2016 American Control Conference (ACC)
Boston Marriott Copley Place
July 6-8, 2016. Boston, MA, USA

Spatial Load Balancing in Non-Convex Environments using Sampling-Based Motion Planners

Beth Boardman[1,2], Troy Harden[2], and Sonia Martínez[1]

Abstract— **This paper proposes an algorithm to approximately solve a spatial-load balancing problem for agents, subject to differential constraints, deployed in non-convex environments. A probabilistic roadmap is used to approximate regions via connected sets of vertices, which describe agents' configurations and optimal paths joining them. At each iteration, agents' positions and assigned graph nodes are updated to minimize the cost function. Two graph-node partitions are considered. In the first one, $\widetilde{\mathcal{V}}$, all graph vertices are allocated to one agent or another. The second one, $\widetilde{\mathcal{V}}^{\text{lower}}$, is a lower approximation that only allocates some of the graph vertices to the agents and has the advantage of requiring less communication than required for $\widetilde{\mathcal{V}}$. Algorithm convergence can be guaranteed for $\widetilde{\mathcal{V}}$ to a neighborhood of the continuous-space counterpart, and to its solution as sampling dispersion tends to zero. The convergence of the algorithm using $\widetilde{\mathcal{V}}^{\text{lower}}$ and trade-offs between $\widetilde{\mathcal{V}}^{\text{lower}}$ and $\widetilde{\mathcal{V}}$ are established in simulation for a Euclidean metric case and Dubins' vehicle dynamics.**

I. Introduction

This paper presents a scalable solution to the area-constrained multi-agent spatial load balancing problem in non-convex environments and for agents subject to differential constraints. The solution involves a group of agents dividing the load of acquiring information or of servicing tasks spread over an area in a fair way. Efficient task assignment, which should respect the cost associated with motion, requires accounting for the agents' differential constraints. Environment non-convexities and differential constraints make the description of the coverage regions challenging, therefore an approximate solution is obtained that reduces the communication required among agents.

The gradient-based descent Lloyd algorithm, [1], is the basis of many multi-agent coverage strategies. A limited sample of work building on this approach includes limited sensor footprints [2], heterogeneous agents (different sensing radii) [3], [4], non-holonomic agents [5], [6], [7], and power constraints [8]. The area-constrained problem is studied in [9], [10], [11], where weights added to the cost functional force the partition to have the desired area.

Closely related to this paper is the research on multi-agent coverage in non-convex environments. In [12] a diffeomorphism is used to transform the non-convex environment into an almost convex one, where only a finite number of points have been subtracted. The coverage problem is then solved in the transformed environment and a solution is obtained via the inverse transformation. The diffeomorphism limits this algorithm to two-dimensional environments. Environments with polygon obstacles are considered in [13], [14]. A solution to the multi-agent coverage problem for non-point robots in non-convex environments can be found in [4]. A Voronoi partition and potential field is used in [15] to find a solution to the coverage problem in non-convex environments with unknown obstacles. The authors of [16], [17], [18] build a gridded environment map and then determine which grid points belong to each agents' generalized Voronoi cells. This grid-based approach is limited to low dimensional spaces and differential constraints are difficult to handle in this manner.

The algorithm presented in this paper builds a probabilistic roadmap star (PRM*) graph, G, to recover the cost of an agent subject to differential constraints moving between any two configurations in a known non-convex environment. The nodes of G are assigned to the agents by means of a partition $\widetilde{\mathcal{V}}$, then the nodes are employed to estimate the area covered by each agent. At each iteration each agent's subset of nodes is updated; then agents move to a neighboring node in the graph with the lowest cost functional value. In order to keep the algorithm's computations as scalable as possible, a lower approximation of the cost (e.g. one which only partially considers obstacles) is exploited, which leads to an assignment of nodes, $\widetilde{\mathcal{V}}^{\text{lower}}$, corresponding to a subset of the total number of nodes and for which the area constraint is further relaxed. Algorithm convergence can be guaranteed for $\widetilde{\mathcal{V}}$ to a neighborhood of the continuous-space counterpart, and to its solution as sampling dispersion tends to zero. The convergence of the algorithm using $\widetilde{\mathcal{V}}^{\text{lower}}$ and $\widetilde{\mathcal{V}}$ to specific configurations is shown in simulation for a Euclidean metric case and Dubins' vehicle dynamics. A description is given on how to recover $\widetilde{\mathcal{V}}$ using the lower approximation which establishes trade-offs between the use of $\widetilde{\mathcal{V}}^{\text{lower}}$ and $\widetilde{\mathcal{V}}$ in reaching an acceptable solution of the problem.

II. Preliminaries

Let $X \subseteq \mathbb{R}^d$ be the d-dimensional configuration space of an agent and the obstacle space be X_{obs}. The free-space, $Q = X \setminus X_{\text{obs}}$, is the set of all collision free agent configurations. In general, Q is non-convex and simply connected by dynamic paths. A partition of Q, $\mathcal{W} = \{W_1, \ldots, W_n\}$, is a collection of n regions, $W_i \subset Q$, $i \in \{1, \ldots, n\}$, whose interiors are disjoint and union covers Q, $\cup_{i=1}^{n} W_i = Q$.

