Persistent Intelligence, Surveillance, and Reconnaissance Using Multiple Autonomous Vehicles With Asynchronous Route Updates

Cameron K. Peterson, David W. Casbeer, Satyanarayana G. Manyam, and Steven Rasmussen

***Abstract*—Providing persistent intelligence, reconnaissance, and surveillance of targets is a challenging, but important task when time-critical information is required. In this letter, we provide a decentralized routing algorithm for coordinating multiple autonomous vehicles as they visit a discrete set of pre-defined targets with weighted revisit priorities. The algorithm utilizes a block coordinate ascent algorithm combined with a Monte Carlo tree search to tractably decide each vehicle's route. The result is a non-myopic algorithm for multiple vehicles that is decentralized, computationally tractable, and allows for target prioritization. Guarantees are provided that all targets will have finite revisit times and that the block coordinate ascent algorithm will converge to a block optimal solution. Numerical simulations illustrate the utility of this method by showing that the results are comparable to those of a centralized exhaustive search and that they degrade gracefully with limited communication and scale under increasing numbers of targets and vehicles.**

***Index Terms*—Cooperating robots, multi-robot systems, path planning for multiple mobile robots or agents.**

I. Introduction

PROVIDING persistent intelligence, reconnaissance, and surveillance (PISR) of target locations is beneficial for many scientific and military applications [1], [2]. This is an ideal application for cooperating unmanned air vehicles (UAVs) that can autonomously monitor targets. UAVs, however, are typically constrained in their payload capacity which results in limited computational and communication capabilities. This paper presents a cooperative routing algorithm that satisfies typical UAV constraints while guiding them to persistently monitor locations of interest.

Manuscript received February 22, 2020; accepted June 20, 2020. Date of publication July 9, 2020; date of current version July 21, 2020. This letter was recommended for publication by Associate Editor A. Prorok and Editor Nak Young Chong upon evaluation of the reviewers' comments. This work was supported by the US Air Force Research Laboratory (AFRL) under the summer faculty fellowship program. *(Corresponding author: Cameron K. Peterson.)*

Cameron K. Peterson is with the Electrical and Computer Engineering Department, Brigham Young University, Provo, UT 84602 USA (e-mail: cammy.peterson@byu.edu).

David W. Casbeer is with the Control Center of Excellence, Air Force Research Laboratory, WPAFB, Dayton, OH 45435, USA (e-mail: david.casbeer@us.af.mil).

Satyanarayana G. Manyam is with Infoscitex Corporation, Dayton, OH 45431 USA (e-mail: msngupta@gmail.com).

Steven Rasmussen is with Miami Valley Aerospace LLC, AFRL, Wright-Patterson AFB, Dayton, OH 45435 USA (e-mail: steven.rasmussen.5.ctr@us.af.mil).

Digital Object Identifier 10.1109/LRA.2020.3008140

The cooperative route planning algorithm is designed to minimize the maximum weighted revisit-times of targets. It achieves this by maximizing an objective function that rewards revisiting targets not recently observed. Targets are given weights that prioritize the value in monitoring each location. Due to the weighting, a single target may be visited repeatedly prior to all the other targets being visited.

The PISR application is similar to the well-studied multi-vehicle travelling salesman problem. However, it differs by allowing targets to be revisited more than once per tour [3]. In [4] a closed-walk path that allowed multiple target visits was found using a mixed integer linear programming. The solution assumed a pre-determined number of target visits and was solved for a single vehicle with equal target weighting.

The work in [5] provided a variant of the travelling salesman problem where all target locations are visited exactly once, but are interspersed with visits to a depot to transmit gathered information. The trade-off between visiting the depot often (to promptly disseminate gathered information) and minimizing the cycle time (with infrequent visits to the depot) is handled using a linear program that scales exponentially in the number of targets. To combat the poor scaling inherent in exact solutions, the authors of [6] determined routes using genetic algorithms (GA) for a single vehicle.

In [7] the practical challenges of implementing a routing algorithm with multiple cooperating vehicles were addressed. The decentralized solution provided scalable optimization using heuristic methods and assuming a myopic control policy. Preliminary flight tests, using two unmanned vehicles, showed promising results. However, the decentralized implementation assumed all vehicles were provided with identical inputs, a restriction the authors found challenging to enforce.

Similar to [7], this work focuses on the practical implementations of the multiple vehicle routing problem with the goal to execute on hardware typical for a small to medium sized unmanned vehicle. This necessitates solving the objective function heuristically, since exact solutions are too costly [2]. We mitigate many shortcomings found in prior research by providing a decentralized, non-myopic, route-planning algorithm that is computationally tractable, incorporates multiple vehicles, allows for target prioritization, and degrades gracefully under communication loss.

We choose vehicle routes using a combined block coordinate ascent (BCA) optimization with a Monte Carlo tree search

(MCTS). The BCA enables each vehicle to plan independently whereas MCTS judiciously selects which of the potential routes to consider, mitigating the need for an exhaustive search. Routes are planned to a pre-determined event horizon, then replanned when a UAV reaches a target. Targets are assumed to be static and deterministic with positions and weights known prior to operations. Since targets have nonuniform distances between each other, routes have unique execution times necessitating asynchronous vehicle planning.

Block coordinate ascent algorithms are derivative-free optimization approaches where a block of variables or coordinates is optimized conditioned on all other variables/coordinates being held constant [8]. In this paper, the route of an individual vehicle is parameterized by its policy, an ordered list of fixed length containing targets to visit. The vehicle's policy defines a block of variables that are optimized, assuming all other vehicle routes (peer blocks) remain constant. BCA reduces the computational complexity of the planning algorithm while providing an intuitive approach for decentralized route-planning that is robust to lost or delayed communication messages.

Coordinate ascent algorithms typically require strict assumptions, such as the objective function being strictly convex [9], to ensure convergence (even to a local minima). If these requirements are not met then it is possible that the solution will cycle infinitely [8]. However, for this application, despite using a non-convex objective function, we can guarantee that the BCA algorithm will converge to a block optimal solution where each vehicle's policy becomes fixed. This is a necessary condition for guaranteeing that targets will have finite revisit times.

To further reduce the computational complexity in the route-finding algorithm, we incorporate MCTS to heuristically solve BCA. MCTS is a tree-search method where the nodes selected provide a balance between the exploration of new branches with the exploitation of ones known to have a high reward [10]. It is a popular approach to solving challenging problems, such as the AI version of the game Go [10] and has been extended to various path planning algorithms [11]–[13].

