ELSEVIER

Artificial Intelligence 95 (1997) 155-167

Artificial Intelligence

Research Note

Determination of social laws for multi-agent mobilization

Shmuel Onn [1], Moshe Tennenholtz *

Faculty of Industrial Engineering and Management, Technion—Israel Institute of Technology, 32000 Haifa, Israel

Received April 1996; revised June 1997

Abstract

In previous work on artificial social systems, social laws for multi-agent mobilization were hand-crafted for particular network structures. In this work, we introduce an algorithm for the *automatic synthesis* of social laws for multi-agent mobilization in *any* 2-connected graph environment. We introduce the notion of a graph routing and show that the problem of computing a useful social law can be reduced to that of finding a routing in a graph. Then we show that any given 2-connected graph has a routing which can be efficiently constructed, yielding the desired social law for multi-agent mobilization on the given graph. © 1997 Elsevier Science B.V.

Keywords: Synthesis of social laws; Multi-agent mobilization; Graph routing

1. Introduction

A number of coordination methods have been discussed in the AI literature; Some of these methods are concerned with centralized approaches to coordination [4, 15, 26, 27]. Other methods are concerned with decentralized approaches [2], where issues such as negotiations [14], deals [28], and consensus [7] play a major role. The artificial social systems methodology (e.g., [1, 18, 25]) attempts to bridge the gap between the fully centralized and the fully decentralized approaches to coordination. The basic

* Corresponding author. Email: moshet@ie.technion.ac.il. Supported by a Gutwirth Fellowship and by a Technion VPR Grant.

[1] Email: onn@ie.technion.ac.il. Supported by an Alexander von Humboldt Stiftung Fellowship, by the Fund for the Promotion of Research at the Technion, and by a Technion VPR Grant.

idea is that the society will adopt a set of social laws; each agent will be required to obey these laws, and will be able to assume that all others will as well. These laws will on the one hand constrain the plans available to the agent, but on the other hand will guarantee certain behaviors on the part of other agents. A social law may include communication protocols which lead to rational deals in multi-agent encounters [21], or embody rationality constraints for cooperation without communication in such encounters [11]. In general, however, the social law may also include rules which prevent agents from arriving at various multi-agent encounters (e.g., agents may be required to keep the right of the road, which will prevent them from arriving at multi-agent encounters where they might move one against the other), and other forms of rules. The exact semantics of artificial social systems and of the relationships of this approach with the above-mentioned and other lines of research are discussed in detail in [19,25].

The idea of traffic laws for mobile robots illustrates important aspects of the artificial social systems approach. Rather than install a centralized controller or have the robots continually negotiate in order to avoid collisions, we will have robots obey traffic laws such as "keep to the right of the road". This idea has been explored in the work by Shoham and Tennenholtz [24,25]. However, the corresponding laws have been hand-crafted for the particular domain of a grid structure without obstacles. In this paper we wish to advance the state-of-art in the work on social laws by considering two extensions of previous work on social laws for mobile robots:

(1) We consider social laws for robots moving in any 2-connected graph structure. This gives a very general framework for the discussion of social laws.
(2) We discuss the *automatic synthesis* of social laws for mobile robots. Although the synthesis of social laws has been discussed in the context of abstract models [23], work on applying social laws to the domain of mobile robots has been concerned with laws which are hand-crafted for a particular domain. In this paper, we bridge the gap between the related lines of research, by introducing a polytime algorithm for the synthesis of an appropriate social law for any 2-connected graph environment.

Our results here allow for the synthesis of social laws for any specified network. An important issue to be studied in future concerns situations where the structure of the underlying graph is *part of the design* and not prescribed ahead of time. The notion of *graph routing* introduced in this paper and our algorithm suggest that certain classes of *polytopal graphs* (such as those studied in [8,13,20]) may provide network structures over which particularly efficient social laws can be synthesized.

