Situational awareness for distributed mobile robot teams under limited communication

Maksim Kenzin, Igor Bychkov and Nikolai Maksimkin

Matrosov Institute for System Dynamics and Control Theory of the Siberian Branch of the Russian Academy of Sciences, 134 Lermontov str., 664033 Irkutsk, Russia

E-mail: `gorthauers@gmail.com, bychkov@icc.ru, mnn@icc.ru`

Abstract. A high level of team situational awareness is essential during complex, large-scale missions of autonomous mobile robots. When a situation appears that needs inter-agent interaction for cooperative decision-making, the basic understanding of the current conditions ought to be identical within the group. To achieve this requirement, all emergent information of acute importance must be promptly shared among team members. It is a non-trivial problem for large-sized and distributed robotic teams, especially under hard communication constraints. The problem considered in this paper is to find an efficient emergency broadcasting strategy for search and survey operations of the robotic groups providing the fastest way for any agent to aware the remaining team in case of any unexpected changes. A number of simple ruled-based heuristics is proposed to treat the problem. The comparison between the suggested approaches is made regarding both the quality of the obtained solutions and the working speed.

1. Introduction

The rapid evolution of the robotics technologies in recent years has led to the significant reliability improvement of research, military, and commercial autonomous vehicle systems. Currently, the coordinated use of autonomous mobile robot teams seems to be one of the most promising and ambitious technologies to provide an efficient solution to a range of problems, that are either too difficult or too costly to solve using traditional techniques.

However, an extensive range of fundamental problems still needs to be solved in order to achieve high-performance cooperative teamwork in a hostile and ever-changing environment [1]. In these circumstances, the ability to quickly and effectively react to unexpected changes and emergent events becomes the first challenge on the way to real-world applications. This problem can be decomposed into two components: decentralized broadcasting to share crucial data among the group and team decision making to manage them properly [2]. The latter problem we have extensively addressed in our previous works [3, 4], this work is focused on the former one.

In this paper, we present a distributed broadcasting problem for mobile robot teams under communication constraints. The problem is to find the shortest route for the starting aware agent to consecutively share some emergent information with the rest unaware team members, which are keeping their prescribed routes. As a first approximation, we propose here a more abstract and simplified discrete model of the problem and suggest several strategies to treat it.

The remainder of the paper is organized as follows. The mathematical statement of the distributed broadcasting problem is given in the next section along with short discussion on its classification aspects. A number of construction heuristics to solve the problem are proposed in Section 3. Section 4 provides an in-detail description of the problem software implementation techniques, including data sets generation schemes. The results of computational experiments are explored in Section 5. The last section concludes the paper.

2. Problem statement

In general, search and survey robotic missions require a group of mobile agents to visit a set of locations inside the operational area to perform some interaction or research activities. As the operational field is usually large compared to both sensing and manipulating capabilities of each robot, it is commonly-accepted to allocate tasks among the group to manage the job in a distributed way.

As the robots spread across the area, the current group strategy may become inefficient or even disadvantaging due to the dynamic mission conditions. For instance, an agent may find the object of search in no time or become aware of some information that is crucial for mission success [5]. In that event, this agent should decide if it is time/resource-worthy to warn the rest of the group, or it is more reasonable to let other robots finish their prescribed routes. If the shared information is crucial enough, other informed agents discontinue their work to join the broadcasting task implementation.

In this situation, communication constraints become a significant limitation. It is a problem for many real-world autonomous platforms, whether it is a micro-robotic swarm system with low range communication equipment or full-scaled underwater vehicle group, which communication abilities are limited due to the environmental properties. There is usually no all-seeing supervisor for global broadcasting, whereas inner-vehicle communication channel could be established only for close-enough agents [6].

Summarizing the above, the broadcasting problem is to find a route for the starting agent and future informed agents to achieve the fastest data distribution among the group. As unaware group members are non-stationary and time-dependent, even if their future positions are prescribed and known, it still lays an additional layer of complexity to the routing problem.

