Distributed Strategies for Local Minima Escape in Motion Planning for Mobile Networks

Zhenwang Yao and Kamal Gupta
Robotic Algorithms and Motion Planning (RAMP) Lab,
School of Engineering Science, Simon Fraser University, Canada
{zyao,kamal}@cs.sfu.ca

***Abstract*—This paper studies the problem of controlling networked mobile agents while maintaining connectedness among them. Our previous work in [13] and other existing distributed methods use potential field based techniques and hence suffer from local minimum problems. In this paper we propose a preliminary categorization of different types of local minima that can arise and distributed strategies to deal with these local minima, based on our previous backbone based connectivity control (BBCC) framework in [13], where a communication backbone was used to maintain system connectivity. The local minima in our problem can arise either due to obstacles or due to connectivity constraints, or a combination of the two. Our categorization and the corresponding strategies to escape local minima are an initial attempt to deal with these issues in a systematic manner. As backbone is an effective and efficient representation of the formation topology, it provides a good leverage to exploit all members in the mobile network to gain knowledge of the environment and make decisions, and our simulations show that these backbone based strategies are very effective in escaping these local minima.**

I. Introduction

A network of autonomous mobile robots (agents) has the desirable capability of performing spatially distributed tasks including sensing, coverage, surveillance, exploration, target detection, etc. Such mobile sensor networks have received much attention in last few years. However, controlling such a group of agents is a fundamental and challenging problem. In particular, we study the problem of controlling networked mobile agents while maintaining connectedness among them, i.e., all agents are required to remain connected to each other (either directly, or via other agents). Connectedness is essential in coordinated and cooperative control, since team members need to communicate and share information with each other, and more importantly, in many cases connectedness is a necessary condition for the stability of the system [7], [8].

Connectedness constraints have been considered in the context of different problems, such as formation control [10] consensus problem [6], and flocking and swarming [9], [4], [3]. Many recent works have addressed connectedness constraints in path planning [5], [1] and more general motion control of networked systems [11], [14], [15], [13]. Among these works, [10], [4] assumed a predefined and fixed topology of system (e.g., "constraint graph"), and [6], [3] assumed either the goal formation to be a subgraph of the initial formation, or vice versa. Clearly, fixed topology does not capture full dynamics of multi-agent systems (formations), since formation topology changes as agents move in and out of communication range of each other. Such presumptions on formation topology are quite limiting, are not realistic in many applications, and may prevent certain tasks from being achieved. [1], [5], [14] proposed centralized approaches, and suffer from high dimensionality of the problem for large systems and may not suitable for real time applications. [11], [15], [13] proposed distributed approaches, which use only local information to achieve overall system objectives and hence are intrinsically more desirable. These three approaches use certain sub-graphs to represent system topologies, and guarantee connectedness by maintaining existing links in the representative sub-graphs. Note that it is difficult to embed connectedness constraints into geometric and analytical models typically used in distributed motion control or planning algorithms, due to the combinatorial and global nature of connectedness constraints [11]. The above distributed approaches use potential field based techniques to maintain critical links, and suffer from local minima problem when multiple criteria are considered, such as achieving goal, maintaining connectedness, and avoiding collisions. Furthermore, as mentioned in [11], in some scenarios, using only local information is doomed to failure and global decision needs to be made in order to achieve a certain task. In this paper, we extend our previous work [13] to a general motion planning framework that is capable of escaping from local minima, and making global decisions when necessary. To the best of our knowledge, ours is the first distributed approach to attack the local minimum problem in mobile networks.

In [13], we proposed a framework for motion planning with connectedness constraints, called BBCC, Backbone Based Connectivity Control. *Backbone* is a concept we borrowed from communication literature [2]. It is a virtual network formed by a relatively small subset of the network, and provides a hierarchical organization of the original network. The basic idea of forming a backbone is to group a set of agents based on physical proximity, and represent each group by a single agent as *clusterhead*; clusterheads are connected to each other, or via connecting agents called *gateways*. Clusterheads, gateways and selected connections among them form the backbone. Clusterheads are chosen in a way such that all other agents connect to at least one clusterhead (i.e., clusterheads form a dominating set; for more details please refer to [13]). In order to maintain the connectedness of the system, we first maintain a connected backbone, by maintaining existing connections (communication links) in the backbone; and then for a non-backbone agent, one of the backbone agents is chosen as a leader, and the non-backbone agent maintains connection to the leader by following it. The key advantages of using backbone are that it captures the system connectivity nicely, and it can

ROBOCOMM 2009, 31st Mar – 2nd Apr 2009, Odense, Denmark.
 ISBN 978-963-9799-51-6
DOI 10.4108/ICST.ROBOCOMM2009.5856