Our article is organized as follows. In Section 2 we discuss multi-robot networks, which are a general model for agent mobilization, and define (useful) social laws in the context of this model. In Section 3 we introduce the notion of a graph routing, and show that the existence of a routing implies the existence of useful social laws. In Section 4 we show that any 2-connected graph has a routing, and introduce an efficient algorithm for constructing a routing (and therefore an appropriate useful social law) for any given 2-connected multi-robot network. In Section 5 we conclude with a discussion of our setting and results.

2. Social laws

In this paper we discuss *multi-robot networks*, and the use of social laws for agent mobilization in such systems. The multi-robot networks are general models where one may discuss the coordination of multi-agent systems. In particular, these models include the models discussed in the Automated Guided Vehicles (AGVs) literature (see [12,22] for some overview of the related literature).

Definition 1. A *multi-robot network* consists of a graph $G = (V, E)$, a set R of robots, and a strictly positive length function $\lambda : E \rightarrow \mathbb{R}$. The function λ associates with each edge uv of the graph a particular length which determines the distance a robot needs to travel in order to get from u to v. We assume the existence of a clock which measures time units.[2] At each point in time each robot is in a particular node, or at a particular point along the edge between two nodes.

The action taken by a robot is a decision about its direction and velocity. The velocity of a robot is the number of units of distance it passes in a time unit. The robot decides on its direction and velocity whenever it is in a node of the graph. We assume the robots initially enter the graph from a particular (arbitrary) node with a certain time offset from one another, which is determined by the system's designer. When the robots move in the graph, they can not observe each other. In addition, they need to achieve goals which arrive to them in a dynamic fashion. A goal is a request for visiting a particular node of the graph (and leaving it afterwards for obtaining another goal). We would like that all of the robots will be able to obtain their goals while avoiding collisions among the robots. In the body of this paper we refer to point robots, which collide when they are at the same point (i.e., on the same node or at the same point along an edge). However, our algorithm can be used also in the case where a robot occupies a non-negligible space. We return back to this point in Section 5.

In order to guarantee the agents to obtain their goals while avoiding collisions, we use the idea of a social law.

Definition 2. Given a graph $G = (V, E)$, a *social law* determines a subset $A \subseteq E$ of the edges in which the robots are allowed to move, restricts the direction of movement along each edge of A, and restricts the velocity in which the robots are allowed to move along each edge of A.

Hence, a social law in this case is a traffic law. This traffic law should guarantee that each robot will be able to achieve its goals (while avoiding collisions with other robots) *regardless* of what the other robots do, as long as all of the robots obey the law as manifested by the designer.

[2] The particular way in which time is measured, does not play a role in our results.

Definition 3. Given a multi-robot network, a *useful social law* is a social law that guarantees that if all the robots initially enter the graph from an arbitrary fixed node with an offset of one time unit from one another, and all the robots obey the social law, then the following hold:

(1) Collisions are avoided (i.e., two robots will not be in the same location at the same time).

(2) Given a goal for a robot, the robot will be able to devise a plan for reaching this goal regardless of the robot's current location and regardless of the behavior of the other robots.

3. Social laws via routings

In this section we introduce the notions of *graph routing* and *graph rooting*, and show that the design of a social law can be reduced to the problem of finding a routing in a graph. We show that given a routing in the graph one can construct a corresponding useful social law. In the next section we show that routings exist for any 2-connected graph and that routings can be efficiently computed for such graphs.

In this paper we consider only simple graphs—those which contain neither loops nor parallel edges. We now briefly recall some basic graph-theoretic terminology which we shall need (see [3] for details). A cut-vertex in a graph is one whose removal leaves the graph disconnected. A *block* is a graph containing no cut-vertex. A graph is *2-connected* if it is a block and has at least three vertices. The subgraph *induced* by a subset $U \subseteq V$ of the vertices of a graph $G = (V, E)$ is its subgraph $G[U]$ with vertex set U containing all edges $uv \in E$ for which $u, v \in U$. A *block in a graph* is an induced subgraph which is a block and is not properly contained in any other induced subgraph which is a block. If $G[U], G[W]$ are two distinct blocks in G then U and W share at most one vertex. Each edge of G is contained in precisely one block. A digraph (directed graph) is *strongly connected* if it contains a dipath from every vertex to every other vertex.

