. In a deterministic setting with equal target rewards a one-agent MRCP is an instance of a Traveling Salesman Problem (TSP), [12],[13]. The multi-agent MRCP is similar to the Vehicle Routing Problem (VRP) [14]. These are combinatorial problems for which globally optimal solu-

tions are found through integer programming. For example, in [15],[16], a deterministic MRCP with a linearly decreasing reward model is cast as a dynamic scheduling problem and solved via heuristics.

Because of the MRCP complexity, it is natural to resort to decomposition techniques. One approach is to seek a functional decomposition that divides the problem into smaller sub-problems [17],[18] which may be defined at different levels of the system dynamics. An alternative is a time decomposition where the main idea is to solve a finite horizon optimization problem, then continuously extend this *planning horizon* forward (either periodically or in event-driven fashion). This is in the spirit of receding horizon techniques used in Model Predictive Control (MPC) to solve optimal control problems for which obtaining infinite horizon feedback control solutions is extremely difficult [19]. In such methods, the current control action is calculated by solving a finite horizon open-loop optimal problem using the current state of the system as the initial state. At each instant, the optimization yields an optimal control sequence executed over a shorter *action horizon* before the process is repeated. In the context of multi-agent systems, a Cooperative Receding Horizon (CRH) controller was introduced in [20] with the controller steps defined in event-driven fashion (with events dependent on the observed system state) as opposed to being invoked periodically, in time-driven fashion. A key feature of this controller is that it does not attempt to make any explicit agent-to-target assignments, but only to determine headings that, at the end of the current planning horizon, would place agents at positions such that a total expected reward is maximized. Nonetheless, as shown in [20], a stationary trajectory for each agent is guaranteed under certain conditions, in the sense that an agent trajectory always converges to some target in finite time.

In this paper, we consider MRCPs in uncertain environments where, for instance, targets appear/disappear at random times and a target may have a random initial reward and a random reward decreasing rate. The contribution is to introduce a new CRH controller, allowing us to overcome several limitations of the controller in [20], including potential instabilities in the agent trajectories and poor performance due to inaccurate estimation of the reward-to-go function. We accomplish this by reducing, at each event-driven control

The authors' work is supported in part by NSF under Grant CNS-1139021, by AFOSR under grant FA9550-12-1-0113, by ONR under grant N00014-09-1-1051, and by ARO under Grant W911NF-11-1-0227.

evaluation step, the originally infinite-dimensional feasible control set to a finite set and by improving the estimation process for the reward to go, including a new “travel cost factor” for each target which accommodates different target configurations in a mission space. We also establish some properties of this new controller whose overall performance is significantly better relative to the original one, as illustrated through various simulation examples.

In section II, the MRCP is formulated and in Section III we place the problem in a broader context of event-driven optimal control. In Sections IV and V the original CRH controller is reviewed and the proposed new controller and some of its properties are established. In Section VI simulation examples are presented and future research directions are outlined in the conclusions.

## II. PROBLEM FORMULATION

We consider a MRCP where agents and targets are located in a mission space  $S$ . There are  $M$  targets defining a set  $\mathcal{T} = \{1, \dots, M\}$  and  $N$  agents defining a set  $\mathcal{A} = \{1, \dots, N\}$ . The mission space may have different topological characteristics. In a Euclidean topology,  $S \subset \mathbb{R}^2$  as illustrated in Fig. 1 with a triangle denoting a base, circles are agents and squares are targets. In this case, the distance metric  $d(x, y)$  is a simple Euclidean norm such that  $d : S \times S \rightarrow \mathbb{R}$  is the length of the shortest path between points  $x, y \in S$ . Moreover, the feasible agent headings are given by the set  $\mathbf{U}_j(t) = [0, 2\pi]$ ,  $j \in \mathcal{A}$ . If there are obstacles in  $S$ , the feasible headings and the shortest path between two points should be defined accordingly. Alternatively, the mission space may be modeled as a graph  $\mathcal{G}(E, V)$  with  $V$  representing the location of targets and the base. Feasible headings are defined by the (directed) edges at each node and the distance  $d(u, v)$  is the sum of the edge weights on the shortest path between  $u$  and  $v$ . In this paper, we limit ourselves to a Euclidean mission topology.

Targets are located at points  $\mathbf{y}_i \in S$ ,  $i \in \mathcal{T}$ . Target  $i$ 's reward is denoted by  $\lambda_i \phi_i(t)$  where  $\lambda_i$  is the initial maximum reward and  $\phi_i(t) \in [0, 1]$  is a non-increasing discount function. By using the appropriate discounting function we can incorporate constraints such as hard or soft deadlines for targets. An example of a discount function is

$$\phi_i(t) = \begin{cases} 1 - \frac{\alpha_i}{D_i} t & \text{if } t \leq D_i \\ (1 - \alpha_i) e^{-\beta_i(t - D_i)} & \text{if } t > D_i \end{cases} \quad (1)$$

where  $\alpha_i$ ,  $\beta_i$  and  $D_i$  are given parameters. Agents are located at  $\mathbf{x}_j(t) \in S$ . Each agent has a controllable heading at time  $t$ ,  $u_j(t) \in \mathbf{U}_j(t) = [0, 2\pi]$ . The velocity of agent  $j$  is

$$\mathbf{v}_j(t) = V_j \begin{bmatrix} \cos(u_j(t)) \\ \sin(u_j(t)) \end{bmatrix} \quad (2)$$

where we assume that  $V_j$  is a fixed speed.

We define a *mission* as the process of the  $N$  agents cooperatively collecting the maximum possible total reward from  $M$  targets within a given mission time  $T$ . Upon collecting rewards from all targets, the agents deliver it to a *base* located at  $\mathbf{z} \in S$ . Events occurring during a mission can be controllable (e.g., collecting a target's reward)

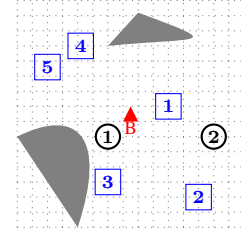


Fig. 1. Sample mission space with filled blue regions as obstacles

or random (e.g., the appearance/disappearance of targets or changes in their location). The event-driven CRH controller we will develop, handles these random events by re-solving the optimal control problem as in the original CRH controller in [20]. In order to ensure that agents collect target rewards in finite time, we assume that each target has a radius  $s_i > 0$  and that agent  $j$  collects reward  $i$  at time  $t$  if and only if  $d(\mathbf{x}_j(t), \mathbf{y}_i) \leq s_i$ .

## III. AN EVENT-DRIVEN OPTIMIZATION VIEW

We view the solution of a MRCP as a sequence of headings for all agents and associated heading switching times. We define a policy  $\pi$  as a vector  $[\mathbf{u}, \xi]$  where  $\xi = [\xi_1, \dots, \xi_K]$  are the switching time intervals over which headings are maintained with  $t_{k+1} = \sum_{l=1}^k \xi_l$ , and  $t_1 = 0$ . The control  $\mathbf{u} = [\mathbf{u}_1, \dots, \mathbf{u}_K]$  with  $\mathbf{u}_k = [u_1(t_k), \dots, u_N(t_k)]$  is the vector of all the agent headings at time  $t_k$ . With  $M$  bounded, there exist policies  $\pi$  such that all targets are visited over a finite number of switching events. Each switching time  $t_k$  is either the result of a controllable event (e.g., visiting a target) or an uncontrollable random event. This is a complex stochastic control problem where the state space  $\Xi$  is the set of all possible location of agents  $\mathcal{X}_k = [\mathbf{x}_1(t_k), \dots, \mathbf{x}_N(t_k)]$  and targets  $\mathcal{Y}_k = [\mathbf{y}_1, \dots, \mathbf{y}_{M_k}]$  with  $M_k = \|\mathcal{T}_k\|$  and  $\mathcal{T}_k$  is the set of unvisited targets at time  $t_k$ . As the mission evolves,  $M_k$  decreases and the mission is complete when either  $M_k = 0$  or a given mission time  $T$  is reached. The complete system state at time  $t_k$  is  $(\mathcal{X}_k, \mathcal{Y}_k) \in \Xi$ . We define the optimization problem  $\mathbf{P}$  as:

$$\max_{\pi} \sum_{k=1}^K R_{\pi}(t_k, \mathcal{X}_k, \mathcal{Y}_k) \quad (3)$$

where

$$R_{\pi}(t_k, \mathcal{X}_k, \mathcal{Y}_k) = \sum_{i=1}^{M_k} \sum_{j=1}^N \lambda_i \phi_i(t_k) \mathbb{1}\{d(\mathbf{x}_j(t_k), \mathbf{y}_i) \leq s_i\}$$

