52nd IEEE Conference on Decision and Control
December 10-13, 2013. Florence, Italy

Shortest Path Set Induced Vertex Ordering and its Application to Distributed Distance Optimal Formation Path Planning and Control on Graphs

Jingjin Yu Steven M. LaValle

Abstract— **For the task of moving a group of indistinguishable agents on a connected graph with unit edge lengths into an arbitrary goal formation, it was shown that distance optimal paths can be computed to complete with a tight convergence time guarantee [30], using a fully centralized algorithm. In this study, we establish the existence of a more fundamental ordering of the vertices on the underlying graph network, induced by a fixed goal formation. The ordering leads to a simple distributed scheduling algorithm that assures the same convergence time. The vertex ordering also readily extends to more general graphs - those with arbitrary integer capacities and edge lengths - for which we again provide guarantees on the convergence time until the desired formation is achieved. Simulations, accessible via a web browser,[1] confirm our theoretical developments.**

I. Introduction

For the task of moving a group of n indistinguishable *agents* (or equivalently, *robots* or *vehicles*) on a connected graph with unit length edges into an arbitrary goal formation, an efficient centralized algorithm in [30] schedules all agents from an initial formation (configuration) to a goal formation, along paths with the smallest total distance. The authors further showed that, the schedule can be completed in $n+\ell-1$ time steps (ℓ is the longest distance between a pair of start and goal vertices), which is a tight bound.

In this paper, we significantly extend the previous results and show that, a directed acyclic graph (DAG) induced by the initial and goal formations admits an integral ordering of the vertices on the involved paths. The ordering, which may be used to compute the distance between any two vertices on a directed path of the DAG, is unique up to an additive constant. A simple algorithm based on this vertex ordering yields the same $n+\ell-1$ convergence time guarantee. This more fundamental structure provides a smooth transition from the problem formulation to the solution, which is missing from the constructive proof offered in [30].

Using this vertex ordering structure, once the initial agent-target assignment is completed, the agents, via local (up to distance two) communication, can achieve the desired formation, again in no more than $n+\ell-1$ time steps. To the best of our knowledge, this work provides the first multi-agent formation path planning algorithm that is both distance optimal and partially distributed, along with a tight convergence time guarantee. In general, global distance optimality is not achievable without direct or indirect global communication under our formulation[2], implying that a fully distributed planning algorithm is not possible. As we will see, the ordering also allows easy extension of the results to graphs with edges having arbitrary integer lengths and non-unit capacities (i.e., more than one agent may be traveling on the same edge at a given instant).

When it comes to problems on *formation*, two sub-problems come up. One of them is on the topic of formation control, which focuses on maintaining a formation of a group of vehicles; a desired formation, in these research, may be important for inter-vehicle communication or for maximizing certain utility functions [5, 24, 32]. Graph theoretic approaches are quite popular here, probably because vehicles and inter-vehicle constraints can be represented naturally with vertices and edges of graphs. The second sub-problem put more emphasis on how to achieve a desired formation via planning [4, 7, 9, 10, 15, 16, 17, 19, 20, 23, 28, 26, 29, 30], rather than to stabilize around a given formation. Among these, [15, 16, 17] appear to be mostly close to our effort in this paper (besides our earlier effort [30]). However, these works did not consider the issue of convergence time.

Generalizing the notion of formation to include multiple agents trying to agree on some common goal leads to the problem of consensus and rendezvous. This more general problem has remained a central research topic in control theory and robotics; see, e.g., [1, 2, 3, 6, 8, 10, 12, 13, 14, 18, 21, 22, 24, 25, 27, 31], to list a few. An early account of the rendezvous problem, as a form of formation control, appeared in [1], in which algorithmic solutions are provided for agents with limited range sensing capabilities. An n-dimensional rendezvous problem was approached via proximity graphs in [3]. For the consensus problem it is shown that averaging the behavior of close neighbors causes all agents to converge to the same behavior eventually [8]. We point out that, although this paper works with initial and goal vertex sets of n distinct elements each, the presented results can be easily generalized to any number of goal

Jingjin Yu is with the Computer Science and Artificial Intelligence Laboratory, Massachusetts Institute of Technology, Cambridge, MA 02139 USA. E-mail: jingjin@csail.mit.edu. Steven M. LaValle is with the Department of Computer Science, University of Illinois at Urbana-Champaign, Urbana, IL 61801 USA. E-mail: lavalle@uiuc.edu. **Acknowledgments.** This work was supported in part by NSF grants 0904501 (IIS Robotics) and 1035345 (Cyberphysical Systems), MURI/ONR grant N00014-09-1-1052, and AFOSR grant FA95501210193. We thank the anonymous reviewers for their constructive comments that helped improve the quality of the final presentation of the materials.