The above graph-theoretic concepts are standard. We now define the concepts of *routing* and *rooting* which play a fundamental role in our considerations. Throughout, let $\mathbb{N}$ stand for the set of nonnegative integers. Let $f : V \rightarrow \mathbb{N}$ be a labeling of the vertices of a graph G by nonnegative integers. For a subset $U \subseteq V$ of vertices, we denote by $\min f(U)$ and $\max f(U)$, respectively, the smallest and largest label of a vertex in U. An *f-minimal* (respectively, *f-maximal*) vertex in U is any $u \in U$ for which $f(u) = \min f(U)$ (respectively, $f(u) = \max f(U)$). A labeling f as above induces a directed graph $G_f = (V, A)$ on the same vertex set as G, whose arc set A is obtained from E by removing each edge $uv \in E$ with $f(u) = f(v)$, and orienting each edge $uv \in E$ with $f(u) < f(v)$, in the direction vu if v is f-maximal and u is f-minimal in the block containing uv, and in the direction uv otherwise. Note that for 2-connected graphs, edges are simply oriented from the f-smaller to the f-larger vertex, except that edges connecting an f-minimal vertex to an f-maximal one are oriented the other way.

Definition 4. A *routing* of a graph G is a labeling $f : V \to \mathbb{N}$ of its vertices by nonnegative integers for which the induced digraph G_f is strongly connected. A routing under which there is a unique f-minimal vertex r in V, called a *root*, will be called a *rooting*. Letting $p := \max f(V) - \min f(V) + 1$, the routing (respectively, rooting) will be called a *p-routing* (respectively, *p-rooting*).

We now show that a routing of a graph can serve as a basis for useful social laws in multi-robot networks. Let a given multi-robot network consist of a graph $G = (V, E)$, a set R containing m robots, and a strictly positive length function $\lambda : E \to \mathbb{R}$. Suppose that f is a p-routing of G and let $G_f = (V, A)$ be the induced digraph. Let $D = 1 + \max\{0, m - p\}$. Define the following social law, which we name "Network Traffic Law", for this multi-robot network.

Network Traffic Law.

(1) Robots are required to enter the network from an f-minimal vertex with an offset of one time unit from one another.

(2) Robots are required to move only along the arcs of the graph G_f induced by the routing f.

(3) Define a velocity function $\nu : A \to \mathbb{R}$ as follows: for $e = uv \in A$, put $\nu(e) = \lambda(e)/D$ if u is f-maximal and v is f-minimal, and put $\nu(e) = \lambda(e)/(f(v) - f(u))$ otherwise. Robots are required to move only in the calculated velocities.

A multi-robot network is *2-connected* if its underlying graph is. The significance of the above Network Traffic Law is manifested by the following theorem.

Theorem 5. *Suppose a given 2-connected multi-robot network admits a routing. Then the derived Network Traffic Law is a useful social law.*

Proof. First, it is clear that if agents collide, then this collision should occur in a node of the graph: this is implied by the fact that agents move in fixed velocity along each edge of the graph. Without loss of generality, let $f(v) = 0$ for the f-minimal node v and let the range of f be the integers between 0 to $p - 1$. By construction we get that if a robot gets to a node u, which is not f-minimal, then it gets there exactly after $f(u)$ time units from the last time it visited the f-minimal node, regardless of the actual route the robot has followed. The calculation of the velocity for moving from an f-maximal node to the f-minimal node guarantees that the time for completing a round trip starting at the f-minimal node is independent of the route taken, and that each robot would not collide with a robot which has not entered the graph yet. This implies that no collisions might occur. By definition of a routing, each robot can get from any node of the graph to any other node of the graph. Combining the above we get the desired result. □

Remark. If, due to physical limitations on the robots, an upper bound on the allowed velocities is prescribed, then Network Traffic Law above can be modified so as to obey this bound simply by dividing all velocities by a suitable factor.