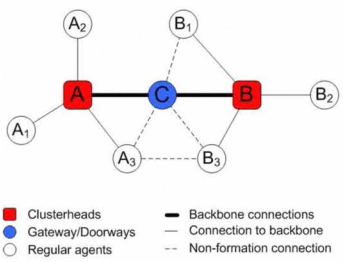
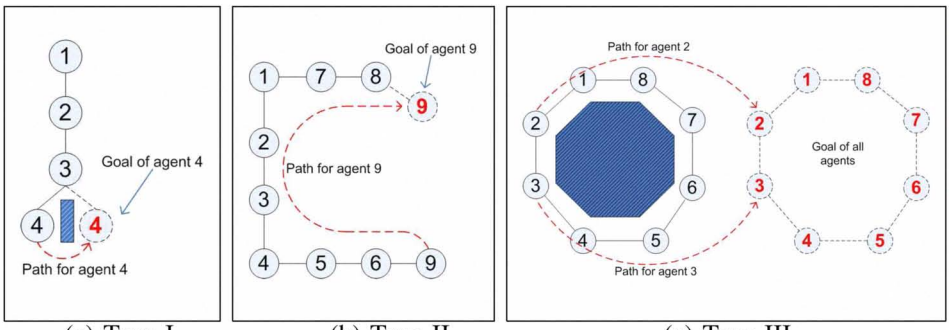


Fig. 1. Backbone-based formation.



(a) Type I. (b) Type II. (c) Type III.

Fig. 2. Categories of local minimum. Dotted nodes are goal positions of corresponding agents

be constructed efficiently in a distributed manner.

In this paper, we bring local minimum avoidance and global decision making capability into the BBCC framework. Local minima are detected when one or more agents do not progress toward their goals. We classify local minima into three different categories: Type-I (Regional obstacle-induced local minimum), Type-II (Individual connectivity-induced local minimum), and Type-III (Structural compound local minimum). As we will detail later, different types imply different natures of the minima. In the first category, the agent may be able to escape the minimum merely by simple local behavior (e.g.,random walk), whereas in latter two categories, an agent needs help from others in order to make global decisions for local minimum escaping. Corresponding to these different types, three respective strategies are used to tackle these local minima: Random Walk, Backbone-based Navigation, and Backbone-based Leader-following. The first strategy is simply a local strategy for obstacle avoidance, and the latter two strategies incorporate distributed global decision making and exploit existing backbone constructed under BBCC toward this purpose. Backbone based Navigation strategy uses the backbone to take advantage of the knowledge (sensing) embedded in the entire system, gathers path planning information (roadmap) that is beyond sensing and communication range of one single agent, and provides guidance to agents to escape Type-II local minimum. Backbone based *Leader-following* strategy tries to achieve maximum mobility by reducing the number of connectedness constraints, and looks for maximum reconfigurability in order to escape Type-III local minimum.

The organization of the paper is as follows. The formal problem formulation is given in Section II. After that we briefly review our previous BBCC framework in Section III, outline the proposed local minimum escaping scheme in Section IV, and then detail distributed local minimum escaping strategies in Section V. In Section VI, we show computer simulation results, followed by conclusions in Section VII.

II. Problem Formulation

Consider a group of n mobile agents, $\mathcal{A} = \{1, 2, \cdots, n\}$, and their positions are denoted as $\mathcal{X}(t) = \{x_1(t), x_2(t), \cdots, x_n(t)\}$. Agents know their own positions, and can communicate with each other within a communication range d_c. We model interaction of the group as a time-varying proximity graph (also called communication graph), $G(t) = \big(V(t), E(t)\big)$, whose vertices represent agents, $V(t) \equiv \mathcal{A}$, and edges represent communication links between agents. Let (i, j) denote an edge between agents i and j, thus,

$$(i, j) \in E(t) \Longleftrightarrow d(i, j) = \|x_i(t) - x_j(t)\| \leq d_c$$

Each agent, i, can have its own goal configuration, x_i^g, so $\mathcal{X}_g = \{x_1^g, x_2^g, \cdots, x_n^g\}$. The problem is how to maneuver the group of agents to reach their respective goals, with the constraint that $G(t)$ remains connected throughout the task.

III. Existing Backbone Based Connectivity Control (BBCC)

The *BBCC* proposed in [13] works as follows.

1) With the communication graph, $G(t)$, BBCC first constructs the backbone, $G_B(t) = \big(V_B(t), E_B(t)\big)$, where $E_B(t) \subseteq E(t)$ and $V_B(t) \subseteq V(t)$, in a distributed fashion. The backbone consists of backbone agents and connections among them, and these agents and connections are critical for the system connectivity. Fig.1 shows an example of backbone in a formation. There are 3 backbone agents: A, B, and C, where A, B are clusterheads, and C is a connecting gateway. A_i's are non-backbone agents associated with A, and A along with A_i's forms one cluster. B_i's are non-backbone agents associated with B, and they form the second cluster.
2) Based on the constructed backbone and respective goals, motion of each agent, $\dot{x}_i(t)$, is determined. For the backbone agents, we formulate the backbone as a constraint graph, and motion control is derived such that every connection in backbone is maintained; and for non-backbone agents, we use, loosely speaking, a sort of leader-follower control with the associated backbone agent as the leader. In Fig.1, to guarantee connectedness, connections (A, C), (B, C) are preserved, and within its own cluster, A is the leader and $A_i's$ maintain connections to A. Same for B and B_i's.
3) After agents move with constraints of backbone for a period of time, T, communication graph $G(t)$ may have changed, and thus the backbone $G_B(t)$ is updated, and agents move with the new backbone as constraint graph. We stress that BBCC is a distributed framework, there is no central global representation of $G(t)$, $G_B(t)$, and update of $G(t)$, $G_B(t)$ and $\dot{x}_i(t)$ are done locally.