Let the graph $G = (V, E)$ of m vertices be a connected graph representing the operational field. Each vertex v_i, $i = 1, ..., m$ of G is a particular location of interest, and each existent edge $e_{ij} = 1$ corresponds to a path between adjacent locations. For the sake of simplicity, let the graph G be unweighted, undirected, and static.

The mission is carried out by the group of n homogeneous agents (robots) $a_1, a_2, ..., a_n$. Each agent i travels across the graph G on a discrete-time $T = \{T_0, T_1, T_2, ...\}$ by the prescribed route R_i, which is known in advance. Route $R_i = (r_{i0}, r_{i1}, ..., r_{iq})$ is a path graph with cycles as agents may stay on the same vertex $r_{ik} = r_{ik-1}$ for some time doing time-consuming activities. At any point in time, any number of agents can be located within the same vertex v.

We denote the state of i-th agent at j-th time moment as a list of its current vertex and status $a_i(T_j) = \langle v_{ij}, s_{ij} \rangle$. The agent status $s_{ij} \in \{0, 1\}$ displays if agent i is aware $(s = 1)$ at the time moment j. All agents are unaware at the beginning of the mission $s_{i0} = 0$, $i = 1, ..., n$.

At some instant time moment (without loss of generality, let it be T_1) random agent x became aware of some crucial information $s_{x1} = 1$. Since becoming aware, any agent receives two abilities:

(i) To drop its prescribed routes to become a subject of control;

(ii) To make other unaware and neighbouring (sharing the same vertex) agents aware

$$s_{kj} = \max_{i=1,...,n} \{s_{ij} | v_{ij} = v_{kj}\}. \tag{1}$$

The starting x-th agent has to decide if it is possible to make each other agent of the group aware in $h < q$ time steps (h is an evaluation of the time/resource-worthy period).

Thus, the final broadcasting problem is to find the solution group route in the matrix form $A = \{a_i(T_j)\}$, $i = 1, ..., n$, $j = 1, ..., h$ such that:

- Each agent is unaware when the mission starts $a_i(T_0) = \langle v_{i0}, 0\rangle$, $i = 1, ..., n$;
- Unaware agents travel by their prescribed routes until becoming aware $\{v_{ij} = r_{ij} | s_{ij} = 0\}$;
- The random x-th agent became aware at T_1, $a_x(T_1) = \langle v_{x1}, 1\rangle$;
- Aware agents make other unaware and neighbouring agents aware (1);
- All agents should become aware not later than h, $\prod_{i=1}^{n} s_{if} = 1$, $f \leq h$.
- As there are could be various ways to build an acceptable matrix A, those with lesser f would be preferable, $min_f(A)$.

There is also an alternate acceptance criterion to make all agents aware and gather them at some specific vertex v_R (rendezvous location) not later than h, $v_{if} = v_R$, $i = 1, ..., n$, $f \leq h$. This criterion, actually, makes the problem significantly harder and will be considered in our future works.

Problem classification. As we did not find any direct analogues of the proposed problem in the literature, we still discovered a number of agent-based problems that share some common features with it. Such graph models with static or mobile agents affecting each other states under specific rules are often used to model the information distribution in communication and social networks [7, 8] and even infection disease transmission [9, 10]. The modern concept of burning graphs is also built around similar ideas [11].

And the first-hand problem of the shortest (temporally in our case) group route construction refers to a numerous vehicle routing problem variations. Furthermore, the standard travelling salesman problem can be represented as a subproblem of the distributed broadcasting problem, in which unaware agents standstill on the same vertices for the whole mission and do not join the broadcasting task after becoming aware. It follows here-from that the proposed problem is also NP-hard. With the absence of the actual delivering process, it can be classified as some kind of time-dependent multiple travelling salesman problem (mTSP). It should be noted that the term dynamic cannot be applied here as mission conditions are still deterministic [12].

Among other things, one may also note, that the aware agents' behaviour is similar to one in a playground chasing game of tag, more specifically, its team variations (so-called vampire or werewolf tag).