[1] `http://msl.cs.uiuc.edu/~jyu18/pe/distr-form.html`. A Java plug-in of version 6 or higher is required.

[2] A simple example: two agents occupy the diagonals of a square with two targets located on the other diagonal of the same square. Distance optimality is only possible if the two agents choose different targets before starting moving, which is not achievable without some form of global communication (direct or indirect).

vertices between 1 and n, thus covering additional problems such as multi-agent rendezvous.

The rest of the paper is organized as follows. Section II provides the problem formulation, an example, and its solution. Section III constructively proves the existence of the aforementioned vertex ordering on the induced DAG, followed by an application that schedules a set of distance optimal paths for the agents with a proven convergence time bound in Section IV. Section V then shows the scheduling algorithm can be easily turned into a distributed one, without relaxing the convergence time bound. We generalize the graph to have integer edge lengths and capacities in Section VI and conclude in Section VII.[3]

II. Formation Path Planning on Graphs

Let $G = (V,E)$ be a connected, undirected, simple graph, in which $V = \{v_i\}$ is its vertex set and $E = \{(v_i, v_j)\}$ is its edge set. Let $A = \{a_1, \ldots, a_n\}$ be n agents that move with unit speeds along the edges of G, with initial and goal vertices on G specified by the injective maps $x_I, x_G : A \to V$, respectively. For convenience, V,E also denote cardinalities of the sets V,E, respectively. Let σ be a permutation that acts on the elements of x_G, $(\sigma \circ x_G)$ is a map that defines a possible goal vertex assignment (a target formation).

A *scheduled path* is a map $p_i : \mathbb{Z}^+ \to V$, in which $\mathbb{Z}^+ := \mathbb{N} \cup \{0\}$. Intuitively, the domain of the paths is discrete time steps. A scheduled path p_i is *feasible* for a single agent a_i if it satisfies the following properties: (1) $p_i(0) = x_I(a_i)$, (2) for each i, there exists a smallest $k_{\min} \in \mathbb{Z}^+$ such that $p_i(k_{\min}) = (\sigma \circ x_G)(a_i)$ for some fixed σ (i.e., same σ for all $1 \le i \le n$) (that is, the end point of the path p_i is some unique goal vertex), (3) for any $k \ge k_{\min}$, $p_i(k) \equiv (\sigma \circ x_G)(a_i)$, and (4) for any $0 \le k < k_{\min}$, $(p_i(k), p_i(k+1)) \in E$ or $p_i(k) = p_i(k+1)$.

We say that two paths p_i, p_j are in *collision* if there exists $k \in \mathbb{Z}^+$ such that $p_i(k) = p_j(k)$ (*meet*, or collision on a vertex) or $(p_i(k), p_i(k+1)) = (p_j(k+1), p_j(k))$ (*head-on*, or collision on an edge). If $p(k) = p(k+1)$, the agent stays at vertex $p(k)$ between the time steps k and $k+1$.

Problem 1 *Given a 4-tuple (G, A, x_I, x_G), find a set of paths $P = \{p_1, \ldots, p_n\}$ and a fixed σ such that p_i's are feasible paths for respective agents a_i's for this σ and no two paths p_i, p_j are in collision.*

Note that in the definition, we assume that edges of G have unit lengths and capacities. That is, it takes unit time for an agent to cross an edge and no two agents can be on the same edge at the same time. This implicit assumption is used throughout Section III-V and relaxed in Section VI.

To familiarize readers with the problem and its solution, look at the example in Fig. 1. The underlying graph G is a 6×7 grid with holes. Assigning the top left corner coordinates $(0,0)$ and bottom right coordinates $(6,5)$, $x_I(A) = \{(0, i-1)\}, x_G(A) = \{(6, i-1)\}, 1 \le i \le 6$. That is,

[3]Due to the page limit, theorems are stated without proofs. Readers interested in the proofs of the results can find them in the full version of this paper at `http://msl.cs.uiuc.edu/~jyu18/files/cdc13.pdf`.