The contribution of this paper is the development of a novel route finding algorithm that finds good solutions for the weighted PISR problem for multiple vehicles. The algorithm uses MCTS in an atypical way to account for the uncertainty present in approximating an infinite sequence with a finite horizon. When combined with a BCA optimization the route finding algorithm is decentralized, computationally tractable, and degrades gracefully with communication limitations. Furthermore, we show that it will converge to a block optimal solution, guaranteeing that all targets have finite revisit times.

This paper proceeds as follows. Section II introduces the problem statement. For a single vehicle, guarantees for finite revisit times are provided in Section III. The multi-vehicle case is presented in Section IV where the BCA/MCTS algorithm is also described. Computational and simulation results are provided in Section V and conclusions are made in Section VI.

II. Vehicle Routing Objective Function

Vehicle routes are chosen to minimize the weighted revisit times to targets. This incentivizes frequent observation of high priority targets. However, we also consider the vehicle's travel time to avoid choosing routes with unnecessarily long paths. This combination ensures that targets with large weighted revisit times will be prioritized while concurrently reducing the path lengths. In this section, our notation is closely aligned with that presented in [7].

Let a control policy for vehicle a be described by a sequence of states, $\pi_a = \{s_1, s_2, \ldots, s_L\}$ where each state $s_k = (v_i, t(v_i))$ is composed of a target v_i and the elapsed time since it was last visited, $t(v_i)$. The notation, $s_k[1] = v_i$ and $s_k[2] = t(v_i)$ is used to indicate the target of state s_k and its elapsed time, respectively. Elapsed times are reset when a vehicle arrives at a target's location, i.e. $t(v_i) = 0$. Since revisiting targets is permitted, the target indices in a policy are not necessarily unique.

The time it takes for the vehicle to travel from target v_i to v_{i+1} is $d(v_i, v_{i+1}) = \|x(v_{i+1}) - x(v_i)\|_2/\nu_a$ where $x(v_i)$ is the Cartesian position of target v_i, $\nu_a > 0$ is vehicle a's speed, and $\|\cdot\|_2$ is the Euclidean norm. We assume that the time distance between targets satisfy $d(v_i, v_j) < (d(v_i, v_\ell) + d(v_\ell, v_j))$ ensuring that targets are not observed while a vehicle is transiting.

Let $s_k[1] = v_i$ and $s_{k+1}[1] = v_j$. The cumulative time at which vehicle a will arrive at state s_{k+1} under control policy π_a is $T^{\pi_a}_{k+1} = T^{\pi_a}_k + d(v_i, v_j)$, with $T^{\pi_a}_1 = 0$.

The routing objective function is composed of an instantaneous reward for visiting a target and a discount factor that penalizes the time it takes to reach the target. The instantaneous reward is $r(s_k, \pi_a) = \omega_i(t(v_i) + T^{\pi_a}_k)$, where $\omega_i > 0$ is the weight of target v_i and $t(v_i)$ is the elapsed time since v_i was last seen by any vehicle. The revisit time when the vehicle will arrive at v_i is $t(v_i) + T^{\pi_a}_k$. This rewards the vehicle for visiting targets with high weighted elapsed times. To penalize the vehicle for travelling long distances, the instantaneous reward is combined with a discount factor $e^{-\beta T^{\pi_a}_k}$ where $\beta > 0$ is a discount gain.

Let $\boldsymbol{\pi} = \{\pi_1, \pi_2, \ldots, \pi_A\}$ be the concatenation of the policies for all the vehicles. The cumulative reward for executing the policies of multiple vehicles, each with event horizon L is

$$J(\boldsymbol{\pi}) = \sum_{k=1}^{L} \left(\sum_{a \in V} e^{-\beta T^{\pi_a}_k} r(s_k, \pi_a) \right), \tag{1}$$

where $V = \{1, \ldots, A\}$ is the set of vehicles available for the mission. The dependence of the reward on the elapsed time $t(v_i)$ requires that the objective function be evaluated in order of each vehicle's time of arrival at the targets. Therefore, the outer summation of (1) is often partially evaluated prior to fully evaluating the rewards for all the vehicles (inner summation).

This objective function encourages vehicles to visit high priority targets, but discounts the reward as a function of the cumulative time it takes to reach them. The discount factor is necessary because the event horizon specifies a number of target visits rather than a fixed clock time, as is typical for most receding horizon controllers. Without this factor the highest reward for a vehicle may be realized by maximizing its travel time, allowing the elapsed time for the targets to increase.

We assume that vehicles do not alter their next destination mid-flight, but continue on their current path until they arrive at the next target. Furthermore, the UAVs are assumed to travel at a constant altitude in straight lines from one target to another. We

assume that the difference in travel times between straight-line trajectories and dynamically feasible vehicle paths are negligible and can be ignored.

III. Single Vehicle

In this section, we consider routing a single vehicle. Given that targets may be visited repeatedly, we show that the vehicle does not get trapped into exclusively visiting a subset of the targets, ensuring that all targets will be visited in a finite time.

Let $\bar{t}_d(v_i)$ be the time when a vehicle will decide to visit target v_i. Vehicle's do not change course mid-route, therefore, once a decision is made, the target's revisit time will be the addition of $\bar{t}_d(v_i)$ with the time it takes the vehicle to complete its current route and then continue on to v_i.

We will prove that there is a finite bound on a target's revisit time. Rationale for this result was given for a single vehicle with a myopic control policy in [14] using a proof by contradiction. Using an inductive proof (Lemma 1), we re-prove the result with the addition of an upper bound on the target's revisit time. We then expand this result to include non-myopic control policies in Lemma 2 and multiple vehicles in Theorem 1.

The following two lemmas use definitions on the minimum and maximum time distances between targets within a set $N = \{v_1, v_2, \ldots, v_n\}$ and between targets in N and an additional target v_{n+1}. We define the minimum and maximum travel time between any two targets in N to be $\underline{d}_N = \min_{v_i, v_j \in N}\{d(v_i, v_j)\}$ and $\bar{d}_N = \max_{v_i, v_j \in N}\{d(v_i, v_j)\}$, respectively. Consider an additional target v_{n+1}. The minimum and maximum travel time between v_{n+1} and any target in N are defined as $\underline{d}_M = \min_{v_i \in N}\{d(v_i, v_{n+1})\}$ and $\bar{d}_M = \max_{v_i \in N}\{d(v_i, v_{n+1})\}$. Furthermore, let $\bar{\omega} = \max_{v_i \in N}\{\omega_i\}$ be the maximum weight value for any targets in N.