The spatial load balancing problem aims to find the optimal locations for n agents, $P = \{p_1, \ldots, p_n\}$ ($p_i \in Q$), and weights, $\omega = \{\omega_1, \ldots, \omega_n\}$ ($\omega_i \in \mathbb{R}_{>0}$). Let the optimal cost to move from configuration $q_1 \in Q$ to $q_2 \in Q$ when

[1] Mechanical and Aerospace Engineering, University of California, San Diego, 9500 Gilman Dr, La Jolla, Ca 92093-0411 {bboardman, soniamd}@ucsd.edu

[2] Los Alamos National Laboratory, PO Box 1663, MS J580, Los Alamos, NM 87545 harden@lanl.gov

978-1-4673-8682-1/$31.00 ©2016 AACC

subject to the differential constraint, $\dot{p}_i = f(p_i, u_i)$ with control input u_i, be $J(q_1, q_2) \geq 0$. A probability density function, $\phi(q)$, defined over Q, $\phi : Q \to \mathbb{R}_{\geq 0}$, describes the likelihood of an event occurring at a configuration in Q.

A. Problem Statement

Given $a_1, \ldots, a_n \in \mathbb{R}_{>0}$, such that $\sum_{i=1}^{n} a_i = \int_Q \phi(q)dq$, the n agents solve the following problem subject to the area and dynamic constraints,

Problem 1.

$$\begin{aligned} \min \quad & \mathcal{H}(P, \mathcal{W}) = \sum_{i=1}^{n} \int_{W_i} J(p_i, q)\phi(q)dq \\ \text{s.t.} \quad & p_i \in Q, \qquad i \in \{1, \ldots, n\} \\ & \dot{p}_i = f(p_i, u_i) \\ & a_i = \int_{W_i} \phi(q)dq. \end{aligned}$$

For convex environments, the results in [9] state that, given a set of positions, P, there exists a weight assignment, ω, that makes a generalized additively-weighted Voronoi partition, $\mathcal{V}(P, \omega; J)$, feasible and that this Voronoi partition is the best among the partitions $\mathcal{W}$ that satisfy the area constraints. Here, $\mathcal{V}(P, \omega; J) = \{V_1, ..., V_n\}$ is defined as

$$V_i = \{q \in Q \mid J(p_i, q) - \omega_i \leq J(p_j, q) - \omega_j, \ \ \forall j \neq i\}.$$

It can be shown that the above results do not change for non-convex environments by using the general Leibniz rule,

$$\begin{aligned} \frac{\partial}{\partial \omega_i} \sum_{j=1}^{n} \int_{V_j(\omega)} (J(p_j, q) - \omega_j)\phi(q)dq = & - \int_{V_i(\omega)} \phi(q)dq, \\ + \sum_{j=1}^{n} \int_{\partial V_j(\omega)} & \frac{\partial q}{\partial \omega_i} n_j((J(p_j, q) - \omega_j)\phi(q))dq, \end{aligned}$$

for differentiation over any set [19]. Specifically, the Leibniz rule is used in [9] to prove the weight-to-areas map is gradient, which is used to prove that there exist weights that allow the generalized Voronoi partition satisfy the constraints.

In convex environments, given a fixed partition, the best agent positions are the centroids of their coverage regions However, in non-convex environments more than one agent position may minimize the cost function. These minimizers are generalized centroids, P^*, and $(P^*, \mathcal{V}(P^*, \omega^*; J))$ denotes a solution that satisfies the above results. Note that if for some agent i there exists an agent j such that the weights satisfy $\omega_j - \omega_i > J(p_i, p_j)$, then $V_i = \emptyset$. By using an optimal probabilistic roadmap of Q, this paper recovers an approximate solution to the above Problem 1.

B. Optimal Probabilistic Roadmap Building

This section briefly describes how to construct an asymptotically optimal probabilistic roadmap (PRM*), denoted as G, which allows the agents to recover the approximately optimal path cost between two configurations in the graph as the sum of edge costs. This assumes a known environment. The details of the construction can be found in [20]. The graph G is composed of a set of nodes $\mathcal{N}_G$ and a set of edges $\mathcal{E}_G$. Each node $q \in \mathcal{N}_G$ is a configuration from Q. Each edge, $e \in \mathcal{E}_G$, is an ordered pair, $e = (q_1, q_2)$, which corresponds to an optimal path in Q, satisfies the differential constraints, is collision free, and has a cost $J_e(q_1, q_2)$. The cost $J(q_1, q_2)$ is assumed to be additive; given an optimal path from q_1 to q_2, and a node q' in that path, it holds that, $J(q_1, q_2) = J(q_1, q') + J(q', q_2)$. The out neighbors of q are $\mathcal{N}_G^{\text{out}}(q) = \{q_j \in \mathcal{N}_G \mid (q, q_j) \in \mathcal{E}_G\}$.

Each iteration of the construction of G begins by taking a uniformly sampled random configuration from the free-space, $q_{\text{rand}} \in Q$. Next, all graph vertices that are within a ball centered at q_{rand} with radius, $r = \gamma_G(\log(m)/m)^{1/d}$, are determined and denoted Q_{near}. Here, γ_G is a fixed parameter, m is the number of vertices currently in $\mathcal{N}_G$, and d is the dimension of Q. If Q_{near} is empty then the graph vertex that is closest to q_{rand} is added to Q_{near}. The least-cost paths from q_{rand} to $q_{\text{near}} \in Q_{\text{near}}$ are determined as outgoing edges of q_{rand}. If the direction matters, as with differential constraints, the least-cost path from q_{near} to q_{rand} is determined as incoming edges of q_{rand}. Each collision-free path is added to $\mathcal{E}_G$. The application of spatial load balancing requires that G be strongly connected; a necessary condition is that all $q \in \mathcal{N}_G$ that do not have both an outgoing and incoming edge be removed. A sufficient condition is to only allow edge pairs, $(q_1, q_2) \in \mathcal{E}_G$ if and only if $(q_2, q_1) \in \mathcal{E}_G$.