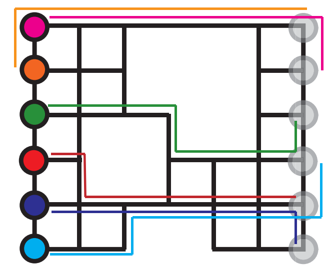

Fig. 1. A 6×7 grid with some vertices removed. The colored discs on the left represent the initial formation and the gray discs the goal formation. The colored paths represent the paths (not yet scheduled to avoid collision).

we want to move the agents from left to right. A solution to this problem that is *distance optimal* is given in Table I, corresponding to a *schedule* of the multi-colored paths in Fig. 1. Here, *distance optimality* seeks to minimize the total path lengths of all agents. Each main entry of the table designates the coordinates of the vertex an agent should be staying at the given time step.

TABLE I

Agent	Time Step								
	0	1	2	3	4	5	6	7	8
1	0,0	1,0	2,0	3,0	4,0	5,0	6,0	6,1	6,1
2	0,1	0,0	1,0	2,0	3,0	4,0	5,0	6,0	6,0
3	0,2	1,2	2,2	3,2	3,3	4,3	5,3	6,3	6,2
4	0,3	1,3	1,4	1,4	2,4	3,4	4,4	5,4	6,4
5	0,4	1,4	2,4	3,4	4,4	5,4	6,4	6,5	6,5
6	0,5	1,5	2,5	2,4	3,4	4,4	5,4	6,4	6,3

III. Formation Induced Vertex Ordering

Algorithm 1 PlanShortestPathSet

Input: G, A, x_I, x_G as described in Problem 1
Output: $Q = \{q_1, \ldots, q_n\}$

1: **for** each $u_i \in x_I(A)$ **do**
2: run breadth first search to get shortest paths q_{ij} for all (u_i, v_j)'s such that $v_j \in x_G(A)$
3: **end for**
4: run Hungarian method on the above set of n^2 paths to get a path set Q.
5: **return** Q

Given x_I and x_G, it is relatively straightforward to obtain an *unscheduled* path set $Q = \{q_1, \ldots, q_n\}$ in which q_i is a sequence of adjacent vertices (we use Q to distinguish these paths from the scheduled paths, denoted P), with the help of the Hungarian method [11]. Our implementation is outlined in Algorithm 1. Let $head(q_i)$, $tail(q_i)$, and $len(q_i)$ denote the start vertex, end vertex, and length of q_i, respectively. The path set Q returned from Algorithm 1 has several obvious properties, listed below.[4]

[4]Properties 2-5 and Proposition 6 are from [30]; they are restated here to make this paper more self-contained.

Property 2 *For all* $1 \le i \le n$, *head*$(q_i) \in x_I(A)$ *and tail*$(q_i) \in x_G(A)$. *For any two paths* q_i, q_j, *head*$(q_i) \neq$ *head*(q_j) *and tail*$(q_i) \neq$ *tail*(q_j).

Property 3 *Each path* q_i *is a shortest path between head*(q_i) *and tail*(q_i) *on* G.

Property 4 *The total length of the path set* Q *is minimal.*

Constructively guaranteed by Algorithm 1, Properties 2 and 3 ensure that the initial and goal vertices are paired up using shortest paths. Property 4 requires the total length of these paths to be minimal. From now on, Q is always assumed to be a path set satisfying properties 2-4. It is not hard to see that Property 4 implies the following.

Property 5 *If the edges of every path* $q_i \in Q$ *are oriented from head*(q_i) *to tail*(q_i), *no two paths share a common edge oriented in different directions.*

Let $V(\cdot), E(\cdot)$ denote the vertex set and the undirected edge set of the input arguments, which can be either a path, q_i, or a set of paths, such as Q. We define an *intersection* between two paths as a maximal consecutive sequence of vertices and edges common to the two paths. Property 5 is a special case of a more general structure of the path set Q, stated in the following proposition.

Proposition 6 *The path set* Q *induces a directed acyclic graph (DAG) structure on* $E(Q)$.