4. Synthesis of social laws and graph rootings

The previous section has shown a reduction of social laws to graph routings. If we are able to generate a graph routing, then we are able to define an appropriate useful social law. One question that arises is whether there are interesting graphs for which a routing exists. As we will show below, any 2-connected graph has a routing. This implies that a very large family of problems can be handled by the social laws methodology. Notice that the mobilization of AGVs is usually discussed in the context of 2-connected graphs [22]. A second question is whether it is possible to automatically and efficiently synthesize the appropriate routings. The answer is positive: our proof that any 2-connected graph has a routing is constructive and provides a simple polytime algorithm for the automatic synthesis of a routing for any given 2-connected graph. This algorithm is the topic of the rest of this section.

The routing produced by our algorithm will in fact be a rooting, and the root could be an arbitrary vertex chosen by the system designer. We therefore allow the input to the algorithm, in addition to a 2-connected graph, to include a designation of an arbitrary vertex to be the root.

Rooting Algorithm.

- Input: A 2-connected graph $G = (V, E)$ with designated vertex $r \in V$.
- Initialization: Pick an edge $sr \in E$, put $f(r) = 0$ and $f(s) = 1$.
- Iteration: While there are unlabeled vertices do:
  1. Pick an edge u_0u_1 connecting a labeled vertex u_0 to an unlabeled vertex u_1 and extend it to a path $u_0, u_1, \ldots, u_k$ whose end points $u_0 \neq u_k$ are already labeled but internal points are not.
  2. Reversing the order of indices of the u_i if necessary, assume that $f(u_0) \leqslant f(u_k)$. Label the internal points $u_1, \ldots, u_{k-1}$ by $f(u_i) = f(u_0) + i$. If $h := f(u_{k-1}) - f(u_k) \geqslant 0$ then increase by $h + 1$ the label of each labeled vertex $u \notin \{u_0, \ldots, u_{k-1}\}$ satisfying $f(u) \geqslant f(u_k)$.
- Output: A rooting f of G with root r.

Remarks. The algorithm can be implemented so as to run in time linear in the number of edges. The algorithm can in fact be used to test the 2-connectivity of the input graph: simply, if in step 1 of the iteration it is impossible to construct a path as desired, then G is not 2-connected.

Before establishing the validity of the algorithm, we demonstrate it through the following example.

Example 6. *Three-dimensional cube.* Let G be the graph of the 3-dimensional cube, drawn in Fig. 1, with the root r being the vertex labeled 0. The construction of the rooting of G is demonstrated by Figs. 1–4.

Now consider the multi-robot network on the 3-cube in which all edge lengths equal a unit (i.e. $\lambda \equiv 1$) and which contains $m \leqslant 6$ robots. The useful Network Traffic Law for

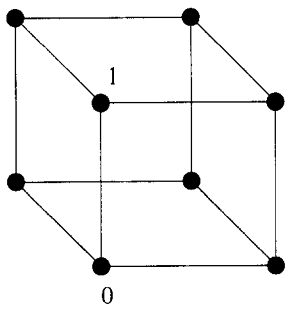

Fig. 1. Rooting the 3-cube: initialization step.

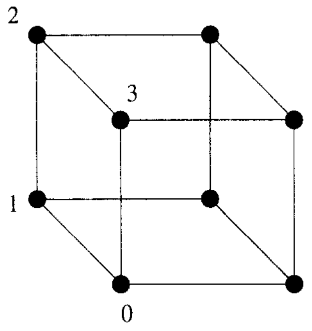

Fig. 2. Rooting the 3-cube: first iteration.

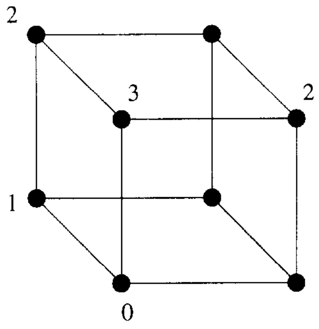

Fig. 3. Rooting the 3-cube: second iteration.

