Robot Task Allocation based on Greedy-Face-Greedy Algorithm

Jelena Stanulovic, Nathalie Mitton, Ivan Mezei

Robot Task Allocation based on Greedy-Face-Greedy Algorithm

Jelena Stanulovic, Nathalie Mitton *Inria*, and Ivan Mezei, *Senior Member, IEEE*

Abstract **—** Two new algorithms (GFGF1 and GFGF2) for event finding in wireless sensor and robot networks based on the Greedy-Face-Greedy (GFG) routing are proposed in this paper. The purpose of finding the event (reported by sensors) is to allocate the task to the closest robot to act upon the event. Using two scenarios (event in or out of the network) and two topologies (random and random with hole) it is shown that GFGF1 always find the closest robot to the event but with more than twice higher communication cost compared to GFG, especially for the outside of the network scenario. GFGF2 features more than 4 times communication cost reduction compared to GFG but with percentage of finding the closest robot up to 90%.

Keywords **— face routing, GFG routing, greedy routing, wireless sensor and robot networks**

I. Introduction

Wireless sensor and robot networks (WSRN) emerged from the wireless sensor networks when some of the network nodes gained additional features (e.g. mobility, actuation etc.). One of the main challenges is to find the optimal robot to do the task - task allocation problem (also known as task assignment problem) [1]. Since the communication aspects of task allocation are not negligible, centralized solutions have a lot of drawbacks (e.g. high communication overhead, low responsiveness etc.). Since in real world applications robots cannot be modeled by a complete graph (i.e. every robot is within communication range of every other robot), there are still ways to improve the task allocation problem in WSRN.

In this paper we explore the WSRN scenario in which sensors get the information about the event (e.g. fire) and a robot is supposed to react upon event. The wireless robot network is depicted in Fig. 1. The question is which robot is the best to react. This problem is formulated as a Multi-robot task allocation problem and there are several instances of this problem and several solutions proposed so far [2].

We assume that the robot network is connected but not forming a complete graph (i.e. robots communicate using multi-hop messaging) and the location of the event is known and reported to the collecting robot (using e.g. LPWAN technology [3]). This is robot S in Fig. 1. Applying the Greedy-Face-Greedy (GFG) [4] algorithm in robot network, the new idea is to find nearby robot(s) depending on the chosen scenario. This scenario is depicted in Fig. 1 by the curve surrounding the event. By this scheme, robots do not activate energy consuming motors to move until one of them is chosen.

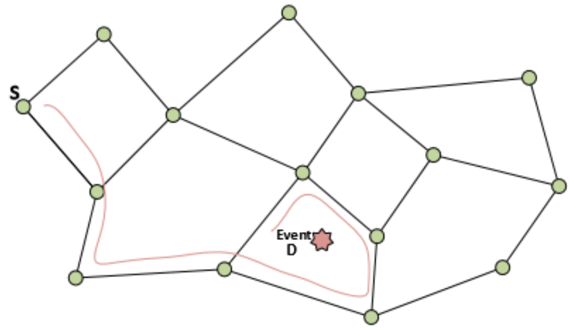

Fig. 1. Applying GFG in robot network

In this paper we present two algorithms that we designate as GFGF1 and GFGF2 where GFGF stands for Greedy-Face-Greedy-Find. In GFGF1 the routing starts by applying the Greedy algorithm: The starting node S forwards the message to the neighboring nodes and selects as a relay node the one that is the closest to the event destination D. If all the neighbors of current node are examined and there is no closer one, then the next step of the search is called the "Face" step. The routing is continued by examining the nodes on the face that are closer to the destination. This greedy-face approach is repeated until the destination is reached. In our scenarios the destination of an event is not part of the network, so it will never be reached. But with this algorithm, the closest node will always be found, as well as the face where D is surrounded by closest robots (i.e. first neighbors). In GFGF2 the communication costs are lowered for cases where outer face is explored incurring high communication overhead.

For the simulations random generated networks are used. Topologies with fully randomized, and networks with circle holes in the middle of the network are used. To work with planar graphs, all the networks are transformed to Gabriel graphs.