Digital Object Identifier: 10.4108/ICST.ROBOCOMM2009.5856
http://dx.doi.org/10.4108/ICST.ROBOCOMM2009.5856

When computing agent motion, $\dot{x}_i(t)$, we model the goal achievement and connectedness maintenance as attractive potentials, and obstacle avoidance as a repulsive potential. The agents then follow the negated gradient of the composite potential. As expected, such composite potential may have local minima, and some agents or the entire team may get stuck. We now extend the above framework to deal with local minima, by introducing a local minimum escaping scheme.

IV. Overview of Proposed Local Minimum Escaping Scheme

A. Local minimum detection

To detect local minima, we keep track of an agent's trajectory. We save the last K positions of the agent, and compute their variance. If the variance is smaller than a threshold, then this agent is deemed to be in a local minimum.

B. Local minimum classification

We categorize local minima into three different levels:

1) **Type I: Regional obstacle-induced local minimum**. This is mainly because of obstacles, and connectedness constraints are not the limiting factor (i.e., they are easily maintained in that region). Intuitively, this is what may occur for a single robot. A local strategy such as random walk may solve the problem. Fig. 2(a) shows one such example, where a small obstacle blocks agent #4 from its goal, and a random walk should be enough for the agent to get around the obstacle and reach its goal.
2) **Type II: Individual connectivity-induced local minimum**. This type of local minimum is normally caused by connectedness constraints, and may or may not be compounded with obstacles. In such a scenario, only a single agent (or small number of agents) is in local minimum. Simple local strategies may not be possible for the agent(s) to escape, and instead, global path planning is needed. For example, in Fig. 2(b), in order for agent#9 to reach its goal position, the agent has to move along the formation to maintain connectedness of the formation, and finding such a path needs to involve all agents.
3) **Type III: Structural compound local minimum**. This type of local minima is also caused by connectedness constraints, and is most likely compound with obstacles. A number of the team agents are stuck into local minima, and even worse these local minima are coupled to one another. A major reconfiguration of the entire mobile network is needed in order to move out of these minima. Fig. 2(c) is an example. Clearly in this case, from each agent's local point of view, all edges in the initial formation are critical in maintaining connectedness. In order to achieve the goal formation, some edge has to be broken ($(2,3)$ in the example); and making decision as to which edge to break needs to involve all agents.

Although given examples illustrate different types of local minima, the distinctions may not be as clean cut. There is ample scope for investigating these further in a more formal manner. We determine which type a local minimum belongs to in a heuristic way, and adopt different strategies to deal with local minima. We now discuss the heuristics and these strategies.

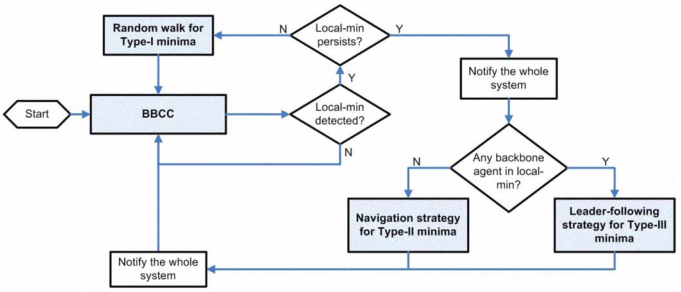


Fig. 3. Overall scheme.

C. Local minimum escaping strategies

The overall local minimum escaping scheme is shown in figure 3. It starts with the original BBCC; Agents update backbone and move subject to connectedness constraint based on the backbone. When an agent detects a local minimum, it first assumes the minimum is a Type-I minimum, and uses some simple local strategies, we used random walk (could easily be substituted with others) to escape. If a local minimum persists, the minimum is deemed to be either Type-II or Type-III minimum, and the difference between the two is whether or not agent (in the local minimum) is a backbone agent. For either type, the agent needs helps from all other agents, so the local minimum agent broadcasts its information to notify all other agents. Upon receiving such notifications, all agents temporarily stop moving, and the backbone becomes *stationary*. Then, the escape strategies are as follows. For Type-II minimum, since no backbone agent is in local minimum, current stationary backbone is used as a navigation roadmap to navigate the (non-backbone) agent to get as close as possible to its goal. We call this strategy "Backbone-based Navigation". For a Type-III minimum, the system constructs a spanning tree based on the stationary backbone. The root agent of the tree is the agent that is farthest away from those agents deemed to be at local minima (and hence most likely to be free from local minimum), and acts as a leader. All other agents follow this leader according to the tree hierarchy and move toward their parents instead of their ultimate goals. The purpose of doing this is to bring agents closer, increase the system connectivity, and thereby increase reconfigurability. We call this strategy "Backbone-based Leader-following". After the escaping procedure, the system resumes to BBCC mode.