The free-space Q is discretized by G while maintaining an asymptotically optimal roadmap of the environment. Each $q \in \mathcal{N}_G$ has an area, $\alpha(q)$, which is calculated as follows. Let $\mathcal{N}_X = \mathcal{N}_G \cup \mathcal{N}_{\text{obs}}$, where $\mathcal{N}_{\text{obs}}$ is a set of configurations inside X_{obs}, then determine the standard Voronoi partition of X using $\mathcal{N}_X$. The $\alpha(q)$ for each $q \in \mathcal{N}_G$ is the area of its associated cell in this partition. A description of the external boundary of X is needed. For the remainder of the paper, nodes refer to the vertices in the graph, $q \in \mathcal{N}_G$, and not the n agents that move in the environment.

III. Graph-Based Spatial Load Balancing

This section details the Graph-based Spatial Load Balancing (GSLB) algorithm that finds an approximate solution to Problem 1. As a preprocessing step, a PRM*, G, is constructed in the non-convex environment Q. The cost to travel between two configurations (q_1, q_2), $J(q_1, q_2)$, is approximated by the sum of edge costs of the best path in G from q_1 to q_2. Define a partition of $\mathcal{N}_G$ as $\widetilde{\mathcal{W}} = \{\widetilde{W}_i\}_{i=1}^n$, such that $\cup_{i=1}^n \widetilde{W}_i = \mathcal{N}_G$ and $\widetilde{W}_i \cap \widetilde{W}_j = \emptyset$. Given $a_1, \ldots, a_n \in \mathbb{R}_{>0}$, such that $\sum_{i=1}^n a_i = \sum_{q \in \mathcal{N}_G} \phi(q)\alpha(q)$, the agents solve the following graph-based problem,

Problem 2.

$$\begin{aligned} \min \quad & \widetilde{\mathcal{H}}(P, \widetilde{\mathcal{W}}) = \sum_{i=1}^{n} \sum_{q \in \widetilde{W}_i} J(p_i, q)\phi(q)\alpha(q) \\ \text{s.t.} \quad & p_i \in \mathcal{N}_G, \qquad i \in \{1, \ldots, n\} \\ & a_i = \sum_{q \in \widetilde{W}_i} \phi(q)\alpha(q). \end{aligned}$$

Algorithm 1 $(P^*, \widehat{\mathcal{V}}(P^*, \omega^*; J)) \leftarrow \mathrm{GSLB}(P_0, \omega_0, Q)$

```
G ← PRM*(Q);
(P, ω) ← Initialize(P_0, ω_0);
for all {Agent i}_{i=1}^n do
  while P ≠ P^+ do
    P = P^+
    while ‖ω − ω^+‖ < error do
      ω = ω^+;
      V̂_i(P, ω; J) ← VoronoiPartition(P, ω, G);
      ω_i^+ ← UpdateWeights(P, ω, V̂_i, G);
      ω^+ ← TransmitAndReceive(ω_i^+);
    end while
    p_i^+ ← UpdateAgentPosition(P_i, V̂_i, G);
    P^+ ← TransmitAndReceive(p_i^+);
  end while
end for
return (P, V̂(P, ω; J));
```

Each agent has a copy of G, and it is assumed that the agents know the other agents' positions P, and the weights ω by communicating with each other to interchange this information. As in [21], this assumption can be relaxed so it is only necessary to know a subset of the other positions and weights; this will be discussed further in Section VI.

Algorithm 1 briefly outlines the GRAPH-BASED SPATIAL LOAD BALANCING algorithm that leads to an approximate solution of Problem 2. Let $\widehat{\mathcal{V}}$ refer to an approximate generalized Voronoi partition in terms of graph nodes. First, the agents determine a partition, see subsection III-A for definition and details. Then, the weights are updated to reflect the error in the area-constraint, the details of which are in Section III-B. These two steps are alternated until the area-constraints are satisfied to within a specified error, which can be reduced by increasing the number of nodes in G. Next, the agents move to a neighboring node that will decrease $\widetilde{\mathcal{H}}$, see Section III-C. The steps are repeated until none of the agents are able to update their positions, $P = P^+$.

A. Approximate Geodesic Voronoi Tesselation

Using G, define one option for $\widehat{\mathcal{V}}$ as the *approximate generalized Voronoi partition*, $\widetilde{\mathcal{V}} = \{\widetilde{V}_1, \ldots, \widetilde{V}_n\}$,

$$\widetilde{V}_i = \{q \in \mathcal{N}_G \mid J(p_i, q) - \omega_i \leq J(p_j, q) - \omega_j, \ \ \forall\, j \neq i\}. \tag{1}$$

Due to the random selection of q when building G, the probability that, in (1), $q \in \widetilde{V}_i$ and $q \in \widetilde{V}_j$ is nearly zero.

To calculate the approximation of its own Voronoi region, $\widetilde{V}_i$, agent i does a Dijkstra graph search, [22], starting from its current configuration, p_i, and keeps a queue of the vertices it needs to check. To start, p_i is added to $\widetilde{V}_i$, and all the outgoing neighboring nodes of p_i are added to the queue. One node, q_{check}, is taken from the queue and checked to see if it belongs in $\widetilde{V}_i$ using (1). The outgoing neighbors of q_{check} are added to the queue only if $q_{\text{check}} \in \widetilde{V}_i$. Agent i has found $\widetilde{V}_i$ when there are no more nodes in the queue.