The contribution of this paper is two new algorithms based on well-known GFG algorithm applied to the new challenge - finding nearby robot(s) to the event location and with lowered communication overhead. GFGF1 algorithm guarantees that all nearby robots to the event can be found for all scenarios with additional communication overhead compared to GFG. GFGF2

Jelena Stanulović, Intel Austria Gmbh, Europastrasse 8, 9500 Villach, Austria (phone +43424290087134 email: jelena.stanulovic@intel.com), University of Novi Sad, FTN, Trg Dositeja Obradovića 6, 21000 Novi Sad, Serbia

Nathalie Mitton, Inria,. Email: nathalie.mitton@inria.fr

Ivan Mezei, University of Novi Sad, FTN, Trg Dositeja Obradovića 6, 21000 Novi Sad, Serbia (phone: +381-21-485-2552, e-mail: imezei@uns.ac.rs)

features additional communication overhead reduction.

In the following sections we will present the basic ideas of GFG routing (Section II), and proposed algorithms GFGF1 and GFGF2 (Section III). This will be followed by the results and discussion (Section IV).

II. GFG ALGORITHM

Greedy routing is the simplest form of routing first proposed in [5]. In greedy phase of GFG algorithm, each node is forwarding the message to the node which is the closest to the destination, among all its neighbors. Only the neighbors which are closer to the destination are considered.

Greedy routing forwards until the packet reaches a node such that all its neighbors are further from the destination than the node itself. The face routing is then applied until the packet reaches another node that is strictly closer to the destination. The greedy algorithm is then resumed. The algorithm can switch between greedy and face modes several times, but guarantees progress and delivery because face routing is always successful, and loops cannot be created since the algorithm always advances in greedy mode, and is guaranteed to further advance while in the face mode, that is, it is guaranteed to recover. Review of the properties of various GFG variations is given in [7].

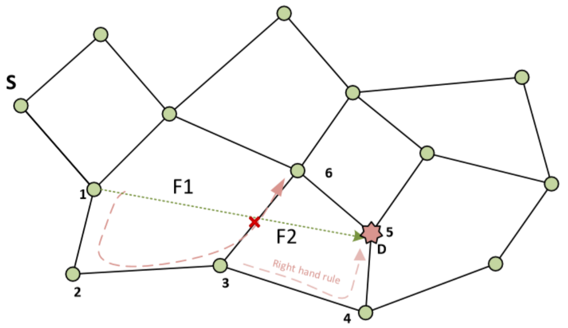

Fig 2. Algorithm GFG, right hand rule example

When the greedy part of GFG cannot find the closest relay node, the face routing phase of GFG begins. This would correspond to the routing from node S to node 1 in Fig. 2. Face routing (first mentioned in [6]) advances based on the intersection between a face and a straight line which connects the last routed node (at the beginning it is S) and destination D. The last routed node is chosen before or after the intersection depending if the before-crossing or after-crossing method is used. A packet is routed along the interiors of the faces until an edge on the route intersects XD between X and D. In Fig. 2, from node 1 is starting the face routing phase and with the right-hand rule the message is forwarded further to nodes 2 and 3. The routing would continue to node 6, but since the intersection with the line 1D occurs, face is changed from F1 to F2 and the routing continues to node 4, and then along the face to 5 and 6. After the change of face, it continues until the D is reached. The boundary of any face can be traversed by applying the right-hand rule (counterclockwise traversal) or the left-hand rule (clockwise traversal). In the right-hand rule the packet is forwarded along the next edge counterclockwise from the edge where it arrived.

III. NEW GFGF ALGORITHMS

In this section, new proposed GFGF algorithms are described. GFG algorithm applied to the scenario where destination D (event location) is not part of the network We call this algorithm GFGF1.

In the GFGF1 algorithm, the stop criterion is defined as the moment when routing is passing the same edge that has already been visited in the same face. The algorithm works as follows:

GFGF1:

1. Follow greedy until delivery or failure
2. **If** (failure) **then**
   - Search next node on the face (by choosing one among the neighbors of the node, based on right-hand rule
   - **While** dist(nextNode, D) > dist(addr, D)
     - Find next node
     - **If** (same edge seen) **then** finish the search
3. **Repeat** steps 1 and 2 until D or same edge visited

Corollary 1. *GFGF1 always finds the closest robot to respond in connected networks.*

Proof:

There are two scenarios – when D is inside of a face in the network (D1) and when D is outside of the network (D2, see Fig. 3). Since GFG is a loop-free algorithm with guaranteed delivery [7] it will traverse the face surrounding D1 (dashed red line in Fig. 3) as well traversing outer face in case of D2 (dotted blue line in Fig. 3). Accordingly, all nodes that are close to D1 or D2 will be traversed and thus the closest could always be found.