It is possible that more than one agent are simultaneously at their respective local minima, and these are handled as follows. A Type-III minimum has the highest priority to be solved, followed by a Type-II, followed by a Type-I minimum. Each strategy handles multiple local minima of the same type. For multiple Type-I minima, agents can random walk at the same time. If there are multiple Type-II minima (but no Type-III minimum), the Backbone-based Navigation strategy creates routes to each minimum agent, i.e., they are simultaneously handled. The Backbone-based Leader-following strategy naturally handles multiple Type-III minima by selecting the leader to be farthest away from *any* local minima.

Digital Object Identifier: 10.4108/ICST.ROBOCOMM2009.5856
http://dx.doi.org/10.4108/ICST.ROBOCOMM2009.5856

V. Distributed Escaping Strategies: Details

For Type-I minima, agents adopt random walks "on the go". While other agents are still moving toward their respective goals, an agent deemed to be in Type-I local minimum moves toward randomly chosen subgoals (for the next R time steps). Certainly, all agents are still subject to connectedness constraints when they move.

Type-II and Type-III minima need global assistance to escape, and thus an agent in either type of local minimum notifies all other agents by sending out a *MSG_DETECT* message, with its information, such as its current and goal position. All agents maintain a list of agents in local minimum, $lmList$. An agent receives the *MSG_DETECT* message, stores the corresponding local minimum information into the list, and forwards the message (if needed). At the end, all agents will have the exact same list. To initiate Backbone-based Navigation or Leader-following escaping procedure, upon receiving the *MSG_DETECT* message, all agents stop moving and thus backbone become stationary; To reflect the topology of the stopped network, agents stop updating the backbone during escaping (i.e.,the backbone is "frozen", as we call it later).

A. Escaping Type-II minima

A backbone-based robot navigation scheme was proposed in [12] for single robot navigation in a static sensor network. Therein a shortest path from current robot location to a given goal sensor node is computed based on the backbone. Please refer to the paper for detailed information about the navigation scheme. Here, we use a simplified version of the scheme, since the backbone is already constructed by *BBCC*. Denote A_m as the non-backbone agent that is in local minimum, and define a cost function of agent i, to be the distance between agent i and the goal of A_m, x_m^g.

$$C(A_i, A_m) = \|x_i - x_m^g\| \tag{1}$$

The basic idea is to propagate a navigation field (with $C(A_i, A_m)$ as the navigation function) over the (stationary) backbone, then the local minimum agent follows the field in the descending direction. Details are given below.

1) Subgoal election: For an A_m, the backbone agent with the smallest cost will be chosen as its subgoal. This is a global election, and involves all agents. A backbone agent proposes itself to be a subgoal candidate if it has smaller cost (i.e., closer to x_m^g) than any of its backbone neighbors. Note that the proposal is only based on an agent's (2-hop) neighbor information, and it is possible that there are more than one agent assuming itself to be the sub-goal. To reach a global decision, the planning step follows.
2) Planning: Candidates broadcast a *MSG_PLAN_NBB* message with its own information (including position). An agent receives the message and stores the route to the candidate. An agent may receive more than one message, and if the received message gives a better (smaller-cost, and shorter) path to the goal of A_m, the agent updates the route, and forwards this message if the receiving agent is a backbone agent. This procedure is similar to the goal dissemination procedure as in [12], except that we may have multiple sources (i.e., multiple subgoal candidates) here. At the end of the procedure, all agents come to a unified conclusion of who is the winning subgoal for A_m, and every agent has the best route to the subgoal. Please note that a backbone agent only processes *MSG_PLAN_NBB* messages from backbone agents it connects to, and simply discards the messages from any other agents; A non-backbone agent receives and processes *MSG_PLAN_NBB* messages but never forwards the message.
3) Navigation: After previous planning step, all agents store the best route to the subgoal of the local minimum agent, so they can provide guidance (to the local minimum agent) regarding the best movement toward its subgoal. To escape from the local minimum, the agent constantly broadcasts a query to the backbone; backbone agents respond with next via-point based on stored routes; the escaping agent chooses the best next via-point as "sub-sub-goal", and move toward that. Such query-respond-move procedure repeats until the subgoal is reached.
4) Back to BBCC: After the agent has reached its subgoal, it broadcasts a *MSG_ESCAPED* message to notify all agents that it is out of local minimum. Agents receive the message and remove the agent from $lmList$. Once all agents have reached their subgoals, the system resumes to BBCC. If one or more of the agents are unable to reach their respective sub-goals, the strategy simply reports a failure. We are currently exploring more sophisticated strategies in such cases.

B. Escaping Type-III minima

For a Type-III minimum, the system uses Backbone-based Leader-following strategy. The basic idea is to construct a leader-following tree hierarchy with a leader that is most likely free from local minima, and then move closer toward the leader to increase connectivity and hence reconfigurability. Define a gain function of agent i, as its distance to the local minimum agent A_m.

$$G(A_i, A_m) = \|x_i - x_m\| \tag{2}$$

The Leader-following escaping procedure then includes the following steps.