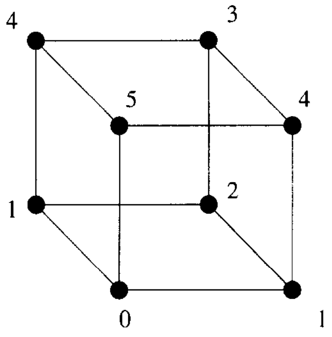

Fig. 4. Rooting the 3-cube: final iteration.

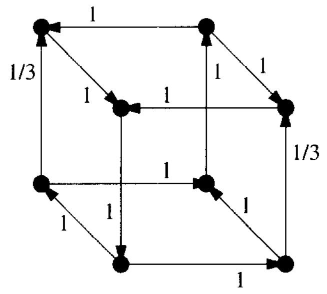

Fig. 5. The derived Network Traffic Law for unit distances and $m \leqslant 6$ robots.

this network derived from the rooting of G is depicted in Fig. 5: the robots are allowed to move along edges in the directions specified by the arrows, and in the velocities specified by the numbers appearing on the edges.

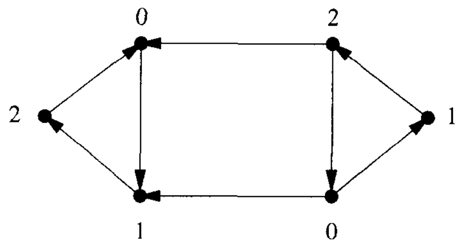

Fig. 6.

We now proceed to prove the correctness of the Rooting algorithm. The characterization of rootings in Proposition 7, together with Proposition 8 will show that the algorithm does what it should. In the sequel, recall that a sink in a digraph is a vertex with no out-going edges, while a source in a digraph is a vertex with no in-going edges.

Proposition 7. *A labeling $f : V \longrightarrow \mathbb{N}$ of the vertices of a 2-connected graph G is a rooting if and only if there is a unique f-minimal vertex and every vertex of G_f is neither a source nor a sink.*

Proof. If G_f has a sink or source then it is not strongly connected, hence f is not even a routing. Conversely, consider any vertex $v \in V$. Since no vertex is a source, a dipath entering v can be traced back, along which the f-value of vertices decreases, so this path can be extended all the way back to r. Likewise, a dipath leaving v can be traced, along which the f-value of vertices increases, so it must reach some f-maximal vertex $s \in V$, and can then be extended by the arc sr, which is in A since s is not a sink. Given now any two vertices u, w, a directed u–w path in G_f is obtained as the concatenation of the u–r dipath and the r–w dipath just shown to exist. Hence G_f is strongly connected. $\square$

Remark. If the requirement for a unique f-minimal vertex is dropped, then f may fail to be even a routing: the labeling f of the vertices of the 2-connected graph G depicted in Fig. 6 induces a digraph G_f which has neither a source nor a sink but is *not* strongly connected, so f is neither a rooting nor a routing of G.

Proposition 8. *Upon the completion of each iteration of the Rooting Algorithm, the labeling f is a rooting of the subgraph $G[U]$ induced by the set U of vertices already labeled, with r being the root.*

Proof. This is easily seen to hold upon the completion of the first iteration, with the resulting $G[U]_f$ being just a directed circuit $(r, u_1, \ldots, u_{k-1}, s, r)$. Suppose that it holds after a certain iteration was completed, and let U be the set of vertices already labeled and $G_f := G[U]_f$ the induced digraph. Consider the next iteration. Since G is 2-connected, a path in step 1 can be constructed, else either the unlabeled vertices are disconnected from the labeled ones or u_1 is a cut-vertex in G. Now, the new labeling in step 2 maintains that r is the unique f-minimal vertex and does not affect the ordering relations $f(u) < f(v)$ between any two old vertices $u, v \in U$. Therefore every vertex of U will remain neither a source nor a sink in the new digraph $G[U \cup \{u_1, \dots, u_{k-1}\}]_f$. Finally, the labeling also assures that each new vertex u_i has the in going arc $u_{i-1}u_i$ and outgoing arc $u_i u_{i+1}$ in that digraph. It follows, by Proposition 7 that f remains a rooting when this next iteration is completed. □

The proposition implies that when the algorithm terminates, the labeling f is a rooting of $G = G[V]$. Hence the following theorem.