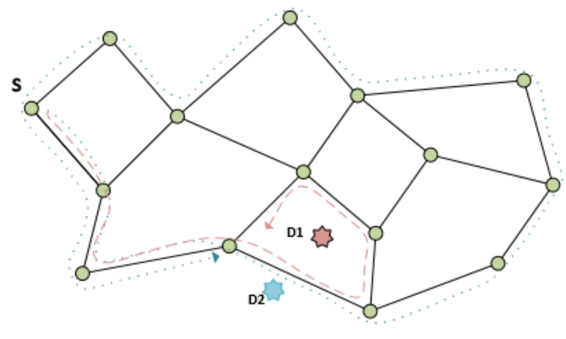

Fig 3. Two scenarios: D1 in the network and D2 outside of the network

In more details, the first scenario from Fig. 3, where D1 is inside of a face, the GFGF1 algorithm will always find the closest node with the 'greedy' part and with the 'face' part which traverses and encircles the D1. The routing will stop when an edge is traversed once again. It is called *the stop criterion*. In that way, a node on the routing path will always be the closest node to D1 and it will be surrounded by the nodes in the face where D1 is.

The main drawback of GFGF1 is that it has additional messaging overhead compared to GFG, for the stop criterion (i.e. in order to determine if the edge has been already traversed). On the other hand, it guarantees finding the closest robot.

In case of the second scenario, where D2 is outside of the network, routing will be done on the outer face thus always finding the closest node. However, communication overhead is significant in this case.

In the case where D2 is outside of the network (Fig. 3), the main drawback is high communication overhead compared to the first scenario. It is due to the routing path that includes many nodes and therefore more messages are needed for the routing. Besides, most of the nodes are irrelevant. The question in such a case would be – how to determine which of routed nodes are really the most significant ones and which one should react?

To lower the communication costs for this scenario the GFGF2 algorithm is designed. The proposed solution would be to define a radius R, and after the first node is found within the range R from D using GFG it is used as the stop criterion. This is depicted in Fig. 4. In order to examine the properties of this algorithm, range R is changed as R, 2R, 3R and 4R.

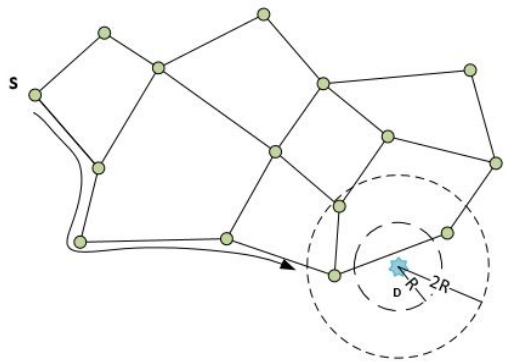

Fig 4. Illustration of GFGF2 algorithm

GFGF2:

1. Follow greedy until delivery or failure
2. **If** (failure) **then**
   - Search next node on the face (by choosing one among the neighbors of the node, based on right-hand rule
   - **While** dist (nextNode, D) > dist(addr, D)
     - Find next node
     - **If** (node in range R to destination) **then** finish the search
3. Search all the neighbors of that node in range R and add them to the routing path
4. **Repeat** steps 1 and 2 until D or node in range R

IV. Results and Discussion

In this section the simulation results and discussion are presented. Simulations are performed for the random network topology and random network topology with hole. Network parameters are set as follows: monitored area is a unit square, with node coordinates between 0 and 1, 100 nodes, parameter r (i.e. communication radius) is changed in range from 0.05 to 1, and 1000 simulations were performed for each setup. Disconnected networks were not taken into consideration.

The simulations showed that for r<0.15, networks are almost never connected, so those cases were not taken into consideration. Besides, for r>0.5 there were no significant change in the results when compared with r=0.5. Accordingly, the results for r<0.15 and r>0.5 are not presented.