3. Solving methods

In this section, we propose several strategies to build the group route A providing fast data broadcasting. The matrix representation $A = \{a_i(T_j)\}$, $i = 1, ..., n$, $j = 1, ..., h$ of the solution is apparently too big for efficient route generation straight in this form, as it contains a vast amount of both non-key and excessive information. For this reason, at first, we have suggested using numerical vector $P = \{x, p_1, ..., p_{n-1}\}$, $p_i \in \{1, ..., n\}$, encoding the order of agent activation (becoming aware and, therefore, controllable). But, as this data structure can be interpreted in multiple ways, we have switched to a schedule form of representation.

Currently, we encode a solution as the list of agent actions in their arising sequence $P = \{p_1, ..., p_{2n-1}\}$, $p_i \in \{0, 1, ..., n\}$, $p_i \neq x$. The requirement for the next action command arises for an unaware agent when it becomes aware and for an aware agent when it activates an unaware one (usually, both events happen simultaneously). The action command $p_i > 0$ refers to an index of the unaware agent to be activated next by the current one, while $p_i = 0$ means that the current agent has no one more to activate, and his broadcasting task is finished.

Two requirements should be met for the schedule P to be both feasible and acceptable:

(i) Each agent (except starting agent x) should be included in the schedule P once and only once, $\{p_i \neq p_j \mid p_i > 0 \bigvee i \neq j\}$, $card\{p_i \mid p_i > 0\} = n - 1$, $p_i \neq x$.

(ii) The number of 0-values ahead of any z-th element of P may never exceed the number of positive elements, $prod\{p_i \mid p_i = 0 \bigvee i \leq z\} \leq prod\{p_i \mid p_i > 0 \bigvee i \leq z\}$, $z = 1, ..., 2n - 1$.

The second requirement ensures the permanent availability of aware agents during the broadcasting task. The matrix solution A can be easily generated from the schedule form P by replacing agent sub-routes after their activation moment with the shortest paths to the unaware agents they are commanded to activate.

Using the representation form P, we have suggested five different construction heuristics to build and further compare group routes for the distributed broadcasting problem.

Random order. First one is a construction heuristic that generates random vector-solution P concerning two feasibility conditions mentioned above. We consider this approach, which is a priori ineffective, not as a real solution technique but as one to compare with.

Simple greedy heuristic. The second method is also a trivial, but working and reasonable in many cases, approach. In this heuristic, the action list P is built in such way, that agents are always commanded to activate the temporally nearest (as quickest to reach) unaware agent if any.

Experimental results have shown that the simple greedy approach has two observable drawbacks. Firstly, it works poorly in the situation, when several single agents are travelling diversely far away from the rest of the team. Secondly, it underperforms in areas with a vast amount of dead-ends. To answer these weaknesses, we have proposed two different modifications of the greedy heuristic, which we have called Near-and-far and Smart greedy heuristic, respectively.

Near-and-far heuristic. This method has a similar greedy structure but is probabilistic. Each time an action command should be generated for an aware agent, heuristic picks either the nearest unaware agent with probability b or the farthest unaware agent with probability $1 - b$. The probability b is changing with time according to the balance of aware and unaware agents in the group. From the mission start, b parabolically decreases from 1 to 0.65 (experimentally defined values) towards the time when the quarter of agents is aware. Then it returns to 1 by the moment of 50/50 ratio and stays on its level until the end of the broadcasting task.

Smart greedy heuristic. Smart greedy heuristic utilizes an alternative conditional check against its simple version. It picks the nearest agent not among all still unaware agents, but among those of them, which he can reach quicker, than any other aware agent in the group. If there are no such agents at all, the current agent receives action command "0".

Branching heuristic. All previously suggested approaches have the common feature as being centralized: the starting agent generates the straight global solution A and shares it with other agents (alongside with the main broadcasting data) to abide by it. Branching heuristic is our attempt to build a decentralized broadcasting technique for the problem. The main idea here is to assign each aware agent a personal subset of unaware agents as its activation task. Each time an aware agent activates an unaware one, it divides his subset into two halves and assigns one of them to the newly activated agent. The main problem here is to build an accurate clustering rule as both spatial and temporal conditions should be considered. For this work, we use the standard single-linkage clustering with the metric being average distance in time.

4. Software implementation

The proposed problem and the heuristics suggested above have been software-implemented as the additional module for our simulation framework "Multiobjective Mission Planner" to run a series of simulation studies (figure 1).