Lemma 1: Let N contain n targets which have a maximum revisit time bounded by $\bar{b}$. When a single vehicle ($V = 1$) with a one-step look-ahead policy ($L = 1$) is routed using the objective function given by (1), target v_{n+1} will have a finite revisit time. Furthermore, after an elapsed time greater than or equal to

$$\bar{t}_d(v_{n+1}) = \max\left\{ e^{-\beta(\underline{d}_N - \bar{d}_M)} \frac{\bar{\omega}}{\omega_{n+1}} (\bar{b} + \bar{d}_N) - \underline{d}_M, 0 \right\}, \tag{2}$$

the vehicle will decide to visit v_{n+1}. The bound on the revisit time for target v_{n+1} is $b_{n+1} = \bar{t}_d(v_{n+1}) + \bar{d}_N + \bar{d}_M < \infty$.

Proof: For a single vehicle, with $L = 1$, the one-step look-ahead policy is $\pi_1 = \{s_1, s_2\}$. Let the targets associated with the current and future state of this policy be $s_1[1] = v_i$ and $s_2[1] = v_j$, respectively. Under these conditions, (1) can be written as a function of v_i and v_j,

$$J(\pi_1) = J(v_i, v_j) = e^{-\beta d(v_i, v_j)} \omega_j (t(v_j) + d(v_i, v_j)). \tag{3}$$

The policy with the largest reward when visiting targets $N = \{v_1, \ldots, v_n\}$ is bounded by $\bar{J}^N_{\max}$ where

$$\max_{\pi_1}\{J(\pi_1)\} \leq e^{-\beta \underline{d}_N} \bar{\omega}(\bar{b} + \bar{d}_N) \triangleq \bar{J}^N_{\max}. \tag{4}$$

By induction, we will show that at time $\bar{t}_d(v_{n+1})$ (2) the reward for visiting target $s_2[1] = v_{n+1}$ exceeds the reward for visiting any target in N and the vehicle will proceed to v_{n+1}, once it completes its current route.

Base Case: With a single target $v_1 \in N$, once the vehicle arrives at v_1, the set will have a bounded revisit time of $\bar{b} = 0$ (i.e. the vehicle will always remain at v_1). In this set, $J(v_1, v_1) = \bar{J}^N_{\max}$ and there is no reward for remaining at target v_1, $J(v_1, v_1) = e^{-\beta d(v_1, v_1)} \omega_1 (t(v_1) + d(v_1, v_1)) = 0$, since both $t(v_1) = 0$ and $\bar{d}_N = d(v_1, v_1) = 0$.

The decision time, which cannot be negative, is $\bar{t}_d(v_2) = 0$ (2). Substituting $\bar{t}_d(v_2) = 0$ for the elapsed time into the objective function yields $J(v_1, v_2) = e^{-\beta d(v_1, v_2)} \omega_2 (\bar{t}_d(v_2) + d(v_1, v_2)) = e^{-\beta \bar{d}_M} \omega_2 \underline{d}_M > \bar{J}^N_{\max}$. The vehicle will arrive at target v_2 at time $b_2 = d(v_1, v_2) < \infty$. The target set $\{v_1, v_2\}$ then has a revisit bound of $\bar{b} = b_2 + d(v_2, v_1)$.

Inductive Step: Let N contains n targets which have a maximum revisit time bounded by $\bar{b}$. We will show that an additional target v_{n+1} will have a bounded revisit time $b_{n+1} < \infty$ and a decision time $\bar{t}_d(v_{n+1})$ given by (2).

The vehicle will decide to visit v_{n+1} when $J(v_i, v_{n+1}) \geq J(v_i, v_j)\ \forall i, j \in N$. Using (4) to bound the reward on $J(v_i, v_j)$ we desire,

$$\begin{aligned} J(v_i, v_{n+1}) &= e^{-\beta d(v_i, v_{n+1})} \omega_{n+1} (t(v_{n+1}) + d(v_i, v_{n+1})) \\ &\geq e^{-\beta \underline{d}_N} \bar{\omega}(\bar{b} + \bar{d}_N) = \bar{J}^N_{\max}, \end{aligned}$$

$\forall\ v_i \in N$. Solving for $t(v_{n+1})$ when $J(v_i, v_{n+1}) > \bar{J}^N_{\max}$ yields

$$t(v_{n+1}) \geq e^{-\beta(\underline{d}_N - d(v_i, v_{n+1}))} \frac{\bar{\omega}}{\omega_{n+1}} (\bar{b} + \bar{d}_N) - d(v_i, v_{n+1}).$$

Maximizing the right side of the inequality provides the decision time given in (2),

$$\bar{t}_d(v_{n+1}) \geq e^{-\beta(\underline{d}_N - \bar{d}_M)} \frac{\bar{\omega}}{\omega_{n+1}} (\bar{b} + \bar{d}_N) - \underline{d}_M.$$

When this inequality holds, the reward for visiting v_{n+1} exceeds the reward for visiting any other target and the optimal policy requires that the vehicle travels there next.

The decision time may occur when a vehicle is traveling between targets. Let $\tilde{d}(v_i)$ be the time distance left in the current route from $d(v_j, v_i)$ at time $\bar{t}_d(v_{n+1})$. Then the bound on the revisit time is $b_{n+1} = \bar{t}_d(v_{n+1}) + \tilde{d}(v_i) + d(v_i, v_{n+1}) < \bar{t}_d(v_{n+1}) + \bar{d}_N + \bar{d}_M < \infty$.

By mathematical induction all additional targets have finite revisit times with decision time dictated by (2). ■

Remark 1: The revisit bound $\bar{b}$ may be computed for a set of targets by beginning with two targets from the list and setting the group's revisit time bound to $2d(v_1, v_2)$ then recursively adding in new targets using bound $b_{n+1} = \bar{t}_d(v_{n+1}) + \bar{d}_N + \bar{d}_M$ with (2). The maximum bound using all permutations of the target list is $\bar{b}$ for the set. Note that, due to the need to find all permutations, computing the bound is only practical using this method when there are a limited number of targets.

We now extend this result to show that the revisit time of a target will also be bounded when executing a multi-step reward function with event horizon L.