Proposition 6 leads to a tight bound on the number of time steps to schedule the path set Q [30]. Somewhat surprisingly, the DAG structure on Q has an even stronger *vertex ordering* property that does not hold for DAGs in general; this is where the contribution of this paper starts. To state the property, we need some definitions for describing relationships between paths. Recall that two paths *intersect* (a symmetric relationship) if they share some common vertices or edges. Two paths q_i, q_j are *linked* (again a symmetric relationship) if either q_i, q_j intersect or both q_i, q_j are *linked* to some q_k (note that this is an inductive definition with a base case). A *cluster* Q_c is a set of paths such that every pair of paths $q_i, q_j \in Q_c$ are linked. A path cluster Q_c is a *maximal* cluster of Q if Q_c is a cluster and no other path $q_i \in Q \backslash Q_c$ is linked to a path $q_j \in Q_c$.

For each path $q_i \in Q$, a *distance value function*, $d_i : V(q_i) \to \mathbb{Z}^+$, is defined as

$$d_i(u) = \begin{cases} 0 & u = head(q_i), \\ dist(head(q_i), u) & otherwise, \end{cases} \tag{1}$$

in which $dist(u,v)$ denotes the shortest distance between u,v on the graph G. Distance value functions can be defined similarly for an arbitrary set of vertices. Given the generalized definition, we say that one distance value function, d', *respects* another one, d, if d' is defined for all of d's domain and for any u,v on which d is defined,

$$d'(u) - d'(v) = d(u) - d(v). \tag{2}$$

In an unscheduled path set Q, for any two paths q_i, q_j that intersect, a distance value function can be constructed to respect both d_i and d_j.

Lemma 7 *If a vertex* u^* *belongs to the intersection of two paths* $q_i, q_j \in Q$, *then the distance value function*

$$d_c(u) = \begin{cases} d_i(u) & u \in V(q_i), \\ d_c(u^*) + d_j(u) - d_j(u^*) & u \in V(q_j), \end{cases} \tag{3}$$

respects both d_i *and* d_j.

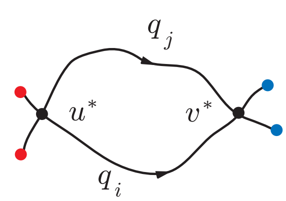

Fig. 2. Two intersections between two paths.

We now show that (3) can be extended to a path cluster.

Theorem 8 *Given a path cluster,* $Q_c = \{q_1, \ldots, q_m\} \subset Q$, *there exists a distance value function* $d_c : V(Q_c) \to \mathbb{Z}^+$, *such that* d_c *respects* d_i *for all* $1 \le i \le m$.

IV. An Ordering-Based Scheduling Algorithm

Assuming that a *time optimal* schedule seeks to minimize the time it takes the last agent to reach its goal, the following was established in [30].

Lemma 9 *In general, distance optimality and time optimality for Problem 1 cannot be simultaneously satisfied.*

Furthermore, let ℓ be the largest pairwise distance between a member of $x_I(A)$ and a member of $x_G(A)$,

$$\ell = \max_{\forall u \in x_I(A), v \in x_G(A)} \text{dist}(u, v). \tag{4}$$

It was also shown in [30] that $n + \ell - 1$ time steps is necessary to schedule a shortest path set Q for an infinite family of instances of Problem 1. It was then shown that an unscheduled path set Q can be turned into a scheduled path set P with a maximum of $n + \ell - 1$ time steps, providing a distance optimal schedule with a tight scheduling time bound. We now show that the vertex ordering induced by x_G leads to a scheduling algorithm with the same guarantees on the scheduled paths' qualities. The new algorithm is simpler to implement and has a better running time of $O(nV \log n)$; it is not clear though, from a first look, that it should provide the said convergence time guarantee.

By Theorem 8, each maximal path cluster $Q_c \subset Q$ can be assigned a distance value function d_c that respects the distance function d_i for each $q_i \in Q_c$. Since these individual d_c's have no common domain, they can be combined to give a global d_c (for a fixed Q). Assuming such a d_c, which can be obtained easily using (**??**). Before scheduling the path set Q, we introduce a subroutine to handle the scenario illustrated in Fig. 3. In the figure, $Q = \{q_1, q_2\}$ with $head(q_i) = u_i, tail(q_i) = v_i$ for $i = 1,2$. This path set cannot be scheduled as is, since q_1 is in the way of q_2. However, as

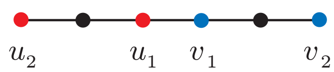

Fig. 3. A path set Q that cannot be scheduled without modification.

agent a_1 reaches v_1, we can dynamically switch the goals of q_1, q_2. Note that the path set after this update still satisfies Properties 2-4. For paths q_i, q_j, denote this path switching subroutine $switch(q_i, q_j)$.