1) Lower Approximation for $\widetilde{\mathcal{V}}$: The above requires agent i either have full communication with all other agents in order to receive the required $J(p_j, q)$ information or agent i do a Dijkstra graph search from each p_j to recover $J(p_j, q)$. These options are communicationally and computationally intensive, respectively. The following subsection details an option for $\widehat{\mathcal{V}}$ that is a lower approximation of $\widetilde{V}_i$, which does not require knowledge of every $J(p_j, q)$. However, it assumes that there exists a lower bounding cost that is easier to compute compared to a Dijkstra graph search. The lower approximation results in a partition of a subset of $\mathcal{N}_G$. Define the *lower approximation of the generalized Voronoi partition* as $\widetilde{\mathcal{V}}^{\text{lower}} = \{\widetilde{V}_1^{\text{lower}}, \ldots, \widetilde{V}_n^{\text{lower}}\}$, such that

$$\widetilde{V}_i^{\text{lower}} = \{q \in \mathcal{N}_G \mid J(p_i, q) - \omega_i \leq J_0(p_j, q) - \omega_j, \ \ \forall\, j \neq i\}, \tag{2}$$

where $J_0(p_i, q) \leq J(p_i, q)$. Let $\widetilde{U} = \mathcal{N}_G \setminus \cup_{i=1}^n \widetilde{V}_i^{\text{lower}}$, then $\widetilde{U}$ can be used to recover $\widetilde{\mathcal{V}}$.

Remark 1. Agent i can reduce the number of $q \in \mathcal{N}_G$ whose $J(p_j, q)$ are needed to determine $\widetilde{V}_i$. By definition, any $q \in \widetilde{V}_i^{\text{lower}}$ is also in $\widetilde{V}_i$, therefore, the corresponding $J(p_j, q)$ are not needed. Secondly, using Theorem 1 from Section IV-A, once agent i finds a $q \notin \widetilde{V}_i$ the children of q are also not in $\widetilde{V}_i$. Also, the parent of q is on the boundary of $\widetilde{V}_i$. Thus, $J(p_j, q)$ is not needed for the descendants of q. In other words, agent i only needs to know $J(p_j, q)$ for $q \in \widetilde{V}_i \setminus \widetilde{V}_i^{\text{lower}}$ and q that are one hop out neighbors of the boundary of $\widetilde{V}_i$. Section IV-C discusses when to recover $\widetilde{\mathcal{V}}$.

B. Updating the Agent Weights

Define the error between the current and specified area as

$$\widetilde{g}(\omega) = \left(\sum_{q \in \widehat{V}_1} \phi(q)\alpha(q) - a_1, \ldots, \sum_{q \in \widehat{V}_n} \phi(q)\alpha(q) - a_n \right).$$

Next, each agent updates ω to reduce the area error.

From [23], the Jacobian update is used to minimize $\widetilde{g}(\omega)$. Approximate the partial derivatives of $\widetilde{g}$,

$$\frac{\partial \widetilde{g}}{\partial \omega_i}(\omega) \approx \frac{\sum_{q \in \widehat{V}_i^i} \phi(q)\alpha(q) - \sum_{q \in \widehat{V}_i} \phi(q)\alpha(q)}{\Delta\omega_i}, \tag{3}$$

where $\widehat{V}_i^i$ is the Voronoi cell for agent i with $\omega = \{\omega_1, \ldots, \omega_i + \Delta\omega_i, \ldots, \omega_n\}$, note that $\Delta\omega_i > 0$ needs to be small enough to guarantee convergence but also large enough that $\widehat{V}_i^i \neq \widehat{V}_i$. The $\widehat{V}_i^i$ can be computed with a single Dijkstra graph search so that agent i can easily compute (3). The approximation of the Jacobian update is

$$\omega_i^+ = \omega_i - \gamma \left(\frac{\partial \widetilde{g}}{\partial \omega_i}(\omega) \right)^{-1} \widetilde{g}_i(\omega),$$

which converges for a small enough $\Delta\omega_i > 0$ and $\gamma > 0$.

The GRAPH-BASED SPATIAL LOAD BALANCING algorithm loops through determining $\widehat{\mathcal{V}}$ and then updating ω until the area constraint is satisfied to within a specified error. The sum-of-areas constraint and each a_i constraint cannot

be satisfied when $\widetilde{\mathcal{V}}^{\text{lower}}$ does not contain all $q \in \mathcal{N}_G$. In this case the area constraint, $\tilde{a} = \{\tilde{a}_1, \cdots, \tilde{a}_n\}$, is further relaxed and varies with each algorithm iteration,

$$\tilde{a}_i = \sum_{q \in \widetilde{\mathcal{V}}^{\text{lower}}} \phi(q)\alpha(q) \frac{a_i}{\sum_{q \in \widetilde{\mathcal{V}}} \phi(q)\alpha(q)}.$$

The $\sum_{q \in \widetilde{\mathcal{V}}^{\text{lower}}} \phi(q)\alpha(q)$ requires knowledge from all agents.