Lemma 2: Let $N = \{v_1, \ldots, v_n\}$ contain n targets for which the maximum revisit time is bounded by $\bar{b}$. When a single vehicle

($V = 1$) is routed using the objective function defined by (1) with a multi-step look-ahead policy ($L > 1$) an additional target, v_{n+1}, will also have a bounded revisit time. Furthermore, a bound on the elapsed time prior to the vehicle deciding to visit target v_{n+1} is given by

$$\bar{t}_d^L(v_{n+1}) = \max\left\{ Le^{-\beta(\underline{d}_N - \bar{d}_M)} \frac{\bar{\omega}}{\omega_{n+1}} (\bar{b} + \bar{d}_N) - \underline{d}_M, 0 \right\}. \tag{5}$$

The revisit time for target v_{n+1} is bounded by $b_{n+1} = \bar{t}_d^L(v_{n+1}) + \bar{d}_N + \bar{d}_M < \infty$.

Proof: When $V = 1$ and $L > 1$ (1) becomes

$$\begin{aligned} J(\pi_1) = {} & e^{-\beta T_1^{\pi_1}} \omega_{s_1[1]} \left(t(s_1[1]) + T_1^{\pi_1}\right) \\ & + e^{-\beta T_2^{\pi_1}} \omega_{s_2[1]} \left(t(s_2[1]) + T_2^{\pi_1}\right) + \\ & \vdots \\ & + e^{-\beta T_L^{\pi_1}} \omega_{s_L[1]} \left(t(s_L[1]) + T_L^{\pi_1}\right), \end{aligned} \tag{6}$$

where $\pi_1 = \{s_1, s_2, \ldots, s_L\}$ is the policy for the vehicle. Since the time $T_k^{\pi_1}$ strictly increases at every step, the exponential discount factor is strictly decreasing and the least penalized reward is provided by the first term in (6). Therefore, the maximum reward for visiting targets in N is bounded by $\bar{J}_{\max}^{N,L} < L\bar{J}_{\max}^{N}$, where $\bar{J}_{\max}^{N}$ is defined in (4).

Base Case: Assuming $N = \{v_1\}$, the vehicle's policy π_1 when operating over N consists of L identical states $s_1 = \cdots = s_L = (v_1, 0)$. The revisit bound and rewards are $\bar{b} = 0$ and $J(\pi_1) = \bar{J}_{\max}^{N,L} = 0$.

Given a second target v_2, the policy which will maximize (6) is $\pi_1 = \{s_1, \ldots, s_L\}$ where $s_k[1]$ is v_2 if k is odd and v_1 if k is even $\forall\ k \in [1, L]$. The time when the vehicle decides to travel to v_2 is $\bar{t}_d^L = 0$, resulting in upper bound of $b_2 = d(v_1, v_2) = \bar{d}_M$. The combined target group $\{v_1, v_2\}$ has a revisit bound of $\bar{b} = b_2 + d(v_1, v_2) = 2\bar{d}_M < \infty$.

Inductive Step: Assume the set $N = \{v_1, \ldots, v_n\}$ contains n targets for which the maximum revisit time is bounded by $\bar{b}$. We will show that an additional target, v_{n+1}, will have a bounded revisit time and an upper bound of $\bar{t}_d^L(v_{n+1})$ on its decision time.

Since targets visited early in a policy are discounted less, the policy that maximizes the reward $\max_{\pi_1}\{J(\pi_1)\}$ must place the first target to be visited as the one with the highest one-step reward. We will evaluate the time at which the one-step reward for visiting v_{n+1} will exceed the reward for visiting any L targets in N.

Let a one-step reward policy be $\pi_1 = \{v_i, v_{n+1}\}$ for any $v_i \in N$. Using (3) and the reward bound for an L-step look-ahead policy we desire $J(\pi_1) > \bar{J}_{\max}^{N,L}$,

$$\begin{aligned} J(\pi_1) &= e^{-\beta d(v_i, v_{n+1})} \omega_{n+1}(t(v_{n+1}) + d(v_i, v_{n+1})) \\ &> Le^{-\beta \underline{d}_N} \bar{\omega}(\bar{b} + \bar{d}_N) = \bar{J}_{\max}^{N,L}. \end{aligned}$$

Solving for $t(v_{n+1})$ when this inequality holds yields

$$t(v_{n+1}) \geq Le^{-\beta(\underline{d}_N - d(v_i, v_{n+1}))} \frac{\bar{\omega}}{\omega_{n+1}} (\bar{b} + \bar{d}_N) - d(v_i, v_{n+1}).$$

Maximizing the right side of the inequality provides the decision time given in (5),

$$\bar{t}_d^L(v_{n+1}) \geq L\left(e^{-\beta(\underline{d}_N - \bar{d}_M)} \frac{\bar{\omega}}{\omega_{n+1}} (\bar{b} + \bar{d}_N) \right) - \underline{d}_M.$$

At this time the reward for visiting v_{n+1} exceeds the reward for visiting any other target. Equation (6) is then maximized when target v_{n+1} is placed as the first target visited. Therefore, at or before time $\bar{t}_d^L$ the vehicle will decide to visit v_{n+1} next, ensuring that the revisit time will be bounded by $b_{n+1} = \bar{t}_d^L(v_{n+1}) + \bar{d}_N + \bar{d}_M < \infty$.

By mathematical induction all additional targets have finite revisit times with decision time dictated by (5). This theorem holds for a multi-step look-ahead policy with an arbitrarily large, but finite event horizon. ■

IV. Multiple Vehicles

We now consider the case with multiple vehicles, $A \geq 2$. An exhaustive search to find the optimal solution to the reward function (1)) is computationally intractable as the complexity grows exponentially in the number of vehicles and the time horizon. BCA/MCTS provides a scalable solution to the multiple-vehicle routing problem. BCA enables separate optimization of each vehicle's route and facilitates the decentralized implementation. MCTS provides an intelligent tree search method that is used to efficiently find high reward vehicle routes without needing to exhaustively search the decision tree. In the following subsections we describe both components of the algorithm and show that BCA is guaranteed to converge to a block optimal solution.

A. Blockwise Coordinate Ascent

Block coordinate ascent algorithms are non-derivative optimization approaches where, for each iteration, the reward is maximized over a block of coordinates while keeping the others fixed [8]. For this application, each vehicle's policy is grouped as the optimization block while the other vehicle's policies are kept fixed.