Algorithm 2 SCHEDULESHORTESTPATHS

Input: G, Q, d_c
Output: scheduled paths, $P = \{p_1, \ldots, p_n\}$

let $p_i(0) = head(q_i)$ for all $1 \le i \le n$
let $v_i = next(q_i, head(q_i))$ for all applicable $q_i \in Q$
let $t = 1$
while some q_i is not fully scheduled **do**
  while some $p_i(t)$ is not set for the current t **do**
    pick a candidate path q_i with largest $d_c(v_i)$
    if v_i is not the same as any $p_j(t)$ already assigned **then**
      $p_i(t) = v_i$
      $v_i = next(q_i, v_i)$ if q_i is not fully scheduled
      if $v_i == tail(q_i)$ and v_i falls on some q_j such that q_j has yet to reach v_i **then**
        $switch(q_i, q_j)$
      end if
    else
      $p_i(t) = p_i(t-1)$
    end if
  end while
  $t = t + 1$
end while
return $P = \{p_1, \ldots, p_n\}$

The path scheduling subroutine is outlined in Algorithm 2, in which the routine $next(q_i, v)$ returns the next vertex of path q_i after vertex v. A path q_i is *fully scheduled* if $tail(q_i)$ is assigned to $p_i(t)$ for some t. The scheduling routine never considers two paths q_i, q_j running in opposite directions since Property 5 excludes such cases. Essentially, the scheduling algorithm let all paths from Q take their respective courses simultaneously. Whenever two paths are competing for going to the same vertex, an arbitrary path is picked to go and the other one to stay put. With the $switch(\cdot,\cdot)$ subroutine to guarantee that no deadlock can occur, it is straightforward to see that the process must converge since at each t, at least one agent will make progress toward its goal. That is,

Proposition 10 *Algorithm 2 terminates in finite time.*

Denote the total path length of Q as ℓ_Q, then the convergence time (the time it takes for the formation to be completed) is no more than ℓ_Q. However, as we have mentioned, Algorithm 2 provides a much stronger guarantee.

Theorem 11 *Algorithm 2 provides a schedule that takes at most $n + \ell - 1$ time steps to complete.*

V. A DISTRIBUTED SCHEDULING ALGORITHM

From the constructive proof of Theorem 11 it is clear that within each maximal path cluster, an agent only needs to be aware of its neighbors within a distance of 2 to take appropriate actions. This implies that once agent-target assignment is done, global coordination is not required to *schedule* these agents, yielding partially distributed scheduling algorithm. Since local communication is often more reliable and easy to implement, such a scheduling algorithm is more desirable in general. In this section, we provide a local communication protocol which leads to a distributed scheduling algorithm, again with a convergence time of $n + \ell - 1$. A common clock is assumed. We omit the pseudo code since it is a straightforward modification of Algorithm 2.

Assuming each agent is assigned a path, we will schedule them along these paths and possibly update their goals (targets) on the fly. Recall that by Property 5, we only need to worry about two agents occupying the same vertex at a given time step. This splits into two cases: (1) two agents want to move to the same vertex in one time step, and (2) one agent moves to a vertex while another agent is staying there. We now give a communication protocol, including a forward communication phase and a backward communication phase at each time step, that handles both cases.

Schedule 12 (Distributed Transfer Schedule) *Repeat the following two communication phases until the desired formation is complete.*

Forward communication phase. *Assume that an agent a_i is located on v_i and wants to move to v_{i+1}. Agent a_i first checks whether v_{i+1} is occupied by some other agent a_j and if it is, notifies a_j of its intention and waits for a_j's response. At this point, a_j will check whether it is already at its goal and if it is, switch its goal with a_i (a_j will also redo its forward communication phase if it already did). If no agent is occupying v_{i+1}, a_i then looks for agents that also want to go to v_{i+1}. If there are, one agent is randomly picked to go to v_{i+1} in the next time step. Alternatively, we could deterministically pick an agent (e.g. based on identities of the vertices occupied by the agents). Other agents wanting to go to v_{i+1} then must wait one time step. Since we are dealing a finite number of agents and there are no cycles on a DAG, the forward communication phase will stop after at most $O(n)$ messages, each with a size of $O(\log V)$.*