The time a target is visited is a controllable event associated with a heading switching. In a deterministic problem, there is no need to switch headings unless a target is visited, but in an uncertain setting the switching times are not limited to these events. We define a subsequence  $\tau^{\pi} = \{\tau_1^{\pi}, \tau_2^{\pi}, \dots, \tau_M^{\pi}\}$  of  $\{t_1, \dots, t_K\}$ ,  $M \leq K$ , so that  $\tau_i^{\pi}$  is the time target  $i$  is visited. Note that  $\tau^{\pi}$  is not a monotonic sequence, since targets can be visited in any order. Therefore, (3) can be rewritten as

$$\max_{\pi} \sum_{i=1}^M \lambda_i \phi_i(\tau_i^{\pi}) \quad (4)$$

Defining the immediate reward as being collected during a time period  $\xi_k$  and the reward-to-go as being aggregated over all  $t > t_k + \xi_k$ , an optimality equation for this problem is:

$$J(t_k, \mathcal{X}_k, \mathcal{Y}_k) = \max_{\mathbf{u}_k, \xi_k} [J_I(t_k, \mathcal{X}_k, \mathcal{Y}_k, \mathbf{u}_k, \xi_k) + J(t_{k+1}, \mathcal{X}_{k+1}, \mathcal{Y}_{k+1})] \quad (5)$$

where  $J(t_k, \mathcal{X}_k, \mathcal{Y}_k)$  denotes the maximum total reward at time  $t_k$  with current state  $(\mathcal{X}_k, \mathcal{Y}_k)$  and  $J_I(t_k, \mathcal{X}_k, \mathcal{Y}_k, \mathbf{u}_k, \xi_k)$  is the immediate reward collected in the interval  $(t_k, t_{k+1}]$ . Finally,  $J(t_{k+1}, \mathcal{X}_{k+1}, \mathcal{Y}_{k+1})$  is the maximum reward-to-go at  $t_{k+1}$  assuming no future uncertainty, i.e., we avoid the use of an *a priori* stochastic model for the environment, opting instead to react to random events by re-solving (5) when this happens. Letting  $\tau^* = \max_{i \in \mathcal{T}} \{\tau_i^*\}$ , we set  $J(\tau^*, \mathcal{X}_K, \mathcal{Y}_K) = 0$ . Henceforth, we write  $J(t_k, \mathcal{X}_k, \mathcal{Y}_k) = J(t_k)$  for brevity. Had we assumed a fixed value for  $\xi_k$  *a priori*, the optimization problem (5) could have been solved using Dynamic Programming (DP) with the terminal state reached when no target is left in the mission space. However, a fixed  $\xi_k$  does not allow for real-time reactions to new events. This fact, along with the size of the state space renders DP impractical and motivates a receding horizon control approach where we set  $\xi_k = H_k$  based on a *planning horizon*  $H_k$  selected at time step  $t_k$ . Then, a finite horizon optimal control problem over  $(t_k, t_k + H_k]$  is solved to determine the optimal control  $\mathbf{u}_k^*$ . This control is maintained for an *action horizon*  $h_k \leq H_k$ . A new optimization problem is re-solved at  $t_{k+1} = t_k + h_k$  or earlier if any random event is observed. Following (5), the optimization problem  $\mathbf{P}_k$  is

$$\max_{\mathbf{u}_k} [J_I(\mathbf{u}_k, t_k, H_k) + J(t_{k+1}, H_{k+1})] \quad (6)$$

where  $J(t_{k+1}, H_{k+1})$  and  $J_I(\mathbf{u}_k, t_k, H_k)$  were defined above assuming  $\xi_k = H_k$ . The immediate reward is zero if agents do not visit any target during  $(t_k, t_k + H_k]$ , otherwise it is the reward collected over this interval. Fixing the value of  $H_k$  is not constraining, since it is always possible to stop and re-solve a new problem at any  $t > t_k$ .

#### IV. CRH CONTROL SCHEME

In this section we briefly review the CRH controller introduced in [20] and identify several limitations of it to motivate the methods we will use to overcome them.

*Cooperation Scheme:* In [20] the agents divide the mission space into a *dynamic* partition at each mission step. The degree of an agent's responsibility for each target depends on the relative proximity of the agent to the target. A neighbor set is defined for each target which includes its  $b$  closest agents,  $b = 1, 2, \dots$ , sharing the responsibility for that target until another agent moves closer. A value of  $b = 2$  is used in the previous and current work for simplicity. Defining  $c_{ij}(t) = d(\mathbf{y}_i, \mathbf{x}_j(t))$  to be the direct distance between target  $i$  and agent  $j$  at time  $t$ , let  $B^l(\mathbf{y}_i, t)$  be the  $l$ th closest agent to target  $i$  at time  $t$ . Formally,

$$B^l(i, t) = \underset{j \in \mathcal{A}, j \neq B^1(i, t), \dots, j \neq B^{l-1}(i, t)}{\operatorname{argmin}} \{c_{ij}(t)\} \quad (7)$$

Let  $\beta^b(i, t) = \{B^1(i, t), \dots, B^b(i, t)\}$  be a neighbor set based on which a *relative distance* function is defined for all  $j \in \mathcal{A}$ :

$$\delta_{ij}(t) = \begin{cases} \frac{c_{ij}(t)}{\sum_{k \in \beta^b(i, t)} c_{ik}(t)} & \text{if } j \in \beta^b(i, t); \\ 1 & \text{otherwise} \end{cases} \quad (8)$$

Obviously, if  $j \notin \beta^b(i, t)$ , then  $\delta_{ij}(t) = 1$ . The *relative proximity function*  $p(\delta_{ij}(t))$  defined in [20] is viewed as the probability that target  $i$  will be visited by agent  $j$ :

$$p(\delta_{ij}(t)) = \begin{cases} 1, & \text{if } \delta \leq \Delta \\ \frac{1-\Delta-\delta}{1-2\Delta}, & \text{if } \Delta \leq \delta \leq 1-\Delta \\ 0, & \text{if } \delta > 1-\Delta \end{cases} \quad (9)$$

Here,  $\Delta \in [0, \frac{1}{2})$  defines the level of cooperation between the agents. By increasing  $\Delta$  an agent will take full responsibility for more targets, hence less cooperation. Each agent takes on full responsibility for target  $i$  if  $\delta_{ij}(t) \leq \Delta$ . As shown in [20], when  $\Delta = \frac{1}{2}$  the regions converge to the Voronoi tessellation of the mission space, with the location of agents at the centers of the Voronoi tiles. There is no cooperation region in this case and each agent is fully responsible for the targets within its own Voronoi tile. On the other hand, when  $\Delta = 0$  no matter how close an agent is to a target, the two agents are still responsible for that target.

*Planning and Action Horizons:* In [20],  $H_k$  is defined as the earliest time of an event such that one of the agents can visit one of the targets:

$$H_k = \min_{l \in \mathcal{T}_k} \left\{ \frac{d(\mathbf{x}_j(t_k), \mathbf{y}_l)}{V_j} \right\} \quad (10)$$

This definition of *planning horizon* for the CRH controller ensures no controllable event can take place during this horizon. It also ensures that re-evaluation of the CRH control is event-driven, as opposed to being specified by a clock which involves a tedious synchronization over agents. Fig. 2 illustrates how  $H_k$  is determined when  $V_j = 1$ . The CRH control calculated at  $t_k$  is maintained for an *action horizon*  $h_k \leq H_k$ . In [20]  $h_k$  is defined either (i) through a random event that may be observed at  $t_e \in (t_k, t_k + H_k]$  so that  $h_k = t_e - t_k$ , or (ii) as  $h_k = \gamma H_k$ ,  $\gamma \in (0, 1)$ . It is also shown in [20] that under (10) the CRH controller generates a stationary trajectory for each agent under certain conditions, in the sense that an agent trajectory always converges to some target in finite time.

##### A. Original CRH Controller Limitations

*Instabilities in agent trajectories:* The optimization problem considered in [20] uses a potential function which is minimized in order to maximize the total reward. The stationary trajectory guarantee mentioned above is based on the assumption that all minima of this function are at the target locations. If this assumption fails to hold, the agents are directed toward the weighted center of gravity of all targets. This can happen in missions where targets attain a symmetric configuration, leading to oscillatory behavior in the agent trajectories. An example is shown in Fig. 5(a) with the original CRH controller applied to a single agent, resulting in oscillations between three targets with equal

rewards. This problem was addressed in [21] by introducing a monotonically increasing cost factor (or penalty)  $C(u_j)$  on the heading  $u_j$ . While this prevents some of the instabilities, it has to be appropriately tuned for each mission. We show how to overcome this problem in Section V.

**Hedging and mission time:** The agent trajectories in [20] are specifically designed to direct them to positions close to targets but not exactly towards them unless they are within a certain “capture distance,” the motivation being that no agent should be committed to a target until the latest possible time so as to hedge against the uncertainty of new, potentially more attractive, randomly appearing targets. This hedging effect is helpful in handling such uncertainties, but it can create excessive loss of time, especially when rewards are declining fast. This can be addressed by more direct movements towards targets, while also re-evaluating the control frequently enough. The feasible control set in the original CRH is the continuous set  $[0, 2\pi]^N$ , and by appropriately reducing this to a discrete set of control values we will show how we can eliminate unnecessary hedging. This also reduces the complexity of the optimal control problem at each time step and facilitates the problem solution over a finite number of evaluations.

**Estimation of reward-to-go:** In the original CRH control scheme, the visit times are estimated as the earliest time any agent  $j$  would reach some target  $i$ , given a control  $\mathbf{u}_k$  at time  $t_k$  and maintained over  $(t_k, t_k + H_k]$ . Thus, the estimated visit time  $\tilde{\tau}_{lj}(\mathbf{u}_k, t_k, H_k)$  for any  $l \in \mathcal{T}_k$  is

$$\tilde{\tau}_{lj}(\mathbf{u}_k, t_k, H_k) = t_k + H_k + d(\mathbf{x}_j(t_k + H_k, u_j(t_k)), \mathbf{y}_l)$$

where  $\mathbf{x}_j(t_k + H_k, u_j(t_k))$  is the location of the agent  $j$  in the next time step given the control  $u_j(t_k)$ . This is a lower bound for  $\tilde{\tau}_{ij}$  feasible only when  $M_k \leq N$ , leading also to a mostly unattainable upper bound for the total reward. We will show how this estimate is improved by a more accurate projection of each agent’s future trajectory.