To optimize a single vehicle's route we use a local reward function for vehicle a defined as

$$\begin{aligned} f_a(\boldsymbol{\pi}) = {} & \underbrace{\sum_{k=1}^{L} e^{-\beta T_k^{\pi_a}} \omega_k \left(t(s_k) + T_k^{\pi_a}\right)}_{\text{variable reward}(\pi_a)} \\ & + \underbrace{\sum_{b \in V \setminus a} \sum_{k=1}^{L} e^{-\beta T_k^{\pi_b}} \omega_k \left(t(s_k) + T_k^{\pi_b}\right)}_{\text{conditional reward}(\boldsymbol{\pi}_{a^C})}. \end{aligned} \tag{7}$$

The vehicle's maximize $f_a(\boldsymbol{\pi})$ by choosing a policy π_a^* conditioned on the policies of its peer vehicles $\boldsymbol{\pi}_{a^C}$. The peer vehicle's policies are the latest policies shared with vehicle a. If communication is interrupted or delayed, the vehicle assumes the peer-vehicle policies are unchanged. As with the global reward function, the reward for visiting a target depends upon the

Algorithm 1: Blockwise Coordinate Ascent (BCA).

Input : target state information (IDs, locations, weights), objective function, and event horizon

Output: $\hat{\boldsymbol{\pi}}$: policies of all vehicles

iteration $p = 0$;
while *converging* **do**
  for $a \leftarrow 1$ **to** A vehicle **do**
    1) Solve $\pi_a^* = \arg\max_{\pi_a} \{f_a(\hat{\boldsymbol{\pi}}_a^p)\}$, let $\hat{\boldsymbol{\pi}}_a^p$ be the policies for all vehicles known to vehicle a at iteration p.
    2) Set $\hat{\boldsymbol{\pi}}_a^{p+1} = (\pi_a^*, \hat{\boldsymbol{\pi}}_{a^C}^p)$
    3) Communicate π_a^* with neighbors
    4) Vehicles in communication range update their policies, $\hat{\boldsymbol{\pi}}_{a^C}^{p+1} = \text{latest}(\hat{\boldsymbol{\pi}}_a^{p+1})$
    5) $p = p + 1$, update the iteration count
  end
end

time it was last seen. Therefore, the summations are evaluated in target-arrival chronological order.

Algorithm 1 presents pseudo-code for the BCA algorithm where each vehicle iteratively finds the policy π_a^* that will maximize it's local reward function (7) using $\hat{\boldsymbol{\pi}}_a^p$, the most up-to-date policies known to vehicle a at iteration p. If reliable communication is available then $\hat{\boldsymbol{\pi}}_a^p$ is identical for all vehicles and consists of each vehicle's locally optimal policy. Convergence is reached when all vehicle policies, $\pi_a\ \forall a \in A$, remain unchanged for a full round (loop through all the vehicles). In practice the algorithm may be cut off after a set number of iterations or once a time limit has been reached.

Note that at every iteration p, the reward function is non-decreasing, i.e. $J(\hat{\boldsymbol{\pi}}^{p+1}) \geq J(\hat{\boldsymbol{\pi}}^p)$. This is because (7) is a subproblem of (1) and a new policy π_a will only be chosen when the reward for $f_a(\hat{\boldsymbol{\pi}}_a^{p+1})$, and thus $J(\hat{\boldsymbol{\pi}}^{p+1})$, is increased.

In the following theorem we show that Algorithm 1 will converge to a block optimal solution. Letting $\tilde{\pi}_a$ represent all possible policies for vehicle a, the solution $\hat{\boldsymbol{\pi}}$ is block optimal if $J(\hat{\boldsymbol{\pi}}) \geq f_a(\tilde{\boldsymbol{\pi}}_a)$ for all $a \in V$ where $\tilde{\boldsymbol{\pi}}_a = (\tilde{\pi}_a, \hat{\boldsymbol{\pi}}_{a^C})$. This definition is satisfied if the policies become constant, meaning a full round of the BCA algorithm is completed without any vehicle updating its locally optimal policy π_a^*.

In the theorem, we utilize ([15], Lemma 6) which guarantees that the sequence of local objective functions will become constant when using a discrete set of bounded input values. This occurs because a finite and bounded set of input values correspond with a finite set of output values. Since the local objective function is nondecreasing, the output values must converge. This is the block optimal solution and does not necessarily correspond with the global optima.

Theorem 1: Given all-to-all communication, A vehicles executing Algorithm 1 will converge to a block optimal policy $\boldsymbol{\pi}^*$. Furthermore, under a constant policy, all targets have finite revisit rates.

Proof: The target ID's and positions provides a finite and bounded set of discrete input values with inter-target distances that are deterministic and bounded. Bounded input values result in bounded outputs for both local $f_a(\hat{\boldsymbol{\pi}}_a^p)$ and global $J(\hat{\boldsymbol{\pi}}^p)$ reward functions. With full communication, each vehicle uses their peer's current policies to update their own policy which ensures that both $f_a(\hat{\boldsymbol{\pi}}_a^p)$ and $J(\hat{\boldsymbol{\pi}}^p)$ weakly increase at every iteration. According to ([15], Lemma 6), the objective functions will weakly increase until they become constant.

By Lemma 2, when a target's elapsed time grows sufficiently large it becomes the first target in the receding horizon control. Once a target is queued to be visited, there are no other targets that can be chosen to give that vehicle a higher reward. Other vehicles may decide to visit that target, but will also choose it as its first target if it increases the local and global reward values. Therefore, once a target becomes the highest one-step reward it will be included as the next target visited for one of the vehicles in any locally optimal policy $\hat{\boldsymbol{\pi}}$. ■

B. Monte Carlo Tree Search

Despite the computational advantages of BCA, solving each subproblem $f_a(\boldsymbol{\pi})\ \forall a \in V$ will frequently be too costly to execute in real-time. To overcome this challenge, we propose using Monte Carlo Tree Search (MCTS) to provide a solution to the decision tree. Because MCTS is an "anytime" solution, we can terminate the search at any point knowing we will have the best current solution.

MCTS searches a tree by selecting new nodes based upon a balance of exploring new branches and expanding high reward branches. The tree is expanded to incorporate the children nodes while tracking a mean reward over paths evaluated under the parent. Given enough samples, MCTS is guaranteed to converge to the subproblem's optimal solution.