1) Leader election: The backbone agent that has maximum gain (i.e., farthest from *any* local minimum agent) is elected as the leader. Similar to subgoal election in previous Navigation strategy, leader election involves all agents. A backbone agent proposes itself to be a leader candidate if it has bigger gain than any of its backbone neighbors.
2) Spanning tree construction: Leader candidates broadcast a *MSG_PLAN_BB* message with its gain. An agent receives the message, and checks if the received message gives a better (with larger gain, and shorter route) leader. If so it updates its route to the new leader, and if the receiving agent is a backbone agent, it forwards the message. At the end, all agents have routes to the winning leader (A_l) with

Digital Object Identifier: 10.4108/ICST.ROBOCOMM2009.5856
http://dx.doi.org/10.4108/ICST.ROBOCOMM2009.5856

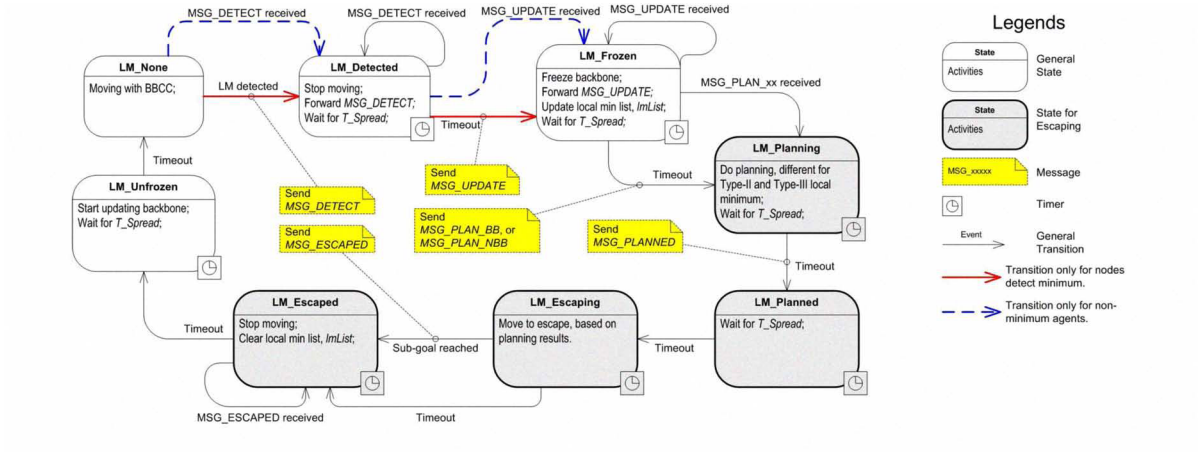


Fig. 4. State machine implementation.

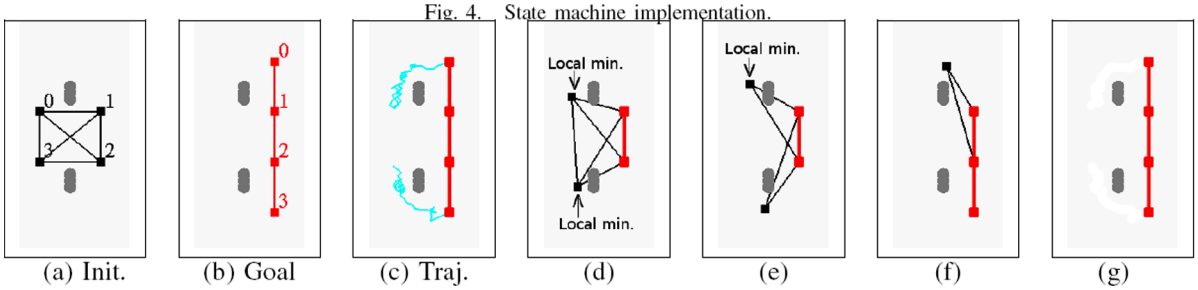


Fig. 5. Escaping from Type-I minimum. (a) Initial formation. (b) Goal formation. (c) Successful trajectories of agent #0 and #3 with Random Walk strategy. (d)-(g) Snapshots of the system along the successful trajectories. (d) Local minima detected by agent #0 and #3. (e) Agent #3 escaped the minimum, while agent #0 detected another minimum.

the highest gain. These routes make a tree hierarchy with A_l as the root. Note that since during the construction, only backbone agents forward the message, the resulting spanning tree has all non-backbone agents as leaves.

3) Leader-following: Once the spanning tree is constructed, agents move in a leader-following fashion. An agent follows its parent in the spanning tree, and move toward its parent, instead of toward its ultimate goal.
4) Back to BBCC: After the (vertex) connectivity of the local minimum agent has increased by a certain degree (a user-defined parameter in the algorithm), or simply after moving for a certain period of time, the system stops and resumes to the regular BBCC to move to original goal.

C. Implementation details

We have introduced general ideas of the framework, and skipped some important technical implementation details for clarity. We present these details in this section. Fig. 4 shows our state machine design for the proposed local minima escaping scheme. The same state machine runs on all agents, but it may take different transitions on different hosts depending on whether the host is a backbone or non-backbone agent, and whether it is an agent in local minimum.