## V. THE NEW CRH CONTROLLER

In this section we present a new version of the CRH controller in [20]. Using the definition of  $\mathbf{x}_j(t_k + H_k, u_j(t_k))$  given above and assuming  $V_j = 1$  for all agents, the feasible set for  $\mathbf{x}_j(t_k + H_k, u_j(t_k))$  is defined as

$$\mathcal{F}_j(t_k, H_k) = \{\mathbf{w} \in S \mid d(\mathbf{w}, \mathbf{x}_j(t_k)) = H_k\} \quad (11)$$

In a Euclidean mission space with no obstacles,  $\mathcal{F}_j(t_k, H_k)$  is the circle centered at  $\mathbf{x}_j(t_k)$  with radius  $H_k$ . Let  $q_i(\mathbf{x}_j(t)) = \mathbb{1}\{d(\mathbf{x}_j(t), \mathbf{y}_i) \leq s_i\}$  be the indicator function capturing whether agent  $j$  visits target  $i$  at time  $t$ . We define the immediate reward at  $t_k$ :

$$J_1(\mathbf{u}_k, t_k, H_k) = \sum_{j=1}^N \sum_{l=1}^{M_k} \lambda_l \phi_l(t_k + H_k) q_l(\mathbf{x}_j(t_k + H_k, u_j(t_k))) \quad (12)$$

Following the definition of  $\tau_i$  as the visit time of target  $i$  in (4), we define  $\tilde{\tau}_{ij}$  as the estimated visit time of target  $i$  by agent  $j$ . Here  $\tilde{\tau}_{ij} > t_k$  and any of the agents in the mission space has a chance to visit target  $i$ . At time  $t_k$  we define an estimate of the reward-to-go  $J(t_{k+1}, H_{k+1})$  for each  $\mathbf{u}_k$  as

$$\tilde{J}(\mathbf{u}_k, t_{k+1}, H_{k+1}) = \quad (13)$$

$$\sum_{j=1}^N \sum_{l=1}^{M_{k+1}} \lambda_l \phi_l(\tilde{\tau}_{lj}(\mathbf{u}_k, t_k, H_k)) \cdot q_l(\mathbf{x}_j(\tilde{\tau}_{lj}(\mathbf{u}_k, t_k, H_k)))$$

We previously mentioned that the original CRH control approach used a lower bound for estimating  $\tilde{\tau}_{ij}$ . We improve this estimate and at the same time address the other two limitations presented above through three modifications: (i) We introduce a new *travel cost* for each target, which combines the distance of a target from agents, its reward, and a local sparsity factor. (ii) We introduce an *active target set* associated with each agent at every control evaluation instant  $t_k$ . This allows us to reduce the infinite dimensional feasible control set at  $t_k$  to a finite set. (iii) We introduce a new event-driven action horizon  $h_k$  which makes use of the active target set definition. With these three modifications, we finally present a new CRH control scheme based on a process of looking ahead over a number of CRH control steps and aggregating the remainder of a mission through a reward-to-go estimation process.

**Travel Cost Factor:** At each control iteration instant  $t_k$ , we define  $\zeta_i(t_k)$  for target  $i$  to measure the sparsity of rewards in its vicinity. Let  $\bar{D}_i > 0$  be such that  $\phi_i(\bar{D}_i) = 0$  for each  $i \in \mathcal{T}$  and set  $D_i = \min(\bar{D}_i, T)$  so that the average reward decreasing rate of  $i$  over the mission is given by  $\lambda_i/D_i$ . Let the set  $\{1, 2, \dots, I\}$  contain the indices of the  $I$  closest targets to  $i$  at time  $t_k$ . We then define the *sparsity factor* for target  $i$  as

$$\zeta_i(t_k) = \sum_{l=1}^I \gamma^l \frac{d(\mathbf{y}_i, \mathbf{y}_l)}{\lambda_l/D_l} \quad (14)$$

where  $\gamma \in [0, 1]$  is a parameter used to shift the weight among the  $I$  targets. Note that  $\zeta_i(t_k)$  is time-dependent since the set of  $I$  closest targets changes over time as rewards are collected. A larger  $\zeta_i(t_k)$  implies that target  $i$  is located in a relatively sparse area and vice versa. The parameter  $I$  is chosen based on the number of targets in the mission space and the computation capacity of the controller. The main idea for  $\zeta_i(t_k)$  comes from [22] where it was used to solve TSP problems with clustering. Next, for any point in  $\mathbf{x} \in S$ , we define target  $i$ ’s travel cost at time  $t_k$  as

$$\eta_i(\mathbf{x}, t_k) = \frac{d(\mathbf{x}, \mathbf{y}_i)}{\lambda_i/D_i} + \zeta_i(t_k) \quad (15)$$

The travel cost is proportional to the distance metric, so the farther a target is from  $\mathbf{x}$  the more costly is the visit to that target. It is inversely proportional to the reward’s average decreasing rate, implying that the faster the reward decreases, the less the travel cost is. Adding  $\zeta_i(t_k)$  gives a target in a sparse area a higher travel cost as opposed to one where there is an opportunity for a visiting agent to collect additional rewards from its vicinity.

**Active Targets:** At each control iteration instant  $t_k$ , we define for each agent  $j$  a subset of targets with the following property relative to the planning horizon  $H_k$ :

$$S_j(t_k, H_k) = \{\ell \mid \exists \mathbf{x} \in \mathcal{F}_j(t_k, H_k) \text{ s.t. } \ell = \underset{i \in \mathcal{T}_k}{\operatorname{argmin}} \eta_i(\mathbf{x}, t_k + H_k), i = 1, 2, \dots, M_k\} \quad (16)$$

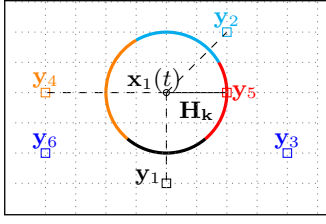


Fig. 2. The Active Target Set for agent 1:  $S_1(\mathbf{x}_1(t_k), H_k) = \{1, 2, 4, 5\}$ . This is termed the *active target set* and (16) implies that  $i \in \mathcal{T}_k$  is an active target for agent  $j$  if and only if it has the smallest travel cost from at least one point on the reachable set  $\mathcal{F}_j(t_k, H_k)$ . This means that every  $\mathbf{x} \in \mathcal{F}_j(t_k, H_k)$  is associated with one of the active targets and, therefore, so does every feasible heading  $u_j(t_k)$ , which corresponds to active target  $l$  if and only if:

$$l = \underset{i \in \mathcal{T}_k}{\operatorname{argmin}} \eta_i(\mathbf{x}(t_k + H_k, u_j(t_k)), t_k + H_k) \quad (17)$$

When  $d(x, y)$  is continuous, active targets partition the reachable set  $\mathcal{F}_j(t_k, H_k)$  into several arcs as illustrated in Fig. 2 where, for simplicity, we assume  $\gamma = 0$  in (14) and all  $\lambda_i$  and  $\phi_i(t)$ ,  $i = 1, \dots, M$  are the same. In this case, agent 1 has four active targets:  $S_1(t_k, H_k) = \{1, 2, 4, 5\}$ . The common feature of all points on an arc is that they correspond to the same active target with the least travel cost.

**Construction of  $S_j(t_k, H_k)$ :** For each target  $l \in \mathcal{T}_k$  and each agent  $j$ , let  $\mathcal{L}_k(\mathbf{x}_j(t_k), l)$  be the set of points  $\mathbf{x} \in S$  defining the shortest path from  $\mathbf{x}_j(t_k)$  to  $\mathbf{y}_l$ . The intersection of this set with  $\mathcal{F}_j(t_k, H_k)$  is the set of closest points to target  $l$  in the feasible set:

$$\mathcal{C}_{l,j}(t_k, H_k) = \mathcal{L}_k(\mathbf{x}_j(t_k), l) \cap \mathcal{F}_j(t_k, H_k) \quad (18)$$

In a Euclidean mission space,  $\mathcal{L}_k(\mathbf{x}_j(t_k), l)$  is a convex combination (line segment) of  $\mathbf{x}_j(t_k)$  and  $\mathbf{y}_l$ , while  $\mathcal{C}_{l,j}(t_k, H_k)$  is a single point where this line crosses the circle  $\mathcal{F}_j(t_k, H_k)$ . The following lemma provides a necessary and sufficient condition for identifying targets which are active for an agent at  $t_k$  using  $\mathcal{C}_{l,j}(t_k, H_k)$ .

**Lemma 1.** *Target  $l$  is an active target for agent  $j$  at time  $t_k$  if and only if,  $\forall i \in \mathcal{T}_k$*

$$\eta_l(\mathcal{C}_{l,j}(t_k, H_k), t_{k+1}) \leq \eta_i(\mathcal{C}_{l,j}(t_k, H_k), t_{k+1}) \quad (19)$$

**Proof:** See Appendix. ■

**Action Horizon:** The definition of  $h_k$  in [20] requires frequent iterations of the optimization problem through which  $\mathbf{u}_k^*$  is determined in case no random event is observed to justify such action. Instead, when there are no random events, we define a new *multiple immediate target event* to occur when the minimization in (10) returns more than one target meaning the agent is at an equal distance to at least two targets. We then define  $h_k$  to be the shortest time until the first multiple immediate target event occurs in  $(t_k, t_k + H_k]$ :

$$h_k = \min \left\{ H_k, \inf \{ t > t_k : \exists l, l^* \in \mathcal{T}_k \text{ s.t.} \right. \quad (20)$$

$$\left. d(\mathbf{x}_j(t_k + t, u_j(t_k)), \mathbf{y}_l) = d(\mathbf{x}_j(t_k + t, u_j(t_k)), \mathbf{y}_{l^*}) \} \right\}$$

Consequently, this definition of  $h_k$  eliminates any unnecessary control evaluation.