Theorem 9. *The Rooting Algorithm constructs a rooting for every 2-connected graph with an arbitrarily chosen root. In particular, every such graph has a rooting with an arbitrarily chosen root.*

Combining Theorems 5 and 9, we obtain the following result.

Corollary 10. *Every 2-connected multi-robot network admits a Network Traffic Law which is a useful social law and which can be efficiently synthesized via the rooting algorithm applied to its underlying graph.*

We finish this section with a few remarks on the *cycle time* of our synthesized Network Traffic Law, that is, the time needed of a robot to move from one destination to the next under this Law. Clearly, the cycle time depends on the p-rooting of the underlying graph G from which the Network Traffic Law was synthesized. Specifically, if the number of robots is less than or equal to p and all edge lengths equal a unit then this time is precisely p. For a network with a small number of robots it is therefore desirable to produce a p-rooting with p small. A lower bound on p is obviously provided by $2\delta(G)$, where $\delta(G)$ is the *diameter* of G, that is, the maximum distance between any pair of vertices. This bound is not always attainable: for example, if G is a circuit of length $2p + 1$ then $\delta(G) = p$ but G has no $2p$-rooting. Our Rooting Algorithm above seems to perform quite well. For instance, for the d-dimensional hypercube C_d, the choice made at step 1 of each iteration can be determined so as to produce the following p-rooting f_d with minimum possible $p = 2d = 2\delta(C_d)$, defined, for each vertex $v = (v_1, \dots, v_d) \in \{0,1\}^d$ of C_d, by

$$f_d(v) := (2d-1) \cdot v_d + (-1)^{v_d} \sum_{i=1}^{d-1} v_i$$

(it easily follows from Proposition 7 that this function indeed is a rooting). The case of the 3-cube is described explicitly in Example 6 and Figs. 1–4 above. A study of the cycle time in the general case is under way and will be treated in more detail elsewhere.

Another issue to be addressed concerns situations where the structure of the underlying graph is part of the design. The above discussion and the superiority of graphs with low diameter suggest that a good source of graphs may be provided by certain classes of *polytopal graphs* (see e.g. [8,13,20]).

5. Discussion

The study presented in this paper refers to point robots. Although this may be a reasonable assumption in illustrating the artificial social systems approach, one may wish to relax this assumption. In order to handle this issue, consider the minimal velocity $v_{\min}$ which the algorithm outputs for a particular graph. Assume that the length of each edge is at least one unit and we wish to guarantee a one unit distance between each pair of robots. In this case, if the robots enter the graph in a time offset of $1/v_{\min}$ time units from one another, it is easy to check that a distance of at least one unit between the robots will be preserved. The number of robots on the network should be limited to at most $p \times v_{\min}$ on the worst case, where f is assumed to be a p-rooting. In order to handle these and other related problems in a more complete fashion, and in order to discuss the efficiency of the social laws (which we have partially discussed in Section 4), further study which is tailored for more specific graph structures is needed. In the special case of a graph which can be embedded in a *2-dimensional grid*, such a study is presented in [5,6]. The algorithm presented there enables to obtain a good *competitive ratio* between the time that it takes an agent to obtain a goal (given the social law that the algorithm outputs) and the time it would have taken it to obtain the goal if the agent would have worked in isolation. However, various mobilization settings correspond to 2-connected graphs which can not be embedded in 2-dimensional grids. In addition, our study introduces a polynomial algorithm for the automatic synthesis of social laws, while the latter work introduces an exponential algorithm which is especially useful (and lead to efficient social laws) for graphs which can be embedded in a two-dimensional grid.