C. Updating the Agent Positions

After $\widehat{V}_i$ and ω have been determined, agent i decides where to move. Ideally, each agent moves to a generalized centroid of $\widehat{V}_i$, which is computationally intensive. Instead, agent i moves in the direction of a generalized centroid by moving to a neighboring node of p_i such that

$$p_i^+ \in \underset{p \in \mathcal{N}_G^{\text{out}}(p_i)}{\operatorname{argmin}} \sum_{q \in \widehat{V}_i} J(p, q)\phi(q)\alpha(q).$$

IV. Analysis

This section contains analytical results for the Graph-based Spatial Load Balancing algorithm. First, properties for $\widehat{\mathcal{V}}$ are examined, then the distributed nature of the Graph-based Spatial Load Balancing algorithm is discussed as well as its convergence properties. Due to space limitations, the proofs have been removed but can be found in a longer version online.

A. Properties of $\widehat{\mathcal{V}}$

For $\widetilde{\mathcal{V}}^{\text{lower}}$ to satisfy the area constraints, the weights must remain within the bound presented in Lemma 1.

Lemma 1. *Given $\widetilde{V}_i^{lower}$ as in* (2)*, if $\omega_j - \omega_i > J_0(p_j, p_i)$ then $\widetilde{V}_i^{lower} = \emptyset$.*

The additive property of J_e allows the agents to only check a connected subset of nodes, $q \in \mathcal{N}_G$, to find the approximated regions $\widetilde{V}_i$ and $\widetilde{V}_i^{\text{lower}}$.

Theorem 1. *Let J be an additive cost and let there exist an optimal path from p_i to q passing through q'. Then, if q' is not part of $\widetilde{V}_i$ ($\widetilde{V}_i^{lower}$), then q is not part of $\widetilde{V}_i$ ($\widetilde{V}_i^{lower}$).*

B. Distributed Algorithm Properties

The information that agent i needs to implement the proposed algorithm is limited to those other j whose approximated regions are connected to the approximated region of i via boundary nodes. Depending on the underlying cost, this property can be exploited to derive a distributed algorithm that allows agent i to find these j. With respect to Euclidean distance, following the procedure of [24], each agent grows a communication-radius ball finding these other agents j until the computation of V_i is complete. Alternatively, limited ranges, as in [21], can reduce the information needed from other agents at the expense of a loss in coverage capacity.

C. Convergence

Recall that convergence in the continuous space is briefly discussed in Section II-A. In the discrete case, there exists many $\mathcal{W}$ such that $\widetilde{W}_i = W_i \cap \mathcal{N}_G$. As the number of nodes in $\mathcal{N}_G$ goes to infinity, $\widetilde{\mathcal{W}}$ will converge to a $\mathcal{W}$. Due to integration properties, and because $\mathcal{H}$ is continuous,

$$|\mathcal{H}(P, \mathcal{W}) - \widetilde{\mathcal{H}}(P, \widetilde{\mathcal{W}})| \leq \epsilon, \tag{4}$$

is true for a sufficiently small sample dispersion.

The Graph-based Spatial Load Balancing algorithm cannot converge to the exact partition and centroids of the continuous problem, but an approximate solution is guaranteed to be found, see Lemma 2.

Lemma 2. *The* Graph-based Spatial Load Balancing *is guaranteed to converge to an approximate solution $(P^*, \widetilde{\mathcal{V}}(P^*))$ and occurs once $\|\mathcal{H}(P^*, \mathcal{V}(P)) - \mathcal{H}(P^*, \mathcal{V}(P^*))\| \leq 2\epsilon$.*

There ways to guarantee that the Graph-based Spatial Load Balancing algorithm using $\widetilde{\mathcal{V}}^{\text{lower}}$ converges to $(P^*, \widetilde{\mathcal{V}}(P^*, \omega^*; J))$. Option one is to determine $\widetilde{\mathcal{V}}$ from $\widetilde{\mathcal{V}}^{\text{lower}}$ at each iteration as in Remark 1. The second option is to let the $\widetilde{\mathcal{V}}^{\text{lower}}$ algorithm run until it converges, then determine $\widetilde{\mathcal{V}}$ until convergence to $(P^*, \widetilde{\mathcal{V}}(P^*, \omega^*; J))$ is reached. Option two is potentially less communicationally intensive. Indeed, simulations show that only a few additional iterations are needed to reach convergence and there are fewer nodes in $\widetilde{U}$ that need to be checked in order to determine $\widetilde{\mathcal{V}}$. A third option is to pursue an event-based triggering idea similar to the triggering condition of [25]. The trigger, based on the area of $\widetilde{U}$, $\int_{q \in \widetilde{U}} \phi(q)\alpha(q) = \int_{q \in \mathcal{N}_G} \phi(q)\alpha(q) - \sum_{i=1}^{n} \sum_{q \in \widetilde{V}_i^{\text{lower}}} \phi(q)\alpha(q)$, always guarantees the cost is decreasing monotonically; hence convergence is reached by using both $\widetilde{\mathcal{V}}^{\text{lower}}$ and $\widetilde{\mathcal{V}}$. We leave the investigation of the benefits of this approach for future work.

V. Simulations

Two different agent types are simulated. Agents without differential constraints have edge cost, J_e, equal to Euclidean distance. Dubins' vehicles have an edge cost equal to the distance traveled between the two nodes. Problem 2 with an equal area constraint and a uniform probability density function, $\phi(q) = 1 \ \forall \, q \in \mathcal{N}_G$ is solved.