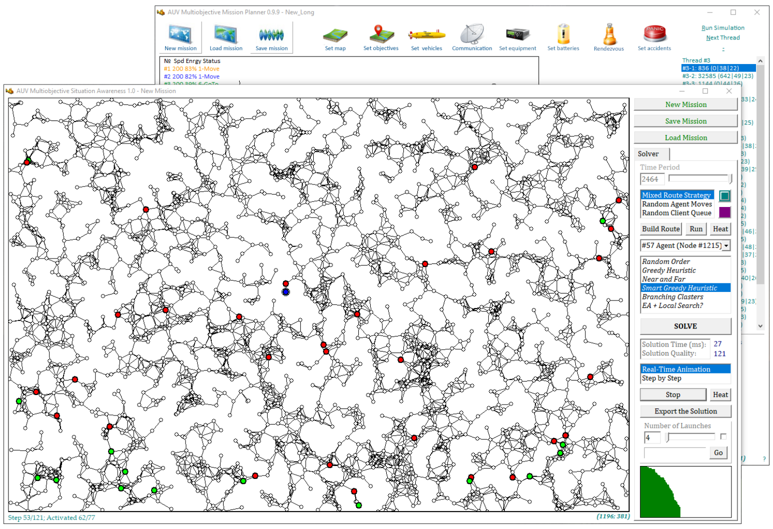

Figure 1. The main window of the developed module within the simulation framework.

To start the problem solving, specification data can be either imported (from the main framework or specific data file) or user-generated through the graphical interface. While generating, we represent graph vertices as a set of two-dimensional points in the plane of $W \times H$ size (figure 2a). Then we use one of two different methods to generate edges based on this set.

First is a standard Delaunay triangulation, in which we remove 10% of the longest edges both to create a more complex environment and to prevent fast travelling through the graph, especially by the graph borders (figure 2b). The second method is an approach to simulate complex non-regular areas and indoor environments. To do this, we choose the neighbourhood size $S = (\alpha WH/n)^{1/2}$ and connect each pair of vertices with Euclidean distance less than S. We use parameter $\alpha = 2$ for big neighbourhoods and $\alpha = 1$ for small neighbourhoods (figures 2c and 2d, respectively). Then we complete the graph construction by adding edges of minimal possible total length sum to provide the graph connectivity.

Up to that moment, we keep the graph in the form of the adjacency matrix. This type of representation is easy to implement, follow and modify during the graph construction stage. As soon as the final graph state is confirmed, we convert it into the adjacency list $L[i][j], i = 1, ..., m, j = 0, ..., L[i][0]$ as it is much more convenient structure for pathfinding procedures. We keep the total number of adjacent vertices for each vertex i as 0-th element $L[i][0]$ since it is a frequently used value.

As there are usually a large number of single runs on the same graph (different combinations of prescribed routes, starting agents and solution methods), we suggest building an additional distance matrix D of $m \times m$ size to speed up further calculations significantly. To do this, we run the standard Dijkstra algorithm on each vertex i of the graph to find a distance $D[i][j]$ to each other vertex j.

For initial group route construction, we propose using three route generating schemes. First one is a "random agent movements", where each time-step agent either chooses to move on a random adjacent vertex or to loiter (standstill on the same vertex). The probabilities for both