## A. Look Ahead and Aggregate Process

In order to solve the optimization problem  $\mathbf{P}_k$  in (6) using the CRH approach, we need the estimated visit time  $\tilde{\tau}_{ij}(\mathbf{u}_k, t_k, H_k)$  for each  $\mathbf{u}_k$  through which  $\tilde{J}(\mathbf{u}_k, t_{k+1}, H_{k+1})$  in (13) can be evaluated. This estimate is obtained by using a projected path for each agent. This path projection consists of a *look ahead* and an *aggregate* step. In the first step, the active target set  $S_j(t_k, H_k)$  is determined for agent  $j$ . With multiple agents in a mission, at each iteration step the remaining targets are partitioned using the relative proximity function in (9). We denote the target subset for agent  $j$  as  $\mathcal{T}_{k,j}$  where:

$$l \in \mathcal{T}_{k,j} \iff p(\delta_{lj}(t_k)) > p(\delta_{lq}(t_k)) \quad \forall q \in \mathcal{A} \quad (21)$$

Let  $|\mathcal{T}_{k,j}| = M_{k,j}$ . All  $\tilde{\tau}_{ij}(\mathbf{u}_k, t_k, H_k)$  are estimated as if  $j$  would visit targets in its own subset by visiting the one with the least travel cost first. We define the agent  $j$ 's tour as the permutation  $\theta^j(\mathbf{u}_k, t_k, H_k)$  specifying the order in which it visits targets in  $\mathcal{T}_{k,j}$ . For simplicity, we write  $\theta^j$  and let  $\theta_i^j$  denote the  $i^{\text{th}}$  target in agent  $j$ 's tour. Then, for all  $l \in \mathcal{T}_{k,j}$  and  $t_{k+1} = t_k + H_k$ :

$$\eta_{\theta_1^j}(\mathbf{x}_j(t_{k+1}, u_j(t_k)), t_{k+1}) \leq \eta_l(\mathbf{x}_j(t_{k+1}, u_j(t_k)), t_{k+1})$$

and with  $n = 2, \dots, M_{k,j} - 1$ , for all  $l \in \mathcal{T}_{k,j} - \{\theta_1^j, \dots, \theta_n^j\}$ ,

$$\eta_{\theta_{n+1}^j}(\mathbf{y}_{\theta_n^j}, \tilde{\tau}_{\theta_n^j}(\mathbf{u}_k, t_k, H_k)) \leq \eta_l(\mathbf{y}_{\theta_n^j}, \tilde{\tau}_{\theta_n^j}(\mathbf{u}_k, t_k, H_k))$$

where

$$\tilde{\tau}_{\theta_n^j}(\mathbf{u}_k, t_k, H_k) = t_k + H_k + \sum_{i=1}^{n-1} d(\mathbf{y}_{\theta_i^j}, \mathbf{y}_{\theta_{i+1}^j}) \quad (22)$$

This results in the corresponding  $\tilde{\tau}_{lj}(\mathbf{u}_k, t_k, H_k)$  for all  $l \in \mathcal{T}_{k,j}$ . We can now obtain the reward-to-go estimate as

$$J_{\mathbf{A}}(\mathbf{u}_k, t_k, H_k) = \quad (23)$$

$$\sum_{j=1}^N \sum_{l=1}^{M_{k+1,j}} \lambda_l \phi_l(\tilde{\tau}_{lj}(\mathbf{u}_k, t_k, H_k)) \cdot q_l(\mathbf{x}_j(\tilde{\tau}_{lj}(\mathbf{u}_k, t_k, H_k)))$$

Recalling the immediate reward in (12), the optimization problem  $\mathbf{P}_k$  becomes:

$$\max_{\mathbf{u}_k \in [0, 2\pi]^N} [J_{\mathbf{I}}(\mathbf{u}_k, t_k, H_k) + J_{\mathbf{A}}(\mathbf{u}_k, t_k, H_k)] \quad (24)$$

In (11) we defined the feasible set for the location of agent  $j$  in the next step  $t_{k+1} = t_k + H_k$ . In a Euclidean mission space, each point  $\mathbf{x} \in \mathcal{F}_j(t_k, H_k)$  corresponds to a heading  $v(\mathbf{x})$  relative to the agent's location  $\mathbf{x}_j(t_k)$ . Using the definition in 18 let:

$$\mathcal{V}_j(t_k, H_k) = \{v(\mathbf{x}) | \mathbf{x} \in \mathcal{C}_{l,j}(t_k, H_k), l \in S_j(t_k, H_k)\}$$

and

$$\mathbf{V}_k = \mathcal{V}_1(t_k, H_k) \times \mathcal{V}_2(t_k, H_k) \times \dots \times \mathcal{V}_N(t_k, H_k)$$

In the next lemma, we prove that in a single-agent mission with the objective function defined in (24) the optimal control is  $u_1(t_k) = v(\mathcal{C}_{l,1}(t_k, H_k))$  for some  $l \in S_1(t_k, H_k)$ .

**Lemma 2.** *In a single agent ( $N = 1$ ) mission, if  $u_1^*$  is an optimal solution to the problem:*

$$\max_{u_k \in [0, 2\pi]} [J_{\mathbf{I}}(u_k, t_k, H_k) + J_{\mathbf{A}}(u_k, t_k, H_k)] \quad (25)$$

then  $u_1^* \in \mathcal{V}_j(t_k, H_k)$



*Proof:* See Appendix. ■

The implication of this lemma is that we can reduce the number of feasible controls to a finite set as opposed to the infinite set  $[0, 2\pi]$ .

**Theorem 1:** In a multi-agent MRCP mission, if  $\mathbf{u}^* = [u_1^*, \dots, u_N^*]$  is the optimal solution to the problem in (24) then  $\mathbf{u}^* \in \mathbf{V}_k$ .

*Proof:* See Appendix ■

Theorem 1 reduces the problem  $\mathbf{P}_k$  to a maximization problem over a finite set of feasible controls:

$$\max_{\mathbf{u}_k \in \mathbf{V}_k} [J_I(\mathbf{u}_k, t_k, H_k) + J_A(\mathbf{u}_k, t_k, H_k)]$$

This reduces the size of the problem compared to the original CRH controller. The following algorithm generates controls in this manner at each step  $t_k$  and is referred to as the “One-step Lookahead” CRH controller (extended to a “ $K$ -step Lookahead” algorithm in what follows).

#### CRH One-step Lookahead Algorithm:

- 1) Determine  $H_k$  through (10).
- 2) Determine the active target set  $S_j(t_k, H_k)$  through (16) for all  $j \in \mathcal{A}$ .
- 3) Evaluate  $J_A(\mathbf{u}_k, t_k, H_k)$  for all  $\mathbf{u}_k \in \mathbf{V}_k$  through (22) and (23)
- 4) Solve  $\mathbf{P}_k$  in (24) and determine  $\mathbf{u}_k^*$ .
- 5) Evaluate  $h_k$  through (20)
- 6) Execute  $\mathbf{u}_k^*$  over  $(t_k, t_k + h_k]$  and repeat Step 1 with  $t_{k+1} = t_k + h_k$ .

#### B. $K$ -Step Lookahead

The One-step Lookahead CRH controller can be extended to a  $K$ -step Lookahead controller with  $K > 1$  by exploring additional possible future paths for each agent at each time step  $t_k$ . In the One-step Lookahead algorithm, the optimal reward-to-go is estimated based on a single tour over the remaining targets. The  $K$ -step Lookahead algorithm estimates this reward by considering more possible tours for each agent as follows. For any feasible  $u_j(t_k) \in \mathcal{V}_j(t_k, H_k)$  the agent is hypothetically placed at the corresponding next step location  $\mathbf{x}_j(t_{k+1})$ . This is done for all agents to maintain synchronicity of the solution. At  $\mathbf{x}_j(t_{k+1})$ , a new active target set is determined, implying that agent  $j$  can have  $|S_j(t_k + H_k, H_{k+1})|$  possible paths. At this point, we can repeat the same procedure by hypothetically moving the agent to a new feasible location from the set  $\mathcal{F}_j(t_{k+1}, H_{k+1})$  or we can stop and estimate the reward-to-go for each available path. Thus, for a Two-Step Lookahead, problem  $\mathbf{P}_k$  becomes:

$$\max_{\mathbf{u}_k \in \mathbf{V}_k} \left[ J_I(\mathbf{u}_k, t_k, H_k) + \max_{\mathbf{u}_{k+1} \in \mathbf{V}_{k+1}} [J_I(\mathbf{u}_{k+1}, t_{k+1}, H_{k+1}) + J_A(\mathbf{u}_{k+1}, t_{k+1}, H_{k+1})] \right] \quad (26)$$

We extend the previous algorithm to a 2-step lookahead in the following. For a  $K$ -step we should repeat steps 1 and 2 for  $K$  times before moving to step 4 of the algorithm.

#### CRH 2-step Lookahead Algorithm:

- 1) Determine  $H_k$  through (10).
- 2) Determine the active target set  $S_j(t_k, H_k)$  through (16) for all  $j \in \mathcal{A}$ .

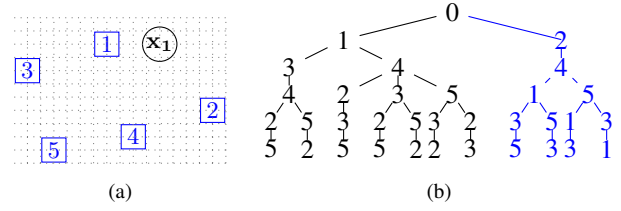


Fig. 3. (a): Five-target mission, (b): The tree structure

- 3) Repeat steps 1&2 for  $t_{k+1} = t_k + H_k$  and  $\mathbf{x}_j(t_{k+1}) = \mathbf{x}_j(t_{k+1}, u_j(t_k))$  for all  $u_j(t_k) \in \mathcal{V}_j(t_k, H_k)$ .
- 4) Evaluate  $J_A(\mathbf{u}_{k+1}, t_{k+1}, H_{k+1})$  for all  $\mathbf{u}_{k+1} \in \mathbf{V}_{k+1}$  through (22) and (23)
- 5) Solve  $\mathbf{P}_k$  in (26) and determine  $\mathbf{u}_k^*$ .
- 6) Evaluate  $h_k$  through (20)
- 7) Execute  $\mathbf{u}_k^*$  over  $(t_k, t_k + h_k]$  and repeat Step 1 with  $t_{k+1} = t_k + h_k$ .