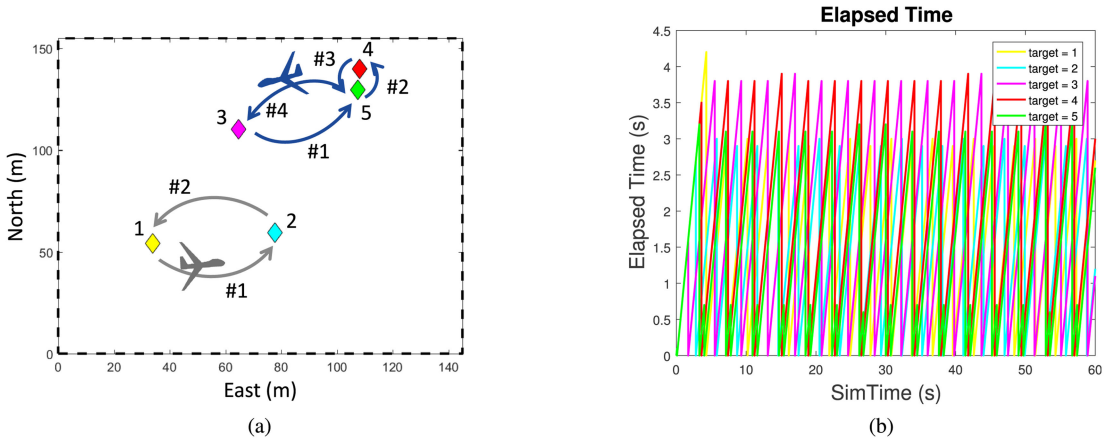
Let the tree for vehicle a be denoted $T_a = (V_a, E_a)$ where V_a are the nodes and contain the total states of the system. The edges E_a encode the transitions. Each node $v \in T_a$ contains a unique sequence of node-edge pairs connecting it to the root node k (or current target). This sequence of decisions is a valid route or policy for the vehicle and the length of this policy is limited by the event horizon length L. The tree encapsulates the exhaustive set of valid policies for the vehicle. Given χ targets, each node has $\chi - 1$ children nodes, describing the potential targets the vehicle may choose to visit next.

The reward for each node, $\bar{f}_a(v)$, is the average reward of all explored control policies containing node v. Nodes are added to the search tree in a cyclic manner following the four general steps: selection, expansion, simulation, and backpropagation.

1) Selection: Node selection maximizes an upper confidence bound for trees: $UCT(p, c) = \bar{f}_a(c) + \kappa_p \sqrt{\ln q(p)/q(c)}$, [16] where p is the parent node, c are children nodes, q returns the number of times a node has been selected while exploring the control policies, and κ_p is a tuning parameter to weight exploration. The UCT metric is iteratively applied at every level of the tree, until an unexplored node is reached.

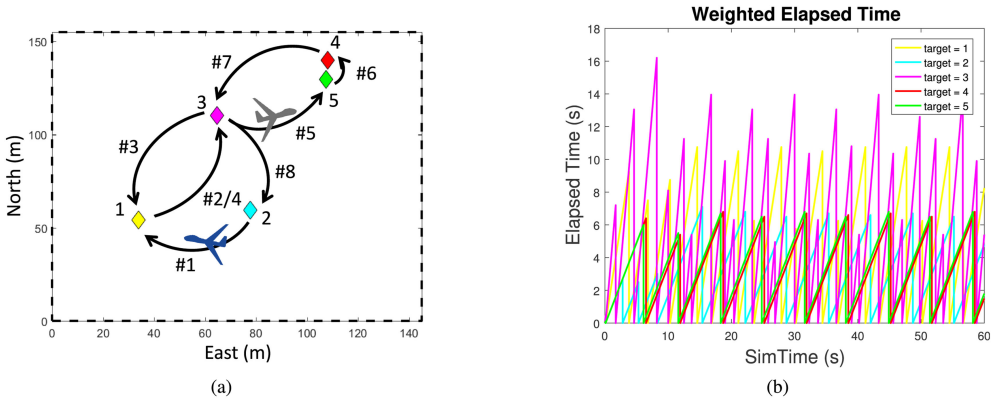
2) Expansion: Chosen, unexplored nodes are expanded to identify it's $(\chi - 1)$ children. Child nodes are initialized with an infinite mean reward $\bar{f}_a(c) = \infty$, ensuring that they will be explored at least once prior to sibling nodes being re-selected.

3) Simulation: Simulation is the process of expanding the route until the event horizon is reached. We use a greedy heuristic



(a) (b)

Fig. 1. The (a) steady-state routes and (b) elapsed times for two UAVs visiting five equally weighted targets.



(a) (b)

Fig. 2. The (a) steady-state route and (b) weighted elapsed times for two UAVs visiting five unequally weighted targets.

policy to complete the control sequence by choosing the next target with the highest instantaneous reward. Once a complete control sequence is chosen, the local reward $f_a(\boldsymbol{\pi})$ is calculated by combining this simulated policy with the policies of other vehicles and solving (7).

4) Backpropagation: Once a new local reward is computed, each node along that control sequence $v \in \pi$ updates their average local reward $\bar{f}_a(v)$ by averaging (or backpropagating) in this new reward.

The process of selecting, expanding, simulating, and backpropagating is repeated until a stopping criterion is reached, such as a time limit or specific number of samples.

V. Results

Numerical simulations are now provided to illustrate the efficacy of the decentralized BCA/MCTS algorithm. First, we present simulations comparing solutions for equally and unequally weighted targets. Then, using Monte Carlo (MC) simulations, we compare the BCA/MCTS algorithm to a centralized algorithm which exhaustively searches a joint decision-tree for all vehicles. The joint optimization is computationally expensive and necessitated restricting the number of UAVs, targets, and look-ahead steps to run MC simulations. Results are also provided for scenarios with varying numbers of vehicles and vehicles where communication is unreliable.

Fig. 1 illustrates the BCA/MCTS solution when all targets are equally weighted. The target positions were chosen randomly and the two UAVs quickly settled into the cyclic patterns shown in Fig. 1(a). Fig. 1 presents the elapsed time since that each target was visited. The revisit times for all targets are nearly equal with a maximum time of 3.9 s.

Fig. 2 compares the result when the targets have weights $\boldsymbol{\omega} = [2.5, 1, 4.5, 1, 1]^T$. All other initial conditions were identical to those used in Fig. 1. In this example, the vehicles cooperatively visit the higher weighted targets. They settle into a single cyclic pattern, as shown in Fig. 2(a), where the two vehicles separate themselves half way through the cycle. There is a wider divergence in the target's weighted elapsed time as shown in Fig. 2(b). However, Target 3, which has the highest revisit time, always has a vehicles travelling to it. Given the target configuration, this cycle minimizes the max revisit times.