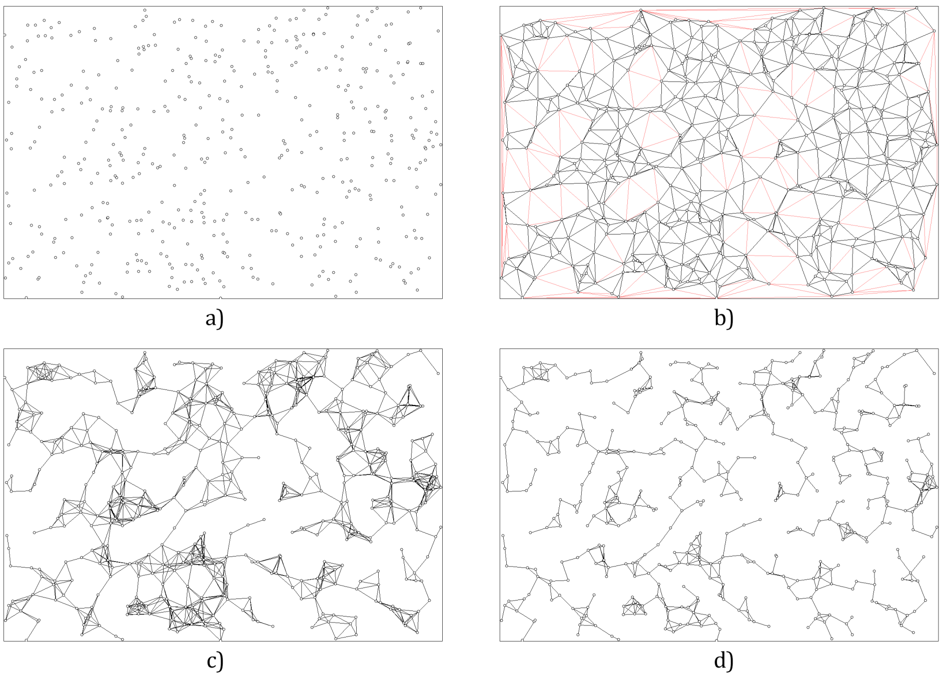

Figure 2. Three different graph generation schemes: a – initial set of vertices; b – reduced (red edges are removed) triangulation; c – big neighbourhoods; d – small neighbourhoods.

loitering and returning to the previously visited vertex are reduced here to prevent an agent from looping (Algorithm 1).

```
Input  : Agents initial positions $v[i]$, graph $G$ adjacency list $L[j][k]$
Output: A group route $R[][]$ of size $n \times h$
1  for $i \leftarrow 1$ to $n$ do
2      $R[i][0] \leftarrow v[i]$;
3      $R[i][1] \leftarrow v[i]$;
4      for $j \leftarrow 2$ to $h$ do
5          AgentPosition $\leftarrow R[i, j-1]$;
6          EdgeIndex $\leftarrow$ random$(1, ..., L[\text{AgentPosition}][0])$;
7          if $L[\text{AgentPosition}][\text{EdgeIndex}] = R[i, j-2]$ and random$(1,2) = 1$ then
8              $Route[i][j] \leftarrow Route[i][j-1]$;
9              go to 4;
10         end
11         $R[i][j] \leftarrow L[\text{AgentPosition}][\text{EdgeIndex}]$;
12     end
13 end
```

Algorithm 1: Group route generation – random agent movements

For the second strategy, we generate single routes not as a sequence of edge movements, but as a queue of checkpoints (random vertices of G) for an agent to visit. For each consecutive pair (i, j) of checkpoints we build the shortest path of $D[i][j]$ length to insert it to the final route (Algorithm 2).

```
Input  : Agents initial positions $v[i]$, graph $G$ distance matrix $D[j][k]$
Output: A group route $R[][]$ of size $n \times h$
for $i \leftarrow 1$ to $n$ do
    $R[i][1] \leftarrow v[i]$;
    AgentPosition $\leftarrow R[i][1]$;
    RouteLength $\leftarrow 1$;
    while RouteLength $< h$ do
        Checkpoint $\leftarrow$ random$(1, ..., m)$;
        Path[] $\leftarrow$ FindShortestPath(AgentPosition, Checkpoint);
        PathLength $\leftarrow D$[AgentPosition][Checkpoint];
        if RouteLength + PathLength $> h$ then
            PathLength $= h -$ RouteLength;
            Path[] $\leftarrow$ CutTheRoute(Path[], PathLength);
        end
        $R[i]$[RouteLength] $\leftarrow$ CopyArray(Path[], PathLength);
        AgentPosition $\leftarrow$ Checkpoint;
        RouteLength $\leftarrow$ RouteLength + PathLength;
    end
end
```

Algorithm 2: Group route generation – random checkpoints queue

In the third mixed approach, route for each agent is randomly chosen to be generated by either the first or the second strategies described above. We also have tried to implement more intelligent agent behaviour that simulates robot search activities by alternating area exploration and exploitation strategies. But, as the overall collective behaviour, in this case, wasn't so different from the third mixed strategy, we did not use it in the computational experiments.