In the simulations, we assume that collecting robot has information about the event (D) location and whether it is located inside or outside of the network. This information was calculated based on the point-in-polygon algorithm presented in [8].

The measured values are average communication costs (i.e. average number of messages used in the routing). The obtained results were compared with the communication cost of the GFG. Average costs for GFGF1 compared with GFG, as well as average number of messages in the face which is surrounding D until the stop criterion is reached, for random topology and random topology with hole are given in the Fig. 5 and 6, respectively.

Results show that GFGF1 has more than twice higher costs compared to GFG which is expected since event D is not part of the nodes in the network. For the random topology GFGF1 costs until the face around D is found, is very similar to the GFG communication costs. In topology with hole it is higher due to network topology (i.e. there are longer routes around the hole).

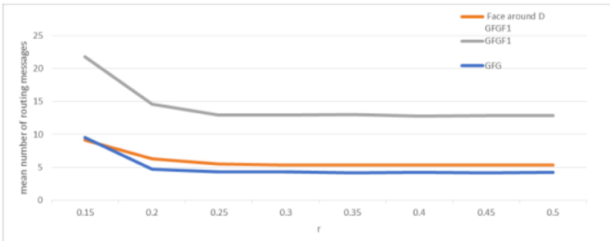

Fig 5. Average communication costs of GFGF1 (random topology)

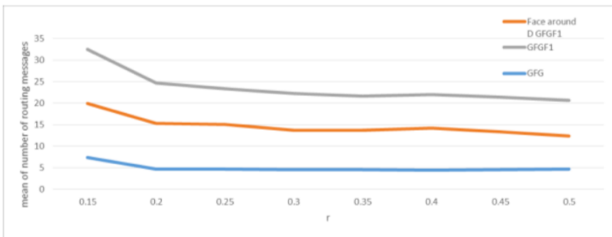

Fig 6: Average communication costs of GFGF1 (random topology with hole)

For both topologies, simulations showed that it is guaranteed to find the closest node to D for connected networks. It is another proof of the Corollary 1 made by simulations.

The main drawback of GFG1 algorithm is in cases when D is placed outside of the network. In that case, the overhead compared to GFG is really significant since the loop around D consists of outer face nodes. It is depicted in Fig. 7.

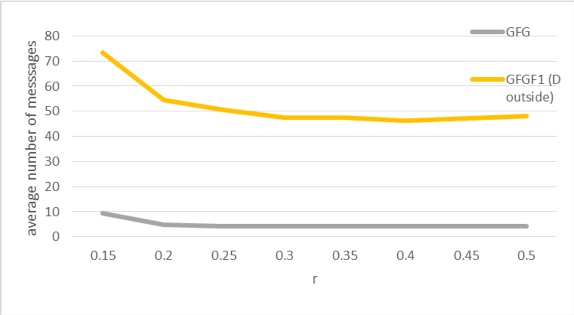

Fig. 7. Average communication costs of GFGF1 (D is outside of the network)

In order to lower the communication costs of GFGF1, GFGF2 is designed. GFGF2 algorithm is simulated with the same set of parameters as for GFGF1 but only for D placed outside of the network. Only difference is the stop criterion based on the parameter R (range in which are neighbor nodes of the node closest to D) which is changed from R to 4R. The results are shown in Fig. 8. It can be seen that there is a significant improvement (up to 800%) in overhead needed for routing using GFGF2 in case when D is outside of the network, compared to GFGF1.

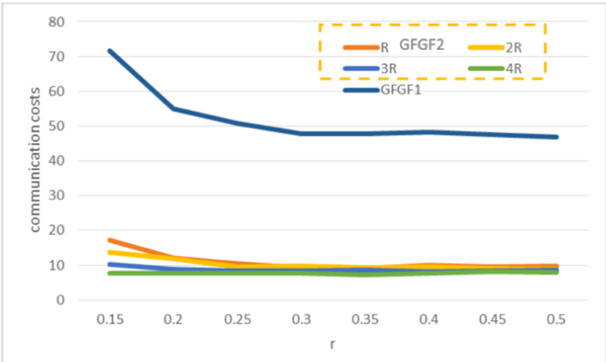

Fig. 8. Average communication costs of GFGF2 (R = 0.1, random topology)