Fig. 3 compares BCA/MCTS to a centralized solution which exhaustively searches the joint action space of all vehicles. BCA/MCTS uses different sample strategies that limit its search to 25%, 50%, 75%, and 100% of each vehicle's individual routing tree. One hundred MC simulations were run using two vehicles, $L = 2$, with $\beta = 0.1$ and a simulation

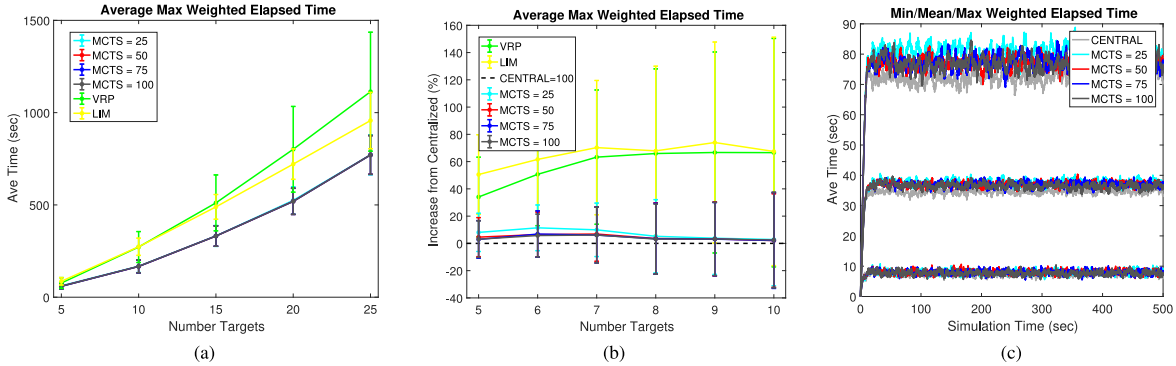


Fig. 3. Monte Carlo simulations showing the average maximum weighted target elapsed times of BCA/MCTS. (a) Elapsed times for target sets $\chi = \{5,\ 10,\ 15,\ 20,\ 25\}$. (b) Elapsed times as a percentage increase from the centralized solution. (c) The average minimum, mean, and maximum target revisit time for six targets.

TABLE I
MEAN RUNTIME AND STANDARD DEVIATION FOR A VEHICLE'S DECISION CYCLE AVERAGED OVER 100 MC RUNS

	5 Targets	10 Targets	15 Targets	20 Targets	25 Targets
MCTS (25%)	10.0±1.5 (s)	22.8±2.9 (s)	38.6±4.4 (s)	58.6±4.9 (s)	82.9±6.2 (s)
MCTS (50%)	14.3±2.3 (s)	37.4±4.8 (s)	69.5±7.3 (s)	110.1±9.0 (s)	161.0±11.2 (s)
MCTS (75%)	18.1±3.0 (s)	52.4±7.0 (s)	100.3±10.4 (s)	163.4±13.6 (s)	244.7±17.2 (s)
MCTS (100%)	22.7±3.8 (s)	69.3±9.25 (s)	131.9±13.6 (s)	219.0±18.4 (s)	333.7±23.7 (s)
Central	58.97±6.9 (s)	686.8±86 (s)	$6.9\times10^3\pm986$ (s)	$6.2\times10^4\pm9.9\times10^3$ (s)	—

time of 500 s. Fig. 3(a) shows results given targets in $\chi = \{5,\ 10,\ 15,\ 20,\ 25\}$. The error bars provide the 1-σ bounds. The results show there is nearly identical performance between each MCTS sampling strategy.

Fig. 3(b) compares the percentage increase of BCA/MCTS from the centralized solution. The BCA/MCTS solutions typically remained within 7% of the centralized algorithm (the 25% sampling strategy, cyan line, being the only exception). The highest divergence from the centralized solution was given with six targets. For this case, the minimum, mean, and maximum average revisit times are provided for the averaged MC runs in Fig. 3(c). Both these plots show that BCA/MCTS solutions are close to those of the centralized solution.

Fig. 3(a) and Fig. 3(b) also provide a comparison for two other routing methods. The first (yellow line) depicts a limited communication method implemented in [7]. In this work a heuristic solution to (3) is presented where the cooperating vehicles broadcast a message once a target visit is complete. Coordination between vehicles does not occur at the planning stage. Both figures show that the coordinated planning presented in this paper provides a significant advantage in reducing the maximum revisit times of targets.

The second comparison shows a multi-vehicle traveling salesman problem solution (green line) implemented using Google's ORTools [17]. This solution found the closed-walk paths that minimized the distance travelled to all the targets. The result illustrates the need for allowing weighted targets to be visited more than once per tour since there can be as much as a 67% increase in the average maximum weighted revisit times when compared to the centralized solution.

Table I shows the runtime advantage in using the BCA/MCTS algorithms for the MC runs depicted by Fig. 3(a). The centralized joint-vehicle tree shows an exponential increase as the number of targets grows. Whereas BCA/MCTS shows more amenable runtime values that scale with the percentage of the tree that is searched.

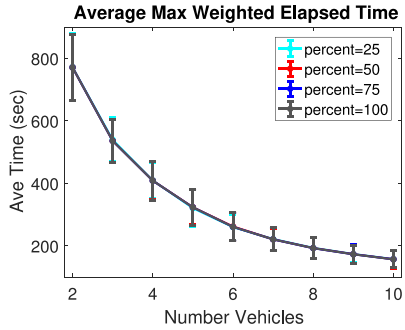


Fig. 4. Average weighted elapsed time as a function of the number of vehicles.

Fig. 4 shows the average maximum weighted elapsed time for 25 targets with varying numbers of vehicles. This figure again shows that exploring only a percentage of the decision space doesn't strongly diminish the results. Additionally, this graph shows the advantage of increasing the number of vehicles in the operational space by the reduction in average revisit times. However, as expected there are diminishing returns with each additional vehicle. Table II shows the effect of increasing the number of vehicles in the algorithm's runtime. Note that all runtime values were obtained using a Matlab implementation of the algorithm and will be greatly improved with a C++ implementation (an item of future work). Fig. 4 and Table II indicate that a reduction in the percentage of the tree searched will not

TABLE II
MEAN RUNTIME AND STANDARD DEVIATION FOR A VEHICLE'S DECISION CYCLE GIVEN 25 TARGETS AVERAGED OVER 100 MC RUNS

	4 Vehicles	6 Vehicles	8 Vehicles	10 Vehicles
MCTS (25%)	208.2±16.2 (s)	382.0±31.1 (s)	584.5±42.6 (s)	808.2±56.4 (s)
MCTS (50%)	373.4±30.2 (s)	650.6±52.2 (s)	957.4±72.1 (s)	1281.9±92.9 (s)
MCTS (75%)	545.1±46.0 (s)	926.9±79.7 (s)	1334.0±105.7 (s)	1756.1±128.8 (s)
MCTS (100%)	740.6±59.5 (s)	1244.1±102.5 (s)	1771.1±128.5 (s)	2299.7±155.2 (s)

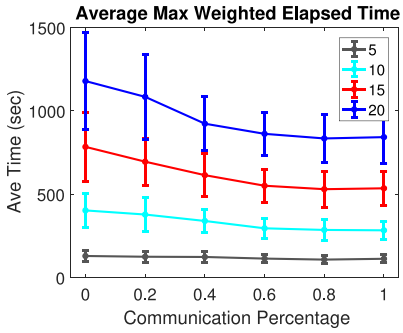


Fig. 5. The average elapsed times when the communication radius is varied from no (0%) to full (100%) communication.

significantly affect the results, but will result in a significant reduction in the runtime.