There are several approaches to the coordination of physically moving entities. One may consult [16,17,22] for an overview. The social system approach superficially resembles some more classical path planning techniques such as quad-trees and cell decomposition. The similarity stems from the idea of partitioning the environment into regions while allowing particular type of movements inside regions and particular type of movements between regions. However, the artificial social systems approach is directed towards overcoming the problem of coordinating agents activities, which make the techniques used in this context much different from the ones used in classical path planning. The artificial social systems approach is also much different from the more centralized approach to the control of multiple robots as well as from decentralized approaches in this regard (e.g., [4,9]).[3]

[3] In [16] the idea of traffic/social laws is discussed in the context of other techniques for path planning in multi-agent systems. In a centralized approach robots are treated as a single entity that moves in a composite space, while in a decentralized approach each robot treats the others as moving obstacles. Traffic/Social laws enable the robots to behave individually, while avoiding many on-line collision avoidance and liveness problems.

In our study we have assumed that the agents may change their velocity, while ignoring the need for acceleration and deceleration. This is quite typical for work on the theory of AGVs, where agents are assumed to stop and reinitiate their movement without any cost [12]. Nevertheless, our study can take into account acceleration and deceleration by allowing the agents to accelerate and decelerate while taking care that they will not collide in spite of these processes. This can be obtained by having the agents enter the graph with time offset which takes into account the time needed for acceleration and deceleration. The discussion of the exact effects of this change on the number of agents which may enter a graph will depend on the particular structure of the graph, and is beyond the scope of this paper.

The reader should notice that in some cases the social law generated may require from the agents to move in somewhat low speeds, although it will guarantee the achievement of goals. This situation may occur for example where there is a short edge uv in the graph, where the difference between the labels (generated by our algorithm) of u and v is large. This point should be addressed by considering the properties of the rooting algorithm for more particular graphs. The emphasis of our study is on a general automatic procedure for the generation of social laws which guarantee goal achievements and no collisions for the whole family of 2-connected graphs. Notice that in other decentralized approaches, where paths are calculated in a decentralized fashion (e.g., [9]) there is no guarantee that the goals of the agents will be actually obtained. Given that our results are obtained for the case of agents which operate with minimal communication and sensing abilities, we believe our results to be more applicable than previously established results in this regard.

There are various issues which we believe can not be easily handled by the techniques explored in this paper. This includes for example the case where the structure of the environment may change. We see the study of applying the artificial social systems approach to such dynamic contexts as a promising direction for future research. An important point to notice, however, is that once the changes in the system have been detected, we may use the fact that our synthesis algorithm is a very efficient (linear) algorithm in order to generate a new social law. This gives another advantage over previous general techniques for multi-robot path planning, which have been either intractable or tailored for very particular domains.

In this paper we concentrated on mobilization of physical entities in 2-connected graphs, which are the settings where Automated Guided Vehicles (see [12,22]) are discussed. Similar ideas can be used for agent mobilization in other contexts, such as communication networks. Notice that the 2-connectivity assumption is quite popular in such settings as well. Moreover, when the system is not 2-connected the approach is to increase the connectivity of the network in order to yield this (and higher) level of connectivity [10]. Our work adopts the artificial social systems approach for the coordination of multi-agent systems and shows how it can be applied to yield coordinated and useful behavior in the related models. We introduced a general set of environments where agents can achieve their goals while avoiding collisions, using the artificial social systems approach; the assumptions made about each agent's perceptual capability have been minimal. Moreover, we have shown that the corresponding social laws can be efficiently synthesized.

Acknowledgments

The second author wishes to thank Joe Halpern for his encouragement to pursue the work reported in this paper.