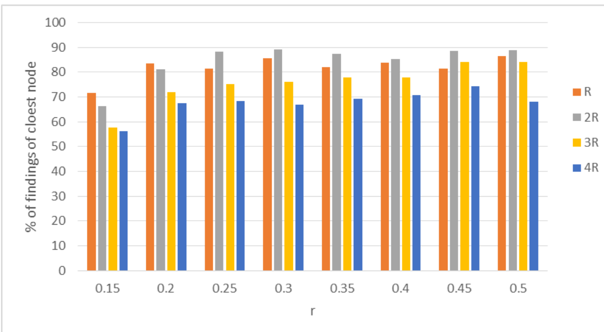

Fig9. Percentage of finding the closest node (R = 0.1, random topology)

In Fig. 9 is shown the percentage of finding the closest node to D, for various R. It is observed that this percentage varies from 55-90% for different parameters. The percentage is the highest for the 2R radius. Although GFGF2 does not find the closest robot, it has the communication costs that are more than 4 times lower featuring the main benefit of GFGF2 – communication overhead reduction.

Fig. 9. Percentage of finding the closest node (R = 0.1, random topology)

Similar results are obtained for the topology with hole.

Conclusion

In this paper two new algorithms for finding the event is proposed for two scenarios (event is within or out of the network). Both GFGF1 and GFGF2 algorithms are described and simulation results are presented. It can be observed that GFGF1 features guaranteed finding of the closest robot to the event but with communication overhead compared to GFG. This overhead is significant in scenarios where the event is outside of the network. To lower it GFGF2 is designed. It features more than 4 times communication cost reduction compared with GFGF1. However, percentage of finding the closest was up to 90%.

Further work on the subject will be based on a more thorough simulations performed using additional topologies (e.g. toroidal with crescent holes etc.). As one of the further research direction will be exploration of the influence and behavior of disconnected networks. Another possible research direction is possible application of these results to the behavioral anomaly detection in WSRN in the context of industrial IoT.

Acknowledgments

Work of the second author is partially supported by CPER DATA. The third author is partially funded by the technological development grant number TR32016, Ministry of Education and Science, Serbia and by the H2020 C4IIoT Innovation action project under Grant Agreement No. 833828 by European Commission.

References

[1] I. Mezei, V. Malbasa, I. Stojmenovic, “Task assignment in wireless sensor and robot networks”, in *Proc. 20th Telecommunications Forum Telfor*, Belgrade, 2012, pp. 596-602

[2] A. Khamis, A. Hussein, and A. Elmogy, “Multi-robot task allocation: A review of the state-of-the-art,” in *Cooperative Robots and Sensor Networks*, A. Koubâa and J. R. M. Dios, Eds. Cham, Switzerland: Springer-Verlag, pp. 31–51, 2015.

[3] M. Lukić, Ž. Mihajlović, I. Mezei, "Data flow in low-power wide-area IoT applications", In *Proc. 26th Telecommunications Forum TELFOR*, Belgrade, 2018., pp. 1-4.

[4] P. Bose, P. Morin, I. Stojmenovic and J. Urrutia, “Routing with guaranteed delivery in ad hoc wireless networks,” *3rd Int'l. Wksp. Discrete Algorithms Methods Mobile Comp. Commun., Seattle*, WA, 1999, pp. 48–55; also in ACM/Kluwer WL Nets., vol. 7, no. 6, Nov. 2001, pp. 609–16.

[5] X. Lin and I. Stojmenović, “Geographic distance routing in ad hoc wireless networks”, *Technical report* TR–98–10, SITE, University of Ottawa, 1998.

[6] E. Kranakis, H. Singh, and J. Urrutia “Compass routing on geometric networks” In *Proc. 11th Canadian Conference on Computational Geometry*, 1999., pp 51-54.

[7] H. Frey and I. Stojmenovic, "On delivery guarantees and worst-case forwarding bounds of elementary face routing components in ad hoc and sensor networks," in *IEEE Transactions on Computers*, vol. 59, no. 9, pp. 1224-1238, 2010.

[8] E. Haines, “Point in polygon strategies”, In *Graphics gems IV, Paul S. Heckbert (Ed.)*, Academic Press Professional, Inc., San Diego, USA, pp. 24-46, 1994.