Backward communication phase. *Next, an agent that has received requests from a following agent needs to respond back. Let such two adjacent agents be a_i and a_j, occupying v_i, v_{i+1}, respectively, with a_i wanting to go to v_{i+1}. There are two sub-cases. In the first sub-case, a_j moves and notifies a_i that it may go ahead and move to v_{i+1}. If a_j gets multiple* *requests to occupy v_{i+1} then a randomly agent is selected to proceed (again, this can be made deterministic). In the second sub-case, a_j cannot move because another agent tells it so. It then simply relay that message backward. Clearly, the backward communication will stop after at most $O(n)$ messages, each with a size of $O(\log V)$.*

Schedule 12 has a similar algorithmic complexity compared with the centralized version. Time wise, we have

Corollary 13 *Schedule 12 transfers all agents to achieve the desired formation in $O(n+\ell-1)$ time steps.*

The scheduling algorithm is fairly simple to implement, as we did in a Java simulation (see abstract for the link). A snapshot of a running session is provided in Fig. 4. We do not provide computational evaluation here since the overall algorithm has similar running time as the algorithm from [30]. Readers interested in computational time on large instances may refer to [30] for more details.

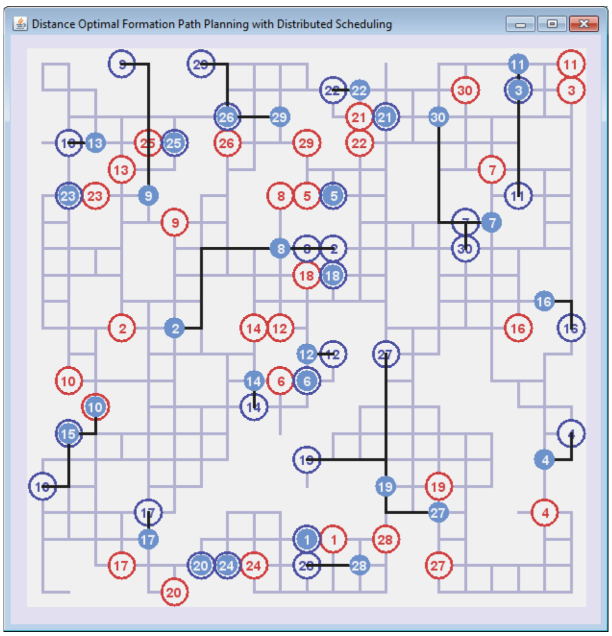

Fig. 4. A simulation capture. The red/blue circles and numbers are the start/goal locations (already assigned to have shortest total distance). The light blue solid discs represent the agents. The bold black lines are the paths yet to be completed.

VI. Integer Edge Lengths and Capacities

So far we have assumed that we work with a graph G with unit edge lengths and capacities. That is, an edge takes a unit of time to cross and can hold one agent at a time. We now relax this assumption to allow non-unit edge lengths and capacities. Formally, let $d, c: E \to \mathbb{Z}^+$ be the edge length map and edge capacity map, respectively. We assume that for any $e \in E, d(e) \geq c(e)$, which is generally true for physical robots with non-negligible sizes (up to a multiplicative constant). The main goal of this section is to extend the results from previous sections under this setup. Note that the definition of *scheduled paths* and *feasible paths* from Section II need to be updated since it may take multiple time steps for an agent to cross an edge. Thus, a scheduled path p_i becomes a partial map as it may be undefined for some time steps. We omit formal descriptions of these required updates since they are intuitive but lengthy to state.

It is clear that Algorithm 1 is insensitive to edge length. Therefore, the algorithm again produces an unscheduled path set Q satisfying Properties 2-5. Moreover, all results from Section III continue to hold with edge lengths that are not all ones. On the other hand, scheduling the path set Q becomes slightly trickier, since depending on edge capacities, one or more agent may be on the same edge during within one time step. To simplify the analysis, we look at two extreme cases: (1) for all $e \in E, c(e) = d(e)$, and (2) for all $e \in E, c(e) \equiv 1$. The first case models scenarios that allow bumper to bumper road traffic. This case is easy to handle, due to the following observation: By subdividing each edge $e \in E$ into $d(e)$ edges of unit length, we obtain a new graph G with unit edge length and capacity. We turn our attention to the second case, which models bottleneck edges such as a long and thin bridge. First we establish a lower bound.