References

[1] W. Briggs and D. Cook, Flexible social laws, in: *Proceedings IJCAI-95*, Montreal, Que. (1995) 688-693.
[2] A.H. Bond and L. Gasser, *Readings in Distributed Artificial Intelligence* (Ablex, Norwood, NJ, 1988).
[3] J.A. Bondy and U.S.R. Murty, *Graph Theory with Applications* (North-Holland, Amsterdam, 1976).
[4] S.J. Buckley, Fast motion planning for multiple moving robots, in: *Proceedings 1989 IEEE International Conference on Robotics and Automation*, Scottsdale, AZ (1989) 322-326.
[5] O. Ben-Yitzhak and M. Tennenholtz, On the synthesis of social laws for mobile robots: a study in artificial social systems (Part I), *Computers and Artificial Intelligence*, to appear.
[6] O. Ben-Yitzhak and M. Tennenholtz, On the synthesis of social laws for mobile robots: a study in artificial social systems (Part II), *Computers and Artificial Intelligence*, to appear.
[7] C. Dwork and Y. Moses, Knowledge and common knowledge in a Byzantine environment: crash failures, *Inform. and Comput.* 88 (2) (1990) 156-186.
[8] M. Deza and S. Onn, Lattice-free polytopes and their diameter, *Discrete Comput. Geom.* 13 (1995) 59-75.
[9] M. Erdmann and T. Lozano-Perez, On multiple moving robots, *Algorithmica* 2 (4) (1987) 477-521.
[10] K.P. Eswaran and R.E. Tarjan, Augmentation problems, *Siam J. Comput.* 5 (1976) 653-665.
[11] M.R. Genesereth, M.L. Ginsberg and J.S. Rosenschein, Cooperation without communication, in: *Proceedings AAAI-86*, Philadelphia, PA (1986) 51-57.
[12] D. Grossman, Traffic control of multiple robot vehicles, *IEEE J. Robotics and Automation* 5 (4) (1988) 491-497.
[13] P. Kleinschmidt and S. Onn, On the diameter of convex polytopes, *Discrete Math.* 102 (1992) 75-77.
[14] S. Kraus and J. Wilkenfeld, The function of time in cooperative negotiations, in: *Proceedings AAAI-91*, Anaheim, CA (1991) 179-184.
[15] A.L. Lansky, Localized event-based reasoning for multiagent domains, Tech. Rept. 423, SRI International, Menlo Park, CA (1988).
[16] J.C. Latombe, How to move (physically speaking) in a multi-agent world, in: *Proceedings MAAMAW91* (1991).
[17] J.C. Latombe, *Robot Motion Planning* (Kluwer, Dordrecht, 1991).
[18] Y. Moses and M. Tennenholtz, Artificial social systems Part I: Basic principles, Tech. Rept. CS90-12, Weizmann Institute, Rehovot, Israel (1990).
[19] Y. Moses and M. Tennenholtz, Artificial social systems, *Computers and Artificial Intelligence* 14 (6) (1995) 533-562.
[20] S. Onn, Geometry, complexity and combinatorics of permutation polytopes, *J. Combin. Theory Ser. A* 64 (1993) 31-49.
[21] J.S. Rosenschein and M.R. Genesereth, Deals among rational agents, in: *Proceedings IJCAI-85*, Los Angeles, CA (1985) 91-99.
[22] D. Sinreich, Network design models for discrete material flow systems: a literature review, *International J. Advanced Manufacturing Technology* 10 (1995) 277-291.
[23] Y. Shoham and M. Tennenholtz, On the synthesis of useful social laws for artificial agent societies, in: *Proceedings AAAI-92*, San Jose, CA (1992) 276-281.
[24] Y. Shoham and M. Tennenholtz, On traffic laws for mobile robots, in: *Proceedings 1st Conference on AI Planning Systems (AIPS-92)* (1992).
[25] Y. Shoham and M. Tennenholtz, Social laws for artificial agent societies: off-line design, *Artificial Intelligence* 73 (1995) 231-252.
[26] C.J. Stuart, An implementation of a multi-agent plan synchronizer, in: *Proceedings IJCAI-85*, Los Angeles, CA (1985) 1031-1033.

[27] M. Tennenholtz and Y. Moses, On cooperation in a multi-entity model, in: *Proceedings IJCAI-89*, Detroit, MI (1989).

[28] G. Zlotkin and J.S. Rosenschein, A domain theory for task oriented negotiation, in: *Proceedings IJCAI-93*, Chambery, France (1993) 416–422.