This procedure can easily be repeated and the whole process can be represented as a tree structure where the root is the initial location of the agent and a path from the root to each leaf is a possible target sequence for the agent. In Fig. 3(a) a sample mission with 5 targets is shown with its corresponding tree in Fig. 3(b). A brute-force method involves  $5! = 120$  possible paths, whereas the tree structure for this mission is limited to 11 paths. The active target set for agent 1 consists of targets 1, 2. Each of these active targets would then generate several branches in the tree, as shown. We calculate the total reward for each path to find the optimal one. Determining the complete tree for large  $K$  is time consuming. The  $K$ -step Lookahead CRH controller enables us to investigate the tree down to a few levels and then calculate an estimated reward-to-go for the rest of the selected path. However, there is no guarantee on the monotonicity of the results with more lookahead steps and in some cases the final result degrades with one more lookahead step.

#### C. Two-Target, One-Agent Case

The simplest case of the MRCP is the case with one agent and two targets. Obviously, this is an easy routing problem whose solution is one of the two possible paths the agent can take. We prove that the One-step Lookahead algorithm solves the problem with any linearly decreasing reward function. Consider a mission with one agent and two targets with initial rewards and deadlines  $\lambda_1, D_1$  and  $\lambda_2, D_2$  respectively. The analytical solution for this case reveals whether path  $\theta_1 = (1, 2)$  or  $\theta_2 = (2, 1)$  is optimal. Following the previous analysis, we assume that  $V_1 = 1$  and set  $\mathbf{x}_1(t_k) = \mathbf{x}$  for the sake of brevity. We also assume the rewards are linearly decreasing to zero:  $\phi_i(t) = 1 - \frac{t}{D_i}$ . The two possible rewards are given by:

$$R_{(1,2)} = \lambda_1 \left[ 1 - \frac{d(\mathbf{x}, \mathbf{y}_1)}{D_1} \right] + \lambda_2 \left[ 1 - \frac{d(\mathbf{x}, \mathbf{y}_1) + d(\mathbf{y}_1 - \mathbf{y}_2)}{D_2} \right] \quad (27)$$

$$R_{(2,1)} = \lambda_2 \left[ 1 - \frac{d(\mathbf{x}, \mathbf{y}_2)}{D_2} \right] + \lambda_1 \left[ 1 - \frac{d(\mathbf{x}, \mathbf{y}_2) + d(\mathbf{y}_2 - \mathbf{y}_1)}{D_1} \right] \quad (28)$$

Therefore, if  $R_{(1,2)} > R_{(2,1)}$ , it follows that the following inequality must hold:

$$\frac{\lambda_1}{D_1} [d(\mathbf{x}, \mathbf{y}_1) - d(\mathbf{x}, \mathbf{y}_2) + d(\mathbf{y}_2, \mathbf{y}_1)] < \frac{\lambda_2}{D_2} [d(\mathbf{x}, \mathbf{y}_2) - d(\mathbf{x}, \mathbf{y}_1) + d(\mathbf{y}_1, \mathbf{y}_2)] \quad (29)$$

and the optimal path is  $\theta^* = \theta_1$ . Letting  $\theta^{CRH}$  denote the path obtained by the One-step Lookahead CRH controller, we show next that this controller recovers the optimal path  $\theta^*$ .

**Theorem 2:** Consider a two-target, one-agent mission. If  $\gamma = 0$  in (14) and target  $i$ 's reward at time  $t$  is  $\lambda_i(1 - \frac{t}{D_i})$ , then  $\theta^{CRH} = \theta^*$ .

*Proof:* See Appendix. ■

#### D. Monotonicity in the Look Ahead Steps

Questions that come into mind after introducing the multiple look ahead steps CRH controller are: How many look ahead steps should we perform? Is it always better to do more look ahead steps? Or in a simple way, does the more steps look ahead always gives a better answer than less?

The answer to the first question is that it depends on the size of the problem and our computation capability. We can even adjust the number of look ahead steps during the course of the solution. We can start with more when there is more targets available and lower the number once there is only a few targets in the mission space. The answer to the other two questions is No. As much as one would like to have a sort of monotonicity effect in this problem, the complexity of the problem and its significant dependence on the mission topology causes the non-monotone results with different number of look ahead steps. Here we are going to show a case with 10 equally important targets and one agent. This is a straight forward TSP for which the optimal path can be obtained through an exhaustive search. For this case the one and two look ahead steps CRH controllers find the same path with a reward of 92.6683. However, once we move up to three look ahead steps, the CRH controller degrade to a worst path with 92.5253 reward. The path for these controllers is shown in figures 4(a) and 4(b). The optimal path that is calculated through the exhaustive search is obtained by the CRH controllers when we go up to six look ahead steps (Fig. 4(d)). The observation is that the non-monotone results from higher number of look ahead is a local effect and once we increase the look ahead steps CRH controller can solve the problem to the optimality. This obviously is not the case for all missions and in some cases the optimal path can not be retrieved by CRH controller with any look ahead steps.

## VI. SIMULATION EXAMPLES

We provide several MRCP examples in which the performance of the original and new CRH controllers is compared. In all examples, we use parameters  $\Delta = 0$ ,  $V_j = 1$ ,  $\alpha_i = 1$ ,  $\beta_i = 1$ .

**TSP Benchmark Comparison:** We use the CRH controller as a path planning algorithm for some benchmark TSP problems. Table I shows the result of the 2-step and 3-step

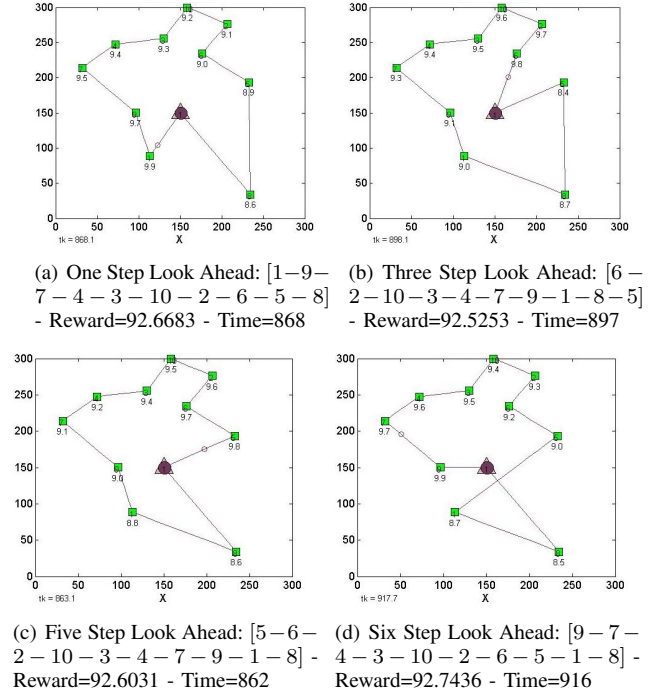


Fig. 4. 10 Target mission with different number of look ahead steps

TABLE I  
TSP BENCHMARK INSTANCES COMPARISON WITH THE CRH CONTROLLER ALGORITHM

TSP Instance	Optimal Tour Length	Two Step Lookahead	Three Step Lookahead	Limited Range Agent	Minimum Error (%)
att48	33522	38011	37492	41112	11.8
eil51	426	547	480	507	12.6
berlin52	7542	8713	8713	8137	7.8
st70	675	840	816	816	20.8
eil76	538	633	635	655	17.6
pr76	108159	146980	131678	146944	21.7
rat99	1211	1451	1470	1591	19.8
rd100	7910	9529	9123	9618	15.3
kroA100	21282	25871	24795	23782	11.7
kroB100	22141	28093	27415	28581	23.8
kroC100	20749	24603	25561	26171	18.5

Lookahead algorithm compared to the optimal results from [23]. We emphasize that the CRH controller is not designed for deterministic TSP problems so it is not expected to perform as well as highly efficient TSP algorithms. Nonetheless, as a starting basis of comparison, we note that the errors are relatively small, ranging from 7.8 to 23.8%.

In an attempt to measure the sensitivity of the results of the new CRH controller to partial mission information, we also tested cases where agents have limited sensing range (see fifth column in table I). In these cases, the agent only senses a target if it is within its sensing range which we have assumed to be 20% of the maximum dimension of the mission space. The results in most cases are comparable to the full-information cases. The computation time for the limited range agents is about an order of magnitude shorter than the other one. These results show the low sensitivity of the CRH controller performance to non-local information for each agent. This observation suggests that CRH controller is likely to provide good performance in a

distributed implementation or in cases where targets are not known a priori and should be locally sensed by the agents.

**Addressing Instabilities:** As already mentioned, the original CRH controller may give rise to oscillatory trajectories and fail to complete a mission. This is illustrated in Fig. 5(a) for a simple mission with three linearly discounted reward targets. In Fig. 5(b), it is shown that the new CRH controller can easily determine the optimal path in this simple case.