1) Synchronization: In the framework, the system may switch from BBCC moving mode to Navigation escaping mode, or to Leader-following escaping mode, and then switch back to BBCC. In different modes, agents move with different constraints. Switching between modes needs synchronization, since we need to make sure agents move with proper constraints engaged, otherwise the system may became disconnected. To synchronize switching, we introduce some intermediate states/modes, and some extra messages. The implementation of an agent's state machine is shown in Fig. 4. While backbone construction involves only local agents and generates $O(\Delta)$ messages for each of backbone agents [12], where Δ is the maximum vertex degree, synchronization messages (including all messages shown in the Fig.4) need to reach all agents, and generate $O(n)$ messages for each round of synchronization, where n is the number of mobile agents.

a) Freezing and unfreezing the backbone: Once an agent detects a Type-II or Type-III minimum, it stops moving and broadcasts *MSG_DETECT* message, and agents receiving the message also stop moving. However, agents should not stop updating backbone while any agent is still moving. As a consequence, the backbone may have changed since the time the local minimum is first detected. Therefore, we introduce two states, *LM_Detected* and *LM_Frozen*, and an extra message, *MSG_UPDATE* to synchronize the backbone. When an agent first detects a local minimum, or receives the *MSG_DETECT* message, it stops, transits into *LM_Detected* state, and waits for *T_Spread* seconds (the maximum time it may take for a message to spread to every agent). Assuming after this wait all agents have stopped and backbone has been stationary, the local minimum agent sends a *MSG_UPDATE* message after timeout to freeze the backbone, and update the $lmList$ in each agent. Similarly, when the system resumes from escape mode to BBCC, agents should start updating the backbone before they can start moving. The *LM_Unfrozen* state serves this purpose and makes sure the backbone is updated by waiting for *T_Spread* seconds before transitioning back to *LM_None* state (BBCC).

Digital Object Identifier: 10.4108/ICST.ROBOCOMM2009.5856
http://dx.doi.org/10.4108/ICST.ROBOCOMM2009.5856

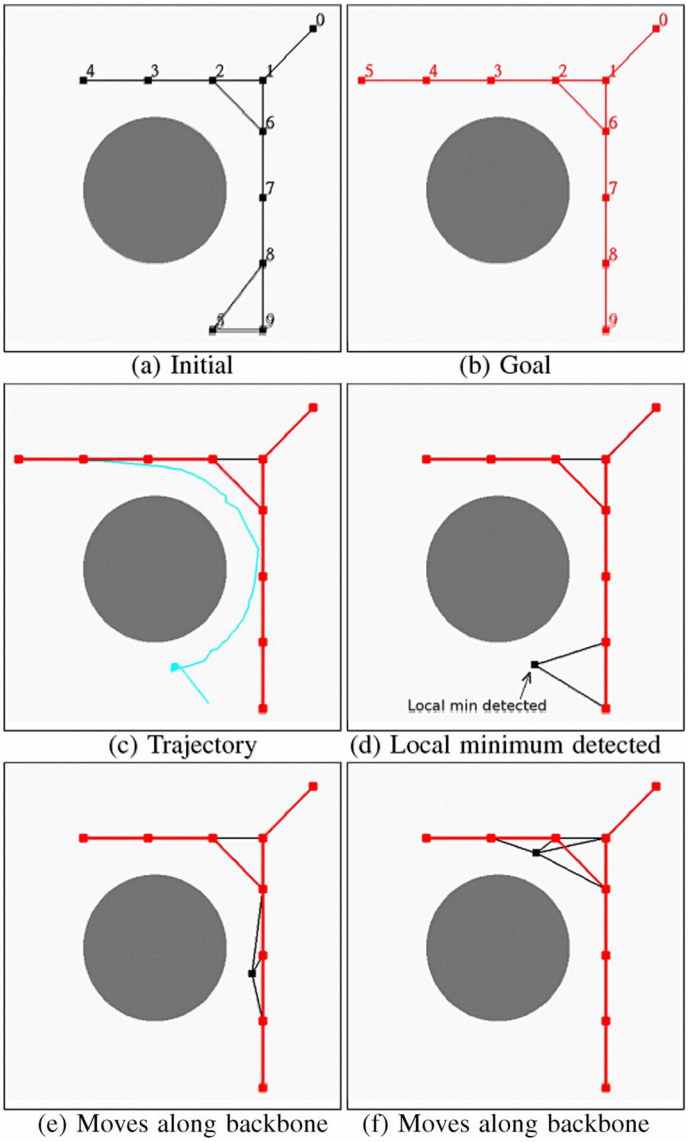


Fig. 6. Escaping from Type-II minimum. (a) Initial formation. (b) Goal formation. (c) A successful trajectory of the local minimum agent, with our Backbone-based Navigation strategy. (d) A local minimum detected. (e)-(f) Snapshots of the system along the successful trajectory.

b) Planning and spanning tree construction: In *LM_Planning* state, agents do path planning or spanning tree construction depending on what type of local minimum the system is dealing with. Once the path has been planned, or the spanning tree has been constructed, all agents transition into *LM_Planned* state where escaping agents can prepare for escape. Specifically, in Leader-following escape mode, the spanning tree (instead of the backbone) is used as connectedness constraints, and should be engaged in *LM_Planned* state.

VI. Computer Simulations

We now present simulations to show the effectiveness of the proposed scheme in escaping local minima. We used an in-house developed software simulator. We simulate a team of agents moving, in an arena of $120m \times 120m$, from an initial formation to a goal formation. The communication range between agents is set to 15m. In this paper, we assume a simplified communication model: two agents can communicate with each other if and only if their distance is within the communication range. We assume obstacles do not obstruct communication (e.g., in wireless communication networks).