Define the lowest cost path from p_i to q as,

$$\sigma = \{(p_i, q_1), (q_1, q_2), \cdots, (q_s, q)\},$$

where each ordered pair is in $\mathcal{E}_G$, with cost

$$J(p_i, q) = J_e(p_i, q_1) + J_e(q_1, q_2) + \cdots + J_e(q_s, q).$$

A. Euclidean Agents

These results explore the differences between the Graph-based Spatial Load Balancing algorithm using $\widetilde{\mathcal{V}}$ and $\widetilde{\mathcal{V}}^{\text{lower}}$ as the partition of G. Where $\widetilde{\mathcal{V}}^{\text{lower}}$ uses $J_0(p_i, q) = \|p_i - q\|$. Each agent is initialized with $\omega_i = 5$.

Both algorithm versions converge to a solution $(P^*, \widehat{\mathcal{V}}(P^*, \omega^*))$, shown in Fig. 1. Each of the six $\widehat{V}_i$

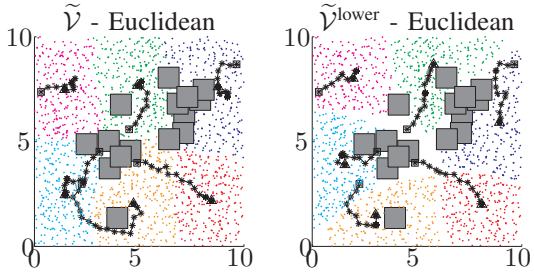

Fig. 1: The final $\widetilde{\mathcal{V}}$ (left) and $\widetilde{\mathcal{V}}^{\text{lower}}$ (right).

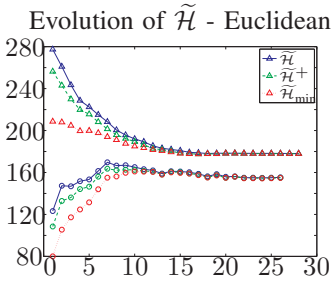

Fig. 2: The $\widetilde{\mathcal{H}}$, for the $\widetilde{\mathcal{V}}$ ($\triangle$) and $\widetilde{\mathcal{V}}^{\text{lower}}$ ($\circ$) algorithm, of the agents' positions (blue), next positions (green), and generalized centroid (red).

are a different color, blue, yellow, cyan, red, green, magenta, respectively. The initial agent positions are marked as $\square$ and the final generalized centroidal positions are $\triangle$. The path taken by an agent is marked with $*$. The two algorithms find similar, but not quite the same partition, due to the approximation of $\widetilde{\mathcal{V}}^{\text{lower}}$ when determining the next positions of the agents. Also note that, there are gaps between any two neighboring $\widetilde{\mathcal{V}}^{\text{lower}}$ cells due to the J_0 approximation.

Fig. 2 is the evolution of $\widetilde{\mathcal{H}}$ from Problem 2. The solid blue lines represent $\widetilde{\mathcal{H}}$ at the current agent positions, the dashed green lines are the values of the agent's best neighboring position, and the red dotted lines are the $\widetilde{\mathcal{H}}$ at the generalized centroid. As expected, in the $\widetilde{\mathcal{V}}$ case ($\triangle$) $\widetilde{\mathcal{H}}$ decreases at each iteration confirming the convergence result. While the $\widetilde{\mathcal{V}}^{\text{lower}}$ case ($\circ$) also converges, the $\widetilde{\mathcal{H}}$ increases because of the increase in total area covered by $\widetilde{\mathcal{V}}^{\text{lower}}$.

Fig. 3 contains the average number of communications an agent needs to perform at each iteration of the algorithm to recover $\widetilde{V}_i$. Note that, as the algorithm converges, fewer communications are needed because $\widetilde{\mathcal{V}}^{\text{lower}}$ becomes a better approximation of $\widetilde{\mathcal{V}}$. The final $\widetilde{\mathcal{V}}^{\text{lower}}$ partition covers

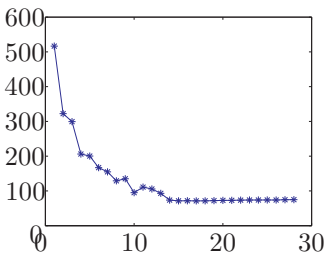

Fig. 3: Communications of $J(p_j, q)$ needed by an agent.

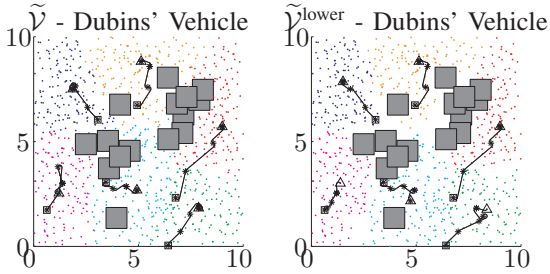

Fig. 4: The final $\widetilde{\mathcal{V}}$ (left) and $\widetilde{\mathcal{V}}^{\text{lower}}$ (right).

89.69% of $\mathcal{N}_G$ (the initial partition only covered 57.47% of $\mathcal{N}_G$). Instead of recovering $\widetilde{\mathcal{V}}$ at each iteration, if the agents converge to $(P^*, \widetilde{\mathcal{V}}^{\text{lower}}(P^*, \omega^*; J))$ then implement the communication procedures from Remark 1 the agents can get a partition that covers the entire set $\mathcal{N}_G$ without the communication overhead. A total of 186 communications of $J(p_j, q)$ are needed to recover $\widetilde{\mathcal{V}}$. The agents can either stop here or continue the algorithm with the communication until it re-converges, which took ten iterations and the agents traveled an average distance of 1.29.