Lemma 14 *Assume $\forall e \in E, c(e) \equiv 1$ and let $d_{\max} = \max_{e \in E} d(e)$. Then $\ell + (n-1)d_{\max}$ time steps is necessary to schedule n agents along a shortest path set Q.*

If we pretend that all edges have the same length $d_{\max}$, Algorithm 2 can be easily extended to schedule a shortest path set Q. Clearly, this provides an overestimate of the total time it takes to schedule Q. Since no agent is delayed more than $(n-1)d_{\max}$ time steps, the following corollary to Theorem 11 is immediate.

Corollary 15 *Assume $\forall e \in E, c(e) \equiv 1$ and let $d_{\max} = \max_{e \in E} d(e)$. Algorithm 2 schedules a shortest path set Q such that the scheduled path set requires at most $\ell + (n-1)d_{\max}$ time steps to complete.*

Thus, the time bound $\ell + (n-1)d_{\max}$ is tight for the unit edge capacity case. Combining the two extreme cases together, we have the following conclusion.

Theorem 16 *For the extension of Problem 1 with integer edge lengths and capacities in which $1 \leq c(e) \leq d(e)$ for all $e \in E$, the time bound $\ell + (n-1)d_{\max}$ is sufficient and necessary to schedule n agents along a shortest path set Q.*

Straightforward complexity analysis shows that for integer edge lengths and capacities, the running time of the entire algorithm becomes $O(nV^2 + nVd_{\max})$.

VII. Conclusion and Future Work

In this paper, for the multi-agent formation path planning problem on graphs, we showed the existence of a vertex ordering structure induced by the initial and goal formations, which in turn admits a simple and natural scheduling algorithm for coordinating the shortest paths amongst the indistinguishable agents with a tight convergence time guarantee.

Furthermore, the ordering allows the scheduling algorithm to be distributed. We then showed that the ordering as well as the convergence time guarantee generalize to integer edge lengths and capacities.

Seeing how the vertex ordering helped us in obtaining a distributed scheduling algorithm without sacrificing convergence time, we plan to study further implications of this order structure. On the practical side, we hope to put the algorithm onto robots to test its performance in real world applications. With increased availability of cheap and fast wireless communication capabilities, we believe our algorithm can be used on formation control problems for a large group of robots or other types of vehicles in practice.

REFERENCES

[1] H. Ando, Y. Oasa, I. Suzuki, and M. Yamashita. Distributed memoryless point convergence algorithm for mobile robots with limited visibility. *IEEE Transactions on Automatic Control*, 15(5):818–828, October 1999.

[2] T. Balch and R. C. Arkin. Behavior-based formation control for multirobot teams. *IEEE Transactions on Robotics & Automation*, 14(6):926–939, 1998.

[3] J. Cortés, S. Martínez, and F. Bullo. Robust rendezvous for mobile autonomous agents via proximity graphs in arbitrary dimensions. *IEEE Transactions on Automatic Control*, 51(8):1289–1298, August 2006.

[4] M. A. Erdmann and T. Lozano-Pérez. On multiple moving objects. In *Proceedings IEEE International Conference on Robotics & Automation*, pages 1419–1424, 1986.

[5] J. A. Fax and R. M. Murray. Information flow and cooperative control of vehicle formations. *IEEE Transactions on Automatic Control*, 49(9):1465–1476, September 2004.

[6] V. Gazi. Stability of a discrete-time asynchronous swarm with time-dependent communication links. *IEEE Transactions on Systems, Man, and Cybernetics: Part B*, 38(1):267–274, February 2008.

[7] Y. Guo and L. E. Parker. A distributed and optimal motion planning approach for multiple mobile robots. In *Proceedings IEEE International Conference on Robotics and Automation*, pages 2612–2619, 2002.

[8] A. Jadbabaie, J. Lin, and A. S. Morse. Coordination of groups of mobile autonomous agents using nearest neighbor rules. *IEEE Transactions on Automatic Control*, 48(6):988–1001, 2003.

[9] K. Kant and S. Zucker. Towards efficient trajectory planning: The path velocity decomposition. *International Journal of Robotics Research*, 5(3):72–89, 1986.

[10] S. Kloder and S. Hutchinson. Path planning for permutation-invariant multirobot formations. *IEEE Transactions on Robotics*, 22(4):650–665, 2006.

[11] H. W. Kuhn. The Hungarian method for the assignment problem. *Naval Research Logistics Quarterly*, 2:83–97, 1955.