Fig. 5 shows that BCA/MCTS degrades gracefully under intermittent communication. Four target sets, $\chi = \{5, 10, 15, 20\}$, are used while the maximum communication distance was varied from 0% (no communication) of the operational area to 100% (constant communication). This simulation used two vehicles in order to depict the extreme effects of communication. Adding additional vehicles would provide similar trends provided the vehicles do not saturation the operational space or target assignments. Vehicles outside of communication range updated their routes using the most recent policy of the peer vehicles. Vehicles within communication range shared their updated policies. This figure shows a graceful decay in the vehicle's performance under limited communication ranges for each target set. Improvements for additional communication become more pronounced with larger targets numbers and taper off when communication is available for ~60% of the operational area.

Intermittent communication may result in multiple vehicles visiting the same targets. However, it won't cause target revisit times to go unbounded since targets with large elapsed times (defined by (5)) will be placed first in the vehicle's policy. Once in that position, it cannot be removed without confirmation that another vehicle has placed it first in their policy. The target will then be visited by at least one vehicle.

VI. CONCLUSION

This paper presented a decentralized BCA/MCTS algorithm that enables persistent monitoring of targets using cooperating UAVs. The vehicles coordinated their routes by optimizing their own policies, conditioned upon the known policies of their peers. Guarantees were provided that the BCA/MCTS algorithm will converge to a block optimal solution, giving all targets finite revisit rates. When limitations are placed on the algorithm, such as computational or communication constraints it was shown that the performance degrades gracefully and typically remains close to the ideal.

REFERENCES

[1] D. Van der Merwe and K. P. Price, "Harmful algal bloom characterization at ultra-high spatial and temporal resolution using small unmanned aircraft systems," *Toxins*, vol. 7, no. 4, pp. 1065–1078, 2015.
[2] S. G. Manyam, S. Rasmussen, D. W. Casbeer, K. Kalyanam, and S. Manickam, "Multi-UAV routing for persistent intelligence surveillance & reconnaissance missions," in *Proc. Int. Conf. Unmanned Aircr. Syst.*, 2017, pp. 573–580.
[3] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*, vol. 174, San Francisco, CA, USA: Freeman, 1979.
[4] S. K. K. Hari, S. Rathinam, S. Darbha, K. Kalyanam, S. G. Manyam, and D. Casbeer, "Efficient computation of optimal uav routes for persistent monitoring of targets," in *Proc. Int. Conf. Unmanned Aircr. Syst.*, 2019, pp. 605–614.
[5] K. Kalyanam, M. Pachter, and D. Casbeer, "Average reward dynamic programming applied to a persistent visitation and data delivery problem," in *Proc. Dyn. Syst. Control Conf.*, 2017, pp. 1–7.
[6] A. L. Von Moll, D. W. Casbeer, K. Kalyanam, and S. G. Manyam, "Genetic algorithm approach for UAV persistent visitation problem," in *Proc. Dyn. Syst. Control Conf.*, Sep. 2018, pp. 1–10.
[7] S. Rasmussen, K. Kalyanam, S. Manyam, D. Casbeer, and C. Olsen, "Practical considerations for implementing an autonomous, persistent, intelligence, surveillance, and reconnaissance system," in *Proc. 1st Annu. IEEE Conf. Control Technol. Appl.*, Aug. 2017, pp. 1847–1854.
[8] S. J. Wright, "Coordinate descent algorithms," *Math. Program.*, vol. 151, no. 1, pp. 3–34, 2015.
[9] P. Tseng and S. Yun, "A coordinate gradient descent method for nonsmooth separable minimization," *Math. Program.*, vol. 117, no. 1-2, pp. 387–423, 2009.
[10] C. B. Browne*et al.*, "A survey of monte carlo tree search methods," *IEEE Trans. Comput. Intell. AI Games*, vol. 4, no. 1, pp. 1–43, Mar. 2012.
[11] G. Best, O. M. Cliff, T. Patten, R. R. Mettu, and R. Fitch, "Dec-MCTS : Decentralized planning for multi-robot active perception," *Int. J. Robot. Res.*, vol. 38, no. 2-3, pp. 316–337, 2019.
[12] M. Schneider, "Receding-horizon planning using recursive monte carlo tree search with sparse action sampling for continuous state and action spaces," in *Proc. Amer. Control Conf.*, Jul. 2016, pp. 5401–5406.
[13] C. A. B. Baker, S. Ramchurn, W. T. L. Teacy, and N. R. Jennings, "Factored monte-carlo tree search for coordinating uavs in disaster response," in *Proc. ICAPS Proc. 4th Workshop Distrib. Multi-Agent Planning*, 2016, pp. 6–15.
[14] C. C. Olsen, K. Kalyanam, W. P. Baker, and D. L. Kunz, "Maximal distance discounted & weighted revisit period: A utility approach to persistent unmanned surveillance," *Unmanned Syst.*, vol. 7, no. 04, pp. 215–232, 2019.
[15] S. Jäger and A. Schöbel, "The blockwise coordinate descent method for integer programs," *Math. Methods Oper. Res.*, vol. 91, pp. 357–381, 2020.
[16] L. Kocsis and C. Szepesvári, "Bandit based monte-carlo planning," in *Mach. Learn.: ECML 2006*, J. Fürnkranz, T. Scheffer, and M. Spiliopoulou, Eds. Berlin, Germany: Springer, 2006, pp. 282–293.
[17] L. Perron and V. Furnon, *Operations Research Tools 7.2*. Google. Accessed: Jul. 19, 2019. [Online]. Available: https://developers.google.com/optimization/