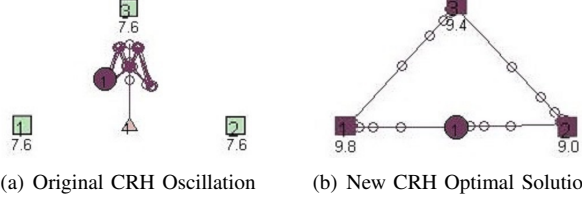


Fig. 5. Comparison of the two CRH controllers for a 3 targets mission

**Comparison between original and new CRH Controller:** A mission with 25 targets distributed uniformly and 2 agents starting at a base is considered as shown in Fig. 6(a), with uniformly distributed initial rewards:  $\lambda_i \sim U(10, 20)$  and  $D_i \sim U(300, 600)$  as in (1). In this case, the original CRH (Fig. 6(b)) underperforms compared to 3-step and 5-step Lookahead CRH controller (Figs. 6(c), 6(d)) by a large margin. We have used a value of  $\gamma = 0.3$  and  $I = 25$  in (14). This comes at the price of a slightly longer mission time in the 3-Step look ahead case, since the original controller never reaches some targets before their rewards are lost. However, minimizing time is not an objective of the MRCP considered here and reward maximization dictates the final length of the mission.

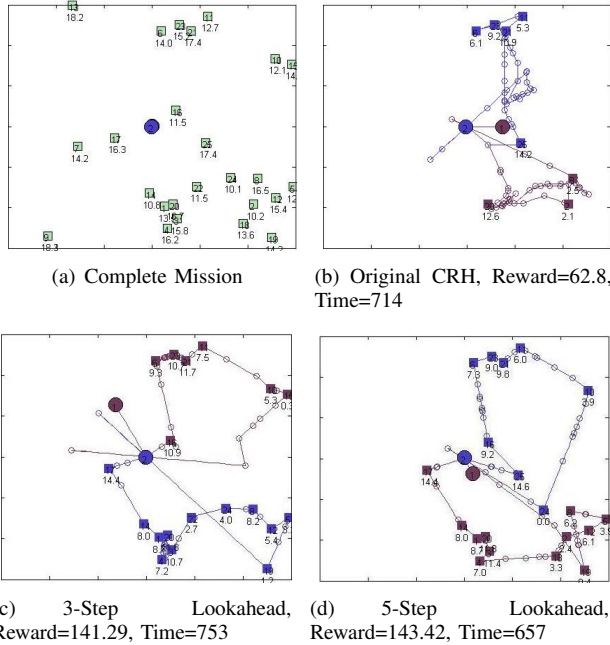


Fig. 6. Performance comparison of the original and new CRH algorithms

**Randomly Generated Missions:** To compare the overall performance of the new CRH controller, we generated 10

TABLE II  
20 TARGET-2 AGENT MISSIONS

Mission #	Original CRH		Three Step Lookahead CRH	
	Reward	Travel Time	Reward	Travel Time
1	33.92	412	45.24	536
2	41.48	439	52.4	426
3	30.93	476	41.19	483
4	32.08	389	37.24	457
5	41.5	444	47.25	537
6	44.61	389	47.91	471
7	23.93	528	35.48	462
8	38.68	415	50.91	489
9	30.92	478	34.08	429
10	36.81	458	44.26	476
Average	35.48	443	43.53	479

TABLE IV  
EFFECT OF THE SPARSITY FACTOR  $\zeta_i$  IN CLUSTERED MISSIONS

Mission #	$\gamma = 0$		$\gamma = 0.3$	
	Reward	Travel Time	Reward	Travel Time
1	40.62	552	61.9	413
2	64.89	447	64.64	420
3	35.24	471	63.8	461
4	63.78	465	64.64	478
5	25.42	493	26.5	449
6	22	454	22	454
7	44.1	458	46.84	449
8	34.26	466	61.21	472
Average	41.29	475	51.44	449

missions, each with 20 targets that are uniformly located in a  $300 \times 300$  mission space and two agents initially at the base. We have used  $\lambda_i \sim U(2, 12)$  and  $D_i = 300$ . The results are shown in Table II where we can see that the average total reward is increased by 22% while the average mission time is increased by 8%. In another case 10 missions were generated, each with 20 targets where 10 targets are only initially available to the agents. The other 10 targets would randomly appear during the mission. We use an initial reward  $\lambda_i \sim U(2, 12)$  and the parameter  $D_i \sim U(300, 600)$ . The comparison of the original and new CRH controller is shown in table III. An increase of 31% is seen in the total reward with a slight 2% increase in the total mission time.

TABLE III  
20 TARGET-2 AGENT MISSIONS, WITH RANDOM TARGET APPEARANCE

Mission #	Original CRH		Two Step Lookahead CRH	
	Reward	Travel Time	Reward	Travel Time
1	51.63	704	58.61	736
2	46.53	716	67.57	632
3	37.59	646	54.42	691
4	31.71	929	53.13	941
5	60.05	668	81.31	528
6	58.16	609	67.91	688
7	45.81	739	61.29	760
8	49.71	722	59.47	732
9	42.64	822	47.95	818
10	40.32	648	58.01	868
Average	46.42	720	60.97	739

**Sparsity Factor in Clustered Missions:** We considered 8 random mission with 20 targets that are located uniformly in one case and in 9 clusters in a second case. The goal here is to investigate the contribution of the sparsity factor  $\zeta_i$  in (14). We have again used  $\lambda_i \sim U(2, 12)$  and  $D_i = 300$ . We consider a case with  $\gamma = 0$  which eliminates the effect of  $\zeta_i$  and a second case with  $\gamma = 0.3$  and  $K = 5$  in (14). The results in table IV indicate that in the clustered missions rewards are improved by about 24% whereas in the uniform cases the reward is unaffected on average.



## VII. CONCLUSIONS AND FUTURE WORK

In this work a new CRH controller was developed for solving cooperative multi-agent problems in uncertain environments using the framework of the previous work in [20]. We overcame several limitations of the controller developed in [20], including agent trajectory instabilities and inaccurate estimation of a reward-to-go function while improving the overall performance. The event-driven CRH controller is developed to solve the MRCP, where multiple agents cooperate to maximize the total reward collected from a set of stationary targets in the mission space. The mission environment is uncertain, for example targets can appear at random times and agents might have a limited sensing range. The controller sequentially solves optimization problems over a planning horizon and executes the control for a shorter action horizon, where both are defined by certain events associated with new information becoming available. Unlike the earlier CRH controller, the feasible control set is finite instead of an infinite dimensional set. In the numerical comparisons, we showed that the new CRH controller has a better performance than the original one. In future work, the same framework will be applied to problems such as data harvesting where each target is generating data that should be collected and delivered to the base. Here the base will act as a target with dynamic reward. Also the new CRH controller can be extended into a decentralized version where each agent is responsible for calculating its own control.

## APPENDIX

**Proof of Lemma 1** From the definition of  $\eta_i(x, t)$  in (15) and  $\mathcal{C}_{l,j}(t_k, H_k)$  in (18) we have:

$$d(\mathcal{C}_{l,j}(t_k, H_k), \mathbf{y}_l) \leq d(\mathbf{x}, \mathbf{y}_l), \quad \forall \mathbf{x} \in \mathcal{F}_j(t_k, H_k) \quad (30)$$

Dividing both sides by  $\lambda_l D_l^{-1}$  and adding  $\zeta_l(t_k + H_k)$  we get, for all  $\mathbf{x} \in \mathcal{F}_j(t_k, H_k)$ ,

$$\eta_l(\mathcal{C}_{l,j}(t_k, H_k), t_k + H_k) \leq \eta_l(\mathbf{x}, t_k + H_k) \quad (31)$$

To prove the forward lemma statement, we use a contradiction argument and assume there exists a target  $r$  such that

$$\eta_l(\mathcal{C}_{l,j}(t_k, H_k), t_k + H_k) > \eta_r(\mathcal{C}_{l,j}(t_k, H_k), t_k + H_k)$$

Using (31), we get  $\eta_r(\mathcal{C}_{l,j}(t_k, H_k), t_k + H_k) < \eta_l(\mathbf{x}, t_k + H_k)$  for all  $\mathbf{x} \in \mathcal{F}_j(t_k, H_k)$ . This implies that there exists no  $\mathbf{x} \in \mathcal{F}_j(t_k, H_k)$  such that  $l = \arg\min_i \eta_i(\mathbf{x}, t_k + H_k)$ . Therefore,  $l$  cannot be an active target, which contradicts the assumption, hence (19) is true.