The Euclidean GRAPH-BASED SPATIAL LOAD BALANCING simulation required a total of $23,776$ communications (individual values of p_j, $J(p_j, q)$ and ω_j for all j) for $\widetilde{\mathcal{V}}$ and only $3,390$ communications (individual values of p_j, $\sum_{q \in \widetilde{V}_j^{\text{lower}}} \phi(q)\alpha(q)$, and ω_j for all j) for $\widetilde{\mathcal{V}}^{\text{lower}}$.

B. Dubins' Vehicle

This subsection looks at the results for six Dubins' vehicle agents. The dynamics for the Dubins' vehicle are

$$\dot{x}(t) = v\cos(\theta),\ \dot{y}(t) = v\sin(\theta),\ \dot{\theta}(t) = u,\ |u| \leq \frac{v}{\rho},$$

where v is the speed of the vehicle and ρ is the minimum turning radius, both are assumed to be constant. The optimal trajectory, assuming no obstacles, between two configurations for these dynamic constraints is discussed in [26] and has cost $J_0(q_1, q_2)$. Each agent is initialized with $\omega_i = 5$.

The final $\widetilde{\mathcal{V}}$ and $\widetilde{\mathcal{V}}^{\text{lower}}$ partitions for the Dubins' vehicle simulations are in Fig. 4 with each color represents a different agents' cell (blue, green, magenta, cyan, red, yellow). The agents' paths are shown in black. The agents' final positions ($\triangle$) in the $\widetilde{\mathcal{V}}$ algorithm are close to those from the $\widetilde{\mathcal{V}}^{\text{lower}}$ algorithm, which means the partitions are also close.

Fig. 5 is the evolution of $\widetilde{\mathcal{H}}$ from Problem 2. The solid blue lines represent $\widetilde{\mathcal{H}}$ at the current agent positions, the dashed green lines are the values of the agent's best neighboring position, and the red dotted lines are the $\widetilde{\mathcal{H}}$ at the generalized centroid. As expected, in the $\widetilde{\mathcal{V}}$ case ($\triangle$), $\widetilde{\mathcal{H}}$ decreases at each iteration. While the $\widetilde{\mathcal{V}}^{\text{lower}}$ case ($\circ$) converges, $\widetilde{\mathcal{H}}$ is allowed to increase because of the increase in area covered by $\widetilde{\mathcal{V}}^{\text{lower}}$.

Fig. 6 shows the average number of communications a Dubins' vehicle agent needs to receive to recover $\widetilde{\mathcal{V}}$ from $\widetilde{\mathcal{V}}^{\text{lower}}$. The $\widetilde{\mathcal{V}}^{\text{lower}}$ Dubins' vehicle algorithm initially finds a partition that covers 79.80% of $\mathcal{N}_G$ but the final $\widetilde{\mathcal{V}}^{\text{lower}}$ partition covers 93.95% of $\mathcal{N}_G$. If the agents converge to $(P^*, \widetilde{\mathcal{V}}^{\text{lower}}(P^*, \omega^*; J))$ then do 58 $J(p_j, q)$ communications,

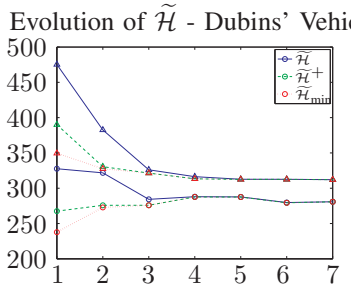

Fig. 5: The $\widetilde{\mathcal{H}}$, for the $\widetilde{\mathcal{V}}$ ($\triangle$) and $\widetilde{\mathcal{V}}^{\text{lower}}$ ($\circ$) algorithm, of the Dubins' vehicle agents' positions (blue), next positions (green), and generalized centroids (red).

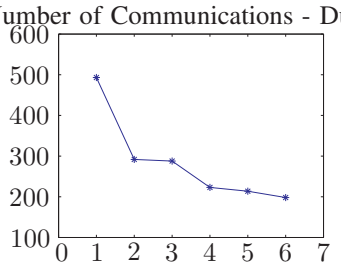

Fig. 6: Communications of $J(p_j, q)$ needed by an agent.

the agents recover a partition that covers the set $\mathcal{N}_G$. The agents either stop here or continue the algorithm with communication until re-convergence. The agents re-converge in four iterations and move an average distance of 1.61.

The Dubins' vehicle GRAPH-BASED SPATIAL LOAD BALANCING simulation required a total of $11,722$ communications (individual values of p_j, $J(p_j, q)$ and ω_j for all j) for $\widetilde{\mathcal{V}}$ and only $2,910$ communications (individual values of p_j, $\sum_{q \in \widetilde{V}_j^{\text{lower}}} \phi(q)\alpha(q)$ and ω_j for all j) for $\widetilde{\mathcal{V}}^{\text{lower}}$.

VI. CONCLUSION AND FUTURE WORK

An algorithm for obtaining an approximate solution to the spatial load balancing problem for agents with differential constraints in non-convex environments is presented. A probabilistic roadmap is used to define two approximate generalized Voronoi partitions, $\widetilde{\mathcal{V}}$ or $\widetilde{\mathcal{V}}^{\text{lower}}$. The convergence of the algorithm is discussed and confirmed via simulation. With extra communication between the agents, $\widetilde{\mathcal{V}}$ can be constructed from $\widetilde{\mathcal{V}}^{\text{lower}}$.