A. Escaping Type-I minimum

Fig. 5 shows a case of Type-I minimum, where two small obstacles blocked agent #0 and #3 from their goals. While the agents were trying to reach their goals, the obstacles kept pushing them away, and therefore local minima were detected around the obstacles. After detecting that they are in local minima, agents tried random walk to avoid the obstacles, and after random walk the agents moved toward their respective goals again. Due to the random nature, it might take several rounds of random walk to escape the minimum. For example, in the shown simulation, it took agent #0 longer to escape. Other deterministic local strategies may yield better performance.

B. Escaping Type-II minimum

Fig. 6 shows a case of Type-II minimum, where all agents were right at their goal position, except one, agent #5, as shown. It tried to move toward its goal position, but the combination of obstacles and connectedness constraints prevented it from doing so, and the agent detected local minimum as shown in (d). After trying random walk for several times, the local minimum persisted, since the obstacle was relatively large. As agent #5 was a non-backbone agent, the local minimum was deemed to be Type-II, and the Backbone-based Navigation strategy was activated for escaping. To navigate agent #5 out of its local minimum, the existing backbone (bold colored lines and agents) was used as planning roadmap, and agent #4 was elected as subgoal for #5. In from (d) to (f), agent #5 moved along the found path, and reached the subgoal. Clearly, from there it could easily reach its final goal.

C. Escaping Type-III minimum

Fig. 7 shows a case of Type-III minimum, where all agents were wrapped around an obstacle in the initial configuration, and the obstacle was so large that an agent could only communicate with its immediate neighbors (a). In this case, all agents were in the backbone (c), because from every agent's local point of view, all its edges were critical in maintaining connectedness. The goal formation was a complete graph away from the obstacle (as in (a)). In order to achieve the goal, the team had to break some links between agents. When the team tried to move toward the goal formation, connectedness constraints kept the agents from moving any further as in (d). Clearly random walk did not help much in this case, and the system detected a Type-III minimum, as a backbone agent (agent #1 as shown) was in local minimum, and hence the Backbone-based Leader-following strategy was engaged. With the strategy, a spanning tree was constructed, as shown in (e). The root of the spanning tree (the leader) was the backbone agent that was farthest away from the local minimum. Then for a certain period of time, all agents moved and followed this leader, resulting in formation in (g), and from there the system easily reached the goal formation.

Digital Object Identifier: 10.4108/ICST.ROBOCOMM2009.5856
http://dx.doi.org/10.4108/ICST.ROBOCOMM2009.5856