To prove the reverse statement, we assume that (19) holds for any  $i \in \mathcal{T}_k$ , i.e.,

$$\eta_l(\mathcal{C}_{l,j}(t_k, H_k), t_k + H_k) < \eta_i(\mathcal{C}_{l,j}(t_k, H_k), t_k + H_k)$$

By the definition of active targets (16), we then know that  $l$  is an active target for agent  $j$  at time  $t_k$ . ■

**Proof of Lemma 2** The active target set creates a partition of the set  $\mathcal{F}_j(t_k, H_k)$  where each subset is an arc in a Euclidean mission space. For an active target  $l \in S_j(t_k, H_k)$ , let the  $l$ th arc be  $\mathcal{F}_j^l(t_k, H_k) \subset \mathcal{F}_j(t_k, H_k)$ . For each  $\mathcal{F}_j^l(t_k, H_k)$ , we prove that the heading  $\mathbf{v}^* = v(\mathcal{C}_{l,1}(t_k, H_k))$  satisfies, for

all  $\mathbf{x} \in \mathcal{F}_j^l(t_k, H_k)$ :

$$J_{\mathbf{I}}(\mathbf{v}^*, t_k, H_k) + J_{\mathbf{A}}(\mathbf{v}^*, t_k, H_k) > J_{\mathbf{I}}(v(\mathbf{x}), t_k, H_k) + J_{\mathbf{A}}(v(\mathbf{x}), t_k, H_k)$$

There are two possible cases:

*Case 1:*  $\mathbf{y}_l \in \mathcal{F}_1(t_k, H_k)$ . This means  $d(\mathbf{y}_l, \mathbf{x}_1(t)) = H_k$ . Also, from (18), this guarantees that  $\forall r \in \mathcal{T}_k$ :

$$q_r(\mathcal{C}_{r,1}(t_k + H_k)) = \begin{cases} 1 & \text{if } r = l \\ 0 & \text{otherwise} \end{cases}$$

Setting  $\tilde{\tau}_r(\mathbf{v}^*, t_k, H_k) = \tilde{\tau}_r^*$ , we have

$$\begin{aligned} J(\mathbf{v}^*, t_k, H_k) &= J_{\mathbf{I}}(\mathbf{v}^*, t_k, H_k) + J_{\mathbf{A}}(\mathbf{v}^*, t_k, H_k) \\ &= \lambda_l \phi_l(t_k + H_k) + \sum_{r=1}^{M_{k+1}} \lambda_r \phi_r(\tilde{\tau}_r^*) q_l(\mathbf{x}_1(\tilde{\tau}_r^*)) \end{aligned}$$

Here,  $M_{k+1} = |\mathcal{T}_{k+1}|$  and  $\mathcal{T}_{k+1} = \mathcal{T}_k - \{l\}$  since reward  $l$  will be already collected at time  $t_k + H_k$ . The estimated visit time  $\tilde{\tau}_r^*$  is determined based on a tour  $\theta$  that starts at point  $\mathbf{y}_l$ . Now let us calculate the objective function for any other heading  $v(\mathbf{x})$  where  $\mathbf{x} \in \mathcal{F}_1^l(t_k, H_k)$ . Setting  $\tilde{\tau}_r(v(\mathbf{x}), t_k, H_k) = \tilde{\tau}_r$ ,

$$\begin{aligned} J(v(\mathbf{x}), t_k, H_k) &= J_{\mathbf{I}}(v(\mathbf{x}), t_k, H_k) + J_{\mathbf{A}}(v(\mathbf{x}), t_k, H_k) \\ &= 0 + \sum_{r=1}^{M'_{k+1}} \lambda_r \phi_r(\tilde{\tau}_r) q_l(\mathbf{x}_1(\tilde{\tau}_r)) \end{aligned}$$

since  $\mathbf{x} \notin \mathcal{C}_{l,1}(t_k, H_k)$  so that  $q_r(\mathbf{x}) = 0$  for all  $r \in \mathcal{T}_k$ . The aggregated tour is determined over the set  $\mathcal{T}'_{k+1} = \mathcal{T}_k$  starting at  $\mathbf{x} \in \mathcal{F}_1^l(t_k, H_k)$ . By definition, the target with the least travel cost from point  $\mathbf{x}$  is the active target  $l$  and this is the first target in the tour. The rest of the tour consists of targets in  $\mathcal{T}_{k+1} - \{l\}$  starting at  $\mathbf{y}_l$ . Let us call this tour  $\theta'$ . Since in both tours  $\theta$  and  $\theta'$  the starting point and the set of available targets are the same, the order of targets will be identical and we have  $\theta' = \{l, \theta\}$ . The visit times in  $\theta$  are given by

$$\tilde{\tau}_{\theta'_n}^* = t_k + H_k + \sum_{i=1}^{n-1} d(\mathbf{y}_{\theta'_i}, \mathbf{y}_{\theta'_{i+1}})$$

In  $\theta'$ , the visit time for target  $\theta'_1 = l$  is:  $\tilde{\tau}_{\theta'_1} = t_k + H_k + d(\mathbf{x}, \mathbf{y}_l)$ . For the rest of the targets, with  $1 < n \leq M'_{k+1}$ ,

$$\tilde{\tau}_{\theta'_{(n),1}} = t_k + H_k + d(\mathbf{x}, \mathbf{y}_l) + \sum_{i=1}^{n-1} d(\mathbf{y}_{\theta'_i}, \mathbf{y}_{\theta'_{i+1}})$$

For all  $1 < n \leq M_{k+1}$ , we have  $\theta'_{n+1} = \theta_n$  and  $\tilde{\tau}_{\theta'_{n+1}} > \tilde{\tau}_{\theta_n}$ . By assumption, for all  $i \in \mathcal{T}$ ,  $\phi_i(t)$  is non-increasing, therefore  $\phi_{\theta'_{n+1}}(\tilde{\tau}_{\theta'_{n+1}}) \leq \phi_{\theta_n}(\tilde{\tau}_{\theta_n})$ , and it follows that

$$\begin{aligned} &\lambda_l \phi_l(t_k + H_k + d(\mathbf{x}, \mathbf{y}_l)) + \sum_{n=2}^{M'_{k+1}} \lambda_{\theta'_n} \phi_{\theta'_n}(\tilde{\tau}_{\theta'_n}) \\ &\leq \lambda_l \phi_l(t_k + H_k) + \sum_{n=1}^{M_{k+1}} \lambda_{\theta_n} \phi_{\theta_n}(\tilde{\tau}_{\theta_n}) \end{aligned}$$

The right-hand-side above is  $J(\mathbf{v}^*, t_k, H_k)$  and the left-hand-side is  $J(v(\mathbf{x}), t_k, H_k)$ , so we have proved that for any  $\mathbf{x} \in \mathcal{F}_1^l(t_k, H_k)$ ,  $\mathbf{x} \notin \mathcal{C}_{l,1}(t_k, H_k)$  we have  $J(v(\mathbf{x}), t_k, H_k) \leq J(\mathbf{v}^*, t_k, H_k)$ .

Case 2:  $\mathbf{y}_l \notin \mathcal{F}_1(t_k, H_k)$ . In this case, for any point  $\mathbf{x} \in \mathcal{F}_j^l(t_k, H_k)$  we have a zero immediate reward. Thus, only the rewards-to-go need to be compared. Using (17), for any  $\mathbf{x} \in \mathcal{F}_j^l(t_k, H_k)$  we know the aggregation tour  $\theta$  for any point  $\mathbf{x}$  starts with target  $l$  and the rest of it would also be the same. Similarly, let us assume  $\theta$  is the tour for  $\mathbf{v}^*$  and  $\theta'$  is the tour for any other point  $\mathbf{x}$ . The estimated visit times for  $\theta$  are:

$$\tilde{\tau}_{\theta_n}^* = t_k + H_k + d(\mathbf{y}_l, \mathcal{C}_{l,1}(t_k, H_k)) + \sum_{i=1}^{n-1} d(\mathbf{y}_{\theta_i}, \mathbf{y}_{\theta_{i+1}})$$

and for  $\theta'$ :

$$\tilde{\tau}_{\theta_n}^* = t_k + H_k + d(\mathbf{y}_l, \mathbf{x}) + \sum_{i=1}^{n-1} d(\mathbf{y}_{\theta_i}, \mathbf{y}_{\theta_{i+1}})$$

By the definition in (18),  $\mathcal{C}_{l,1}(t_k, H_k)$  is on the shortest path from  $\mathbf{x}_j(t_k)$  to  $\mathbf{y}_l$ , i.e.,  $\tilde{\tau}_{\theta_n}^* > \tilde{\tau}_{\theta_n}$ . Again, with  $\phi_i(t)$  being non-increasing we have  $\phi_{\theta_n}(\tilde{\tau}_{\theta_n}^*) \leq \phi_{\theta_n}(\tilde{\tau}_{\theta_n})$ , which implies  $J(v(\mathbf{x}), t_k, H_k) \leq J(\mathbf{v}^*, t_k, H_k)$ .

We have thus proved the lemma statement that the optimal heading of the agent is one of the direct headings towards an active target. ■

**Proof of Theorem 1** In the multi-agent mission, calculating the immediate reward and reward-to-go in (12) and (23) for each agent is like a one-agent mission limited to its own target subset  $\mathcal{T}_{k,j}$ . Therefore, the result follows directly from Lemma 2. ■

**Proof of Theorem 2** We assume WLOG that  $d(\mathbf{x}, \mathbf{y}_1) < d(\mathbf{x}, \mathbf{y}_2)$  so that at time  $t_k$  we have  $H_k = d(\mathbf{x}, \mathbf{y}_1)$ . This implies that target 1 is always an active target (the travel cost of target 1 at time  $t_{k+1} = t_k + H_k$  is equal to 0). Recalling (18) and setting  $\mathcal{C}_{2,1} = \mathcal{C}_{2,1}(t_k, H_k)$ , we have  $d(\mathbf{x}, \mathbf{y}_1) = d(\mathbf{x}, \mathcal{C}_{2,1}) = H_k$ . This results in:

$$d(\mathbf{x}, \mathbf{y}_2) = d(\mathbf{x}, \mathbf{y}_1) + d(\mathbf{y}_2, \mathcal{C}_{2,1}) \quad (32)$$

From Lemma 1, target 2 is an active target if and only if  $\eta_2(\mathcal{C}_{2,1}, t_k + H_k) \leq \eta_1(\mathcal{C}_{2,1}, t_{k+1})$ . Therefore, from (15), target 2 is an active target if and only if:

$$\frac{d(\mathcal{C}_{2,1}, \mathbf{y}_2)}{\lambda_2 D_2^{-1}} \leq \frac{d(\mathcal{C}_{2,1}, \mathbf{y}_1)}{\lambda_1 D_1^{-1}}$$

which is rewritten as:

$$\frac{\lambda_1}{D_1} d(\mathcal{C}_{2,1}, \mathbf{y}_2) \leq \frac{\lambda_2}{D_2} d(\mathcal{C}_{2,1}, \mathbf{y}_1)$$