The algorithm is adaptable to most multi-agent coverage problems. Future work includes analysis and simulations for distributed frameworks and limited ranges. Areas for exploration include deployment of manipulators and environments with unknown obstacles.

ACKNOWLEDGMENT

This work was supported by Los Alamos National Laboratory and is approved for release under LA-UR-15-27452.

REFERENCES

[1] S. Lloyd, "Least squares quantization in PCM," *IEEE Transactions on Information Theory*, vol. 28, no. 2, pp. 129–137, 1982.

[2] K. Laventall and J. Cortés, "Coverage control by robotic networks with limited-range anisotropic sensory," in *American Control Conference*, Seattle, WA, 2008, pp. 2666–2671.

[3] Y. Stergiopoulos and A. Tzes, "Coverage-oriented coordination of mobile heterogeneous networks," in *Mediterranean Conf. on Control and Automation*, 2011, pp. 175–180.

[4] L. Pimenta, V. Kumar, R. C. Mesquita, and G. Pereira, "Sensing and coverate for a network of heterogeneous robots," in *IEEE Int. Conf. on Decision and Control*, Cancun, Mexico, December 2008, pp. 3947–3952.

[5] A. Kwok and S. Martínez, "Unicycle coverage control via hybrid modeling," *IEEE Transactions on Automatic Control*, vol. 55, no. 2, pp. 528–532, 2010.

[6] J. Enright, K. Salva, and E. Frazzoli, "Coverage control for nonholonomic agents," in *IEEE Int. Conf. on Decision and Control*, 2008, pp. 4250–4256.

[7] K. Savla, F. Bullo, and E. Frazzoli, "The coverage problem for loitering Dubins vehicles," in *IEEE Int. Conf. on Decision and Control*, New Orleans, LA, Dec. 2007, pp. 1398–1403.

[8] A. Kwok and S. Martínez, "Deployment algorithms for a power-constrained mobile sensor network," *International Journal on Robust and Nonlinear Control*, vol. 20, no. 7, pp. 725–842, 2010.

[9] J. Cortés, "Coverage optimization and spatial load balancing by robotic sensor networks," *IEEE Transactions on Automatic Control*, vol. 55, no. 3, pp. 749–754, 2010.

[10] R. Patel, P. Frasca, and F. Bullo, "Centroidal area-constrained partitioning for robot networks," *ASME Journal on Dynamic Systems, Measurement, and Control*, vol. 136, no. 3, pp. 031 024–1–031 024–8, 2014.

[11] M. Pavone, A. Arsie, E. Frazzoli, and F. Bullo, "Distributed algorithms for environment partitioning in mobile robotic networks," *IEEE Transactions on Automatic Control*, vol. 56, no. 8, pp. 1834–1848, 2011.

[12] C. H. Caicedo-Nunez and M. Zefran, "Performing coverage on non-convex domains," in *IEEE Conf. on Control Applications*, 2008, pp. 1019–1024.

[13] 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.

[14] A. Breitenmoser, M. Schwager, J.-C. Metzger, R. Siegwart, and D. Rus, "Voronoi coverage of non-convex environments with a group of networked robots," in *IEEE Int. Conf. on Robotics and Automation*, 2010, pp. 4982–4989.

[15] A. Renzaglia and A. Martinelli, "Distributed coverage control for a multi-robot team in a non-convex environment," in *IEEE/RSJ Int. Conf. on Intelligent Robots & Systems*, 2009.

[16] S. Bhattacharya, N. Michael, and V. Kumar, "Distributed coverage and exploration in unknown non-convex environments," in *International Symposium on Distributed Autonomous Robotic Systems*. Springer, 2013, pp. 61–75.

[17] A. Breitenmoser, J.-C. Metzger, R. Siegwart, and D. Rus, "Distributed coverage control on surfaces in 3d space," in *IEEE/RSJ Int. Conf. on Intelligent Robots & Systems*, 2010, pp. 5569–5576.

[18] J. W. Durham, R. Carli, P. Frasca, and F. Bullo, "Discrete partitioning and coverage control for gossiping robots," *IEEE Transactions on Robotics*, vol. 28, no. 2, pp. 364–378, 2012.

[19] H. Flanders, "Differentiation under the integral sign," *The American Mathematical Monthly*, vol. 80, no. 6, pp. 615–627, June 1973.

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

[21] F. Bullo, J. Cortés, and S. Martínez, *Distributed Control of Robotic Networks*, ser. Applied Mathematics Series. Princeton University Press, 2009, available at http://www.coordinationbook.info.

[22] E. W. Dijkstra, "A note on two problems on connexion with graphs," *Numerische Mathematik*, vol. 1, no. 1, pp. 269–271, 1959.

[23] D. P. Bertsekas and J. N. Tsitsiklis, *Parallel and Distributed Computation: Numerical Methods*. Athena Scientific, 1997.

[24] J. Cortés, S. Martínez, T. Karatas, and F. Bullo, "Coverage control for mobile sensing networks," in *IEEE Int. Conf. on Robotics and Automation*, Washington, USA, 2002, pp. 1327–1332.

[25] Y. Ru and S. Martínez, "Coverage control in constant flow environments based on a mixed energy-time metric," *Automatica*, vol. 49, pp. 2632–2640, 2013.

[26] L. E. Dubins, "On curves of minimal length with a constraint on average curvature, and with prescribed initial and terminal positions and tangents," *American Journal of Mathematics*, pp. 497–516, 1957.