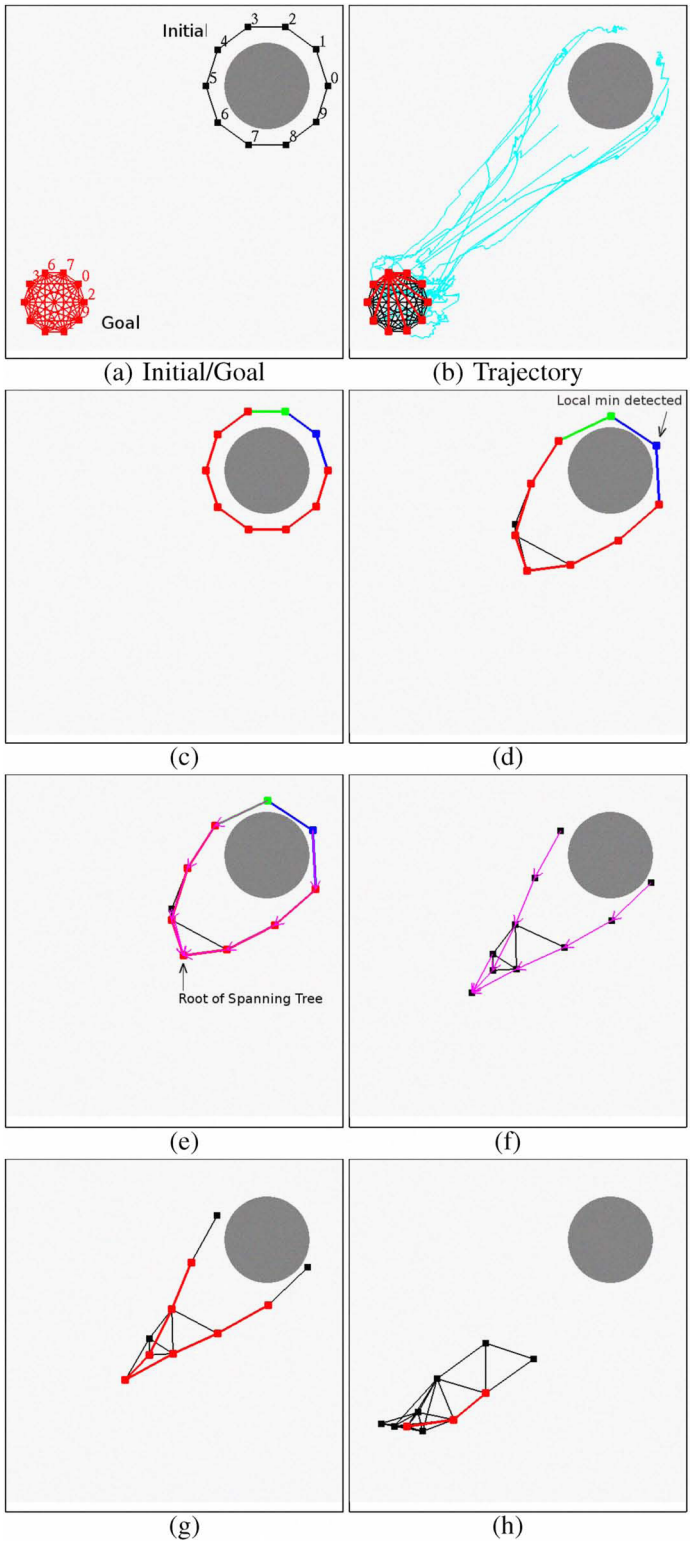


Fig. 7. Escaping from Type-III minimum. (a) Initial and goal formations. (b) Successful trajectories of all agents, with Backbone-based Leader-following strategies. (c)-(h) Snapshots of the system along the successful trajectories. (c) Backbone of initial formation, and all agents were in the backbone. (d) A local minimum detected. (e) Spanning tree was constructed. (f-g) Moving in leader-following mode. (h) Back to BBCC.

VII. Conclusion

We have proposed distributed local minimum escape strategies for motion planning with connectedness constraint for mobile networks. The strategies leverage the backbone constructed from our earlier proposal, Backbone Based Connectivity Control. Backbone-based Navigation strategy is adopted for non-backbone agents in local minimum (i.e., a Type-II minimum); then the backbone is used as a roadmap to navigate agents deemed to be in local minimum, to move toward its goal. Backbone-based Leader-following strategy is used when a backbone agent is in local minimum (i.e., a Type-III minimum); then a spanning tree hierarchy, based on the backbone, is established among agents, and agents follow the hierarchy and move closer to each other for reconfiguration. We showed, via computer simulations, that the proposed strategies are effective in escaping local minima. Our next step is to implement the proposed scheme on a real system. As a first attempt to attack the local minimum problems, our local minima classification and strategies to deal with them are somewhat heuristic based. We are looking into more systematic ways to treat them.

Acknowledgment

The research is supported by Natural Sciences and Engineering Research Council of Canada (NSERC) PGS-D.

References

[1] F. Coutinho, J. Barreiros, and J. Fonseca. Choosing paths that prevent network partitioning in mobile ad-hoc networks. In *IEEE International Workshop on Factory Communication Systems*, pages 65–71, 2004.

[2] B. Das and V. Bharghavan. Routing in ad-hoc networks using minimum connected dominating sets. In *IEEE International Conference on Communications*, volume 1, pages 376–380, 1997.

[3] D. V. Dimarogonas and K. J. Kyriakopoulos. Connectedness preserving distributed swarm aggregation for multiple kinematic robots. *IEEE Trans. on Robotics*, 24(5):1213–1223, 2008.

[4] J. M. Esposito and T. W. Dunbar. Maintaining wireless connectivity constraints for swarms in the presence of obstacles. In *ICRA-2006*, 2006.

[5] Alex Fridman, Jay Modi, Steven Weber, and Moshe Kam. Communication-based motion planning. In *41st Annual Conference on Information Sciences and Systems*, pages 382–387, 2007.

[6] Meng Ji and M. Egerstedt. Distributed coordination control of multiagent systems while preserving connectedness. *IEEE Trans. on Robotics*, 23(4):693–703, 2007.

[7] G. Lafferriere, J. Caughman, and A. Williams. Graph theoretic methods in the stability of vehicle formations. In *Proceeding of ACC*, pages 3729–3734, 2004.

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

[9] R. Olfati-Saber. Flocking for multi-agent dynamic systems: algorithms and theory. *IEEE Trans. on Auto. Control*, 51(3):401–420, 2006.

[10] G. A. S. Pereira, A. K. Das, V. Kumar, and M. F. M. Campos. Decentralized motion planning for multiple robots subject to sensing and communication constraints. In *Multi-robot Systems: From Swarms to Inelligent Automata*, volume 2, pages 267–278, 2003.

[11] D. P. Spanos and R. M. Murray. Motion planning with wireless network constraints. In *Proceeding of ACC*, pages 87–92, 2005.

[12] Zhenwang Yao and Kamal Gupta. Backbone-based roadmaps for robot navigation in sensor networks. In *Proceeding of ICRA*, pages 1023–1029, May 2008.

[13] Zhenwang Yao and Kamal Gupta. Backbone-based connectivity control for mobile networks. 2009. To appear in ICRA-2009.

[14] M. M. Zavlanos and G. J. Pappas. Potential fields for maintaining connectivity of mobile networks. *IEEE Trans. on Robotics*, 23(4):812–816, 2007.

[15] Michael M. Zavlanos and George J. Pappas. Distributed connectivity control of mobile networks. In *Proceeding of CDC*, pages 3591–3596, 2007.

Digital Object Identifier: 10.4108/ICST.ROBOCOMM2009.5856
http://dx.doi.org/10.4108/ICST.ROBOCOMM2009.5856