We now consider two possible cases regarding target 2. First, assume target 2 is not an active target, i.e.,

$$\frac{\lambda_1}{D_1} d(\mathcal{C}_{2,1}, \mathbf{y}_2) > \frac{\lambda_2}{D_2} d(\mathcal{C}_{2,1}, \mathbf{y}_1) \quad (33)$$

Starting with the trivial inequality:

$$0 > \frac{-\lambda_1}{D_1} [d(\mathcal{C}_{2,1}, \mathbf{y}_2) + d(\mathcal{C}_{2,1}, \mathbf{y}_1)]$$

add  $\frac{\lambda_2}{D_2} [d(\mathcal{C}_{2,1}, \mathbf{y}_1)]$  to both sides and use (33) to get:

$$\begin{aligned} \frac{\lambda_1}{D_1} [d(\mathcal{C}_{2,1}, \mathbf{y}_2)] &> \frac{\lambda_2}{D_2} [d(\mathcal{C}_{2,1}, \mathbf{y}_1)] > \\ \frac{-\lambda_1}{D_1} [d(\mathcal{C}_{2,1}, \mathbf{y}_2)] &+ \left(\frac{\lambda_2}{D_2} - \frac{\lambda_1}{D_1}\right) d(\mathcal{C}_{2,1}, \mathbf{y}_1) \end{aligned}$$

Adding the positive quantity of  $\frac{\lambda_2}{D_2} [d(\mathcal{C}_{2,1}, \mathbf{y}_2)]$  to both sides

and invoking the triangle inequality:

$$\begin{aligned} &\left(\frac{\lambda_1}{D_1} + \frac{\lambda_2}{D_2}\right) [d(\mathcal{C}_{2,1}, \mathbf{y}_2)] \\ &> \left(\frac{\lambda_2}{D_2} - \frac{\lambda_1}{D_1}\right) [d(\mathcal{C}_{2,1}, \mathbf{y}_2)] + \left(\frac{\lambda_2}{D_2} - \frac{\lambda_1}{D_1}\right) [d(\mathcal{C}_{2,1}, \mathbf{y}_1)] \\ &> \left(\frac{\lambda_2}{D_2} - \frac{\lambda_1}{D_1}\right) d(\mathbf{y}_1, \mathbf{y}_2) \end{aligned}$$

Rearranging the last inequality and using (32) results in:

$$\begin{aligned} &\frac{\lambda_1}{D_1} [d(\mathbf{x}, \mathbf{y}_2) + d(\mathbf{y}_2, \mathbf{y}_1)] + \frac{\lambda_2}{D_2} d(\mathbf{x}, \mathbf{y}_2) \\ &> \frac{\lambda_1}{D_1} d(\mathbf{x}, \mathbf{y}_1) + \frac{\lambda_2}{D_2} [d(\mathbf{x}, \mathbf{y}_1) + d(\mathbf{y}_2, \mathbf{y}_1)] \end{aligned} \quad (34)$$

which is the same as (29) implying that path  $\theta_1 = (1, 2)$  is optimal, i.e., the CRH controller finds the optimal path.

Next, assume that target 2 is also an active target along with target 1. Let  $u_1$  and  $u_2$  be the headings for target 1 and 2 respectively, i.e.,  $\mathbf{x}_1(t_{k+1}, u_1) = \mathbf{y}_1$  and  $\mathbf{x}_1(t_{k+1}, u_2) = \mathcal{C}_{2,1}$ . The objective function of the CRH controller under  $u_1$  and  $u_2$  is:

$$\begin{aligned} J(u_1, t_k, H_k) &= J_{\mathbf{I}}(u_1, t_k, H_k) + J_{\mathbf{A}}(u_1, t_k, H_k) \\ &= \lambda_1 \phi_1(t_{k+1}) + \lambda_2 \phi_2(t_{k+1} + d(\mathbf{y}_1, \mathbf{y}_2)) \end{aligned}$$

$$\begin{aligned} J(u_2, t_k, H_k) &= J_{\mathbf{I}}(u_2, t_k, H_k) + J_{\mathbf{A}}(u_2, t_k, H_k) \\ &= 0 + [\lambda_2 \phi_2(t_{k+1} + d(\mathcal{C}_{2,1}, \mathbf{y}_2)) \\ &\quad + \lambda_1 \phi_1(t_{k+1} + d(\mathcal{C}_{2,1}, \mathbf{y}_2) + d(\mathbf{y}_1, \mathbf{y}_2))] \end{aligned}$$

Note that in order to evaluate the objective function for  $u_2$  we find a tour starting at point  $\mathcal{C}_{2,1}$  which goes to the target with minimum travel cost. However, for target 2 to be active at  $t_k$  it has to have the smallest travel cost at that point, which results in  $J_{\mathbf{A}}(u_2, t_k, H_k)$  to be the reward of going to target 2 and then target 1. We can see that using the reward of each path from (27) and (28) we can write:

$$J(u_1, t_k, H_k) = R_{(1,2)}, \quad J(u_2, t_k, H_k) = R_{(2,1)}$$

Thus, the objective function of the CRH controller under  $u_1$  and  $u_2$  is identical to the corresponding path rewards. Hence, the CRH controller selects the correct optimal heading at  $t_k$ . ■

## REFERENCES

- [1] J. S. Shamma, *Cooperative control of distributed multi-agent systems*. Wiley Online Library, 2007.
- [2] R. Murphey and P. M. Pardalos, *Cooperative control and optimization*, vol. 66. Springer, 2002.
- [3] A. Jadbabaie, J. Lin, and A. S. Morse, "Coordination of groups of mobile autonomous agents using nearest neighbor rules," *IEEE Trans. on Automatic Control*, vol. 48, no. 6, pp. 988–1001, 2003.
- [4] T. McLain, P. Chandler, S. Rasmussen, and M. Pachter, "Cooperative control of UAV rendezvous," *Proc. of American Control Conference*, pp. 2309–2314, 2001.
- [5] C. Yao, X. C. Ding, and C. Cassandras, "Cooperative receding horizon control for multi-agent rendezvous problems in uncertain environments," in *49th IEEE Conference on Decision and Control (CDC), 2010*, pp. 4511–4516, Dec. 2010.
- [6] C. Cassandras, X. Lin, and X. Ding, "An optimal control approach to the multi-agent persistent monitoring problem," *IEEE Transactions on Automatic Control*, vol. 58, pp. 947–961, April 2013.
- [7] J. Cortes, S. Martinez, T. Karatas, and F. Bullo, "Coverage control for mobile sensing networks," *Robotics and Automation, IEEE Transactions on*, vol. 20, no. 2, pp. 243–255, 2004.

- [8] M. Zhong, , and C. G. Cassandras, "Distributed coverage control and data collection with mobile sensor networks," *IEEE Transactions on Automatic Control*, vol. 56, no. 10, pp. 2445–2455, 2011.
- [9] D. Panagou, M. Turpin, and V. Kumar, "Decentralized goal assignment and trajectory generation in multi-robot networks," 2014.
- [10] W. Ren and R. Beard, *Distributed consensus in multi-vehicle cooperative control: theory and applications*. Springer, 2008.
- [11] M. Zhong and C. G. Cassandras, "Asynchronous distributed optimization with event-driven communication," *IEEE Transactions on Automatic Control*, vol. 55, no. 12, pp. 2735–2750, 2010.
- [12] N. P. Salz, "Anon - a theory for traveling salesman problem," *Operations Research*, vol. S 14, 1966.
- [13] D. L. Applegate, R. E. Bixby, V. Chvatal, and W. J. Cook, *The traveling salesman problem: a computational study*. Princeton University Press, 2011.
- [14] G. Laporte, "The vehicle routing problem: An overview of exact and approximate algorithms," *European Journal of Operational Research*, vol. 59, no. 3, pp. 345 – 358, 1992.
- [15] A. Ekici and A. Retharekar, "Multiple agents maximum collection problem with time dependent rewards," *Computers and Industrial Engineering*, vol. 64, no. 4, pp. 1009 – 1018, 2013.
- [16] H. Tang, E. Miller-Hooks, and R. Tomastik, "Scheduling technicians for planned maintenance of geographically distributed equipment," *Transportation Research Part E: Logistics and Transportation Review*, vol. 43, no. 5, pp. 591 – 609, 2007.
- [17] J. S. Bellingham, M. Tillerson, M. Alighanbari, and J. P. How, "Co-operative path planning for multiple UAVs in dynamic and uncertain environments," in *IEEE Conference on Decision and Control (CDC)*, pp. 2816–2822 vol.3, 10-13 December 2002.
- [18] M. G. Earl and R. D'Andrea, "A decomposition approach to multi-vehicle cooperative control," *Robotics and Autonomous Systems*, vol. 55, pp. 276–291, 2007.
- [19] D. Q. Mayne, J. B. Rawlings, C. V. Rao, and P. O. M. Scokaert, "Constrained model predictive control: Stability and optimality," *Automatica*, vol. 36, no. 6, pp. 789–814, 2000.
- [20] W. Li and C. Cassandras, "A cooperative receding horizon controller for multivehicle uncertain environments," *IEEE Transactions on Automatic Control*, vol. 51, no. 2, 2006.
- [21] W. Li and C. G. Cassandras, "Centralized and distributed cooperative receding horizon control of autonomous vehicle missions," *Mathematical and computer modelling*, vol. 43, no. 9, pp. 1208–1228, 2006.
- [22] J. J. Schneider, T. Bukur, and A. Krause, "Traveling salesman problem with clustering," *Journal of Statistical Physics*, vol. 141, no. 5, pp. 767–784, 2010.
- [23] G. Reinelt, "TSPLIB: A traveling salesman problem library," *ORSA Journal on Computing*, vol. 3, no. 4, pp. 376–384, 1991.