[12] J. Lin, A. S. Morse, and B. D. O. Anderson. The multi-agent rendezvous problem. part 1: The synchronous case. *SIAM Journal on Control and Optimization*, 46(6):2096–2119, November 2007.

[13] J. Lin, A. S. Morse, and B. D. O. Anderson. The multi-agent rendezvous problem. part 2: The asynchronous case. *SIAM Journal on Control and Optimization*, 46(6):2120–2147, November 2007.

[14] Z. Lin, M. Broucke, and B. Francis. Local control strategies for groups of mobile autonomous agents. *IEEE Transactions on Automatic Control*, 49(4):622–629, April 2004.

[15] L. Liu and D. A. Shell. Large-scale multi-robot task allocation via dynamic partitioning and distribution. *Autonomous Robots*, 33(3):291–307, 2012.

[16] L. Liu and D. A. Shell. Tunable routing solutions for multi-robot navigation via the assignment problem: A 3d representation of the matching graph. In *Proceedings IEEE International Conference on Robotics & Automation*, pages 4800–4805, 2012.

[17] D. Miklic, S. Bogdan, R. Fierro, and S. Nestic. A discrete grid abstraction for formation control in the presence of obstacles. In *Proceedings IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3750–3755, 2009.

[18] L. Moreau. Stability of multiagent systems with time-dependent communication links. *IEEE Transactions on Automatic Control*, 50(2):169–182, February 2005.

[19] P. A. O'Donnell and T. Lozano-Pérez. Deadlock-free and collision-free coordination of two robot manipulators. In *Proceedings IEEE International Conference on Robotics & Automation*, pages 484–489, 1989.

[20] J. Peng and S. Akella. Coordinating multiple robots with kinodynamic constraints along specified paths. In J.-D. Boissonat, J. Burdick, K. Goldberg, and S. Hutchinson, editors, *Algorithmic Foundations of Robotics V (WAFR 2002)*, pages 221–237. Springer-Verlag, Berlin, 2002.

[21] W. Ren and R. W. Beard. Consensus seeking in multi-agent systems under dynamically changing interaction topologies. *IEEE Transactions on Automatic Control*, 50(5):655–661, May 2005.

[22] B. Shucker, T. Murphey, and J. K. Bennett. Switching rules for decentralized control with simple control laws. In *American Control Conference*, pages 1485–1492, Jul 2007.

[23] T. Siméon, S. Leroy, and J.-P. Laumond. Path coordination for multiple mobile robots: A resolution complete algorithm. *IEEE Transactions on Robotics & Automation*, 18(1):42–49, February 2002.

[24] B. Smith, M. Egerstedt, and A. Howard. Automatic generation of persistent formations for multi-agent networks under range constraints. *ACM/Springer Mobile Networks and Applications Journal*, 14(3):322–335, June 2009.

[25] S. L. Smith, M. E. Broucke, and B. A. Francis. Curve shortening and the rendezvous problem for mobile autonomous robots. *IEEE Transactions on Automatic Control*, 52(6):1154–1159, June 2007.

[26] D. M. Stipanovic, S. Shankaran, and C. J. Tomlin. Multi-agent avoidance control using an m-matrix property. *Electronic Journal of Linear Algebra*, 12:64–72, May 2005.

[27] H. Tanner, G. Pappas, and V. Kumar. Leader-to-formation stability. *IEEE Transactions on Robotics & Automation*, 20(3):443–455, Jun 2004.

[28] J. van den Berg, J. Snoeyink, M. Lin, and D. Manocha. Centralized path planning for multiple robots: Optimal decoupling into sequential plans. In *Proceedings Robotics: Science and Systems*, 2009.

[29] P. Švestka and M. H. Overmars. Coordinated path planning for multiple robots. *Robotics and Autonomous Systems*, 23(3):125–152, 1998.

[30] J. Yu and S. M. LaValle. Distance optimal formation control on graphs with a tight convergence time guarantee. In *Proceedings IEEE Conference on Decision & Control*, pages 4023–4028, 2012.

[31] J. Yu, S. M. LaValle, and D. Liberzon. Rendezvous without coordinates. *IEEE Transactions on Automatic Control*, 57(2):421–434, February 2012.

[32] M. M. Zavlanos, M. B. Egerstedt, and G. J. Pappas. Graph-theoretic connectivity control of mobile robot networks. *Proceedings of the IEEE*, 99(9):1525–1540, September 2011.