The difference between three proposed route generation methods is illustrated on the figure 3. These heat maps are constructed for the same initial data (graph G and agents set) and represent the distribution of the agents' positions while travelling by the corresponding routes for the significant time. As can be seen, the mixed strategy delivers more even and realistic agent distribution by combining the vertex-oriented approach of the random agent movements (focus on the strongly connected components of G) with edge-oriented features of the checkpoint heuristic (focus on the graph bottlenecks).

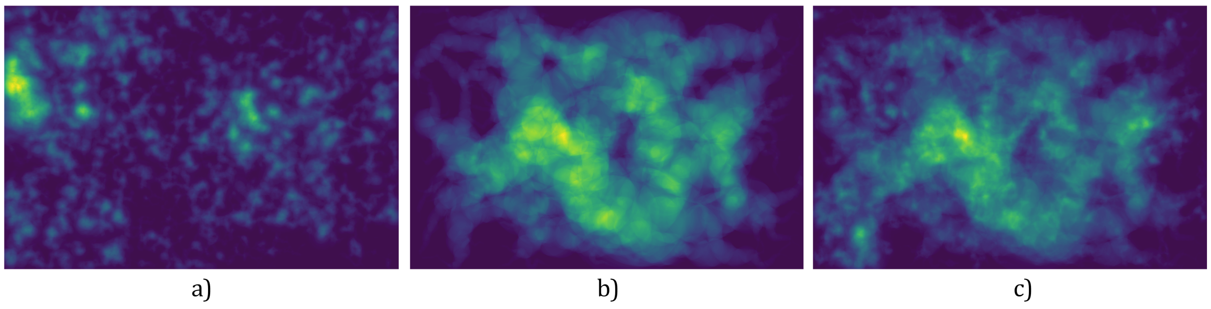

Figure 3. Heat maps for different approaches to group route generation: a – random agent movements; b – random checkpoints queue; c – mixed strategy.

5. Computational experiments

For the computational experiments we have generated the set of problem instances of the form "XY-$m(n)$". These instances are divided into nine categories: three types of graph generation schemes X={T,B,S} for Delaunay **T**riangulation, **B**ig and **S**mall neighbourhoods, respectively; three approaches to group route generation Y={R,C,M} (**R**andom agent movements, **C**heckpoints queue, **M**ixed strategy); and we have instances of $n = \{25, 50, 100\}$ agents on the graphs of $m = \{500, 2000, 5000, 10000\}$ vertices for each combination of X-Y.

As solution values may strongly vary based on the initially aware agent, we use only one fixed starting agent for each instance instead of the average value for combinations of different ones. Nonetheless, as some methods are randomness-involved, we made 100 launches for each instance and each approach to obtain both average solutions f and computational time values t.

Statistical results for a number of executed problem instances are presented in table 1. Solution time values t here are measured in milliseconds (ms) and do not include construction time to obtain distance matrix D for the corresponding graph G. A note on computation time: all calculations are running on the single core of 2.667 GHz processor of an Intel Core 2 Duo E6750 Conroe.

Table 1. Statistical results of computational experiments.

#	Instance	Random		Greedy		Near-and-far		Smart greedy		Branching	
		t	f	t	f	t	f	t	f	t	f
1	TR-500(25)	*0.8*	55	*0.8*	33	*0.8*	34	*1.1*	**30**	*0.8*	38
2	TC-500(25)	*0.9*	46	*0.9*	27	*0.9*	28	*1.0*	**25**	*0.9*	33
3	TM-500(25)	*0.9*	53	*0.8*	30	*0.8*	30	*1.1*	**27**	*0.9*	36
4	BR-500(25)	*0.8*	77	*0.9*	53	*0.9*	51	*1.3*	**38**	*0.8*	49
5	BC-500(25)	*0.9*	56	*0.9*	31	*0.8*	32	*1.1*	**28**	*0.9*	40
6	BM-500(25)	*0.9*	67	*0.8*	40	*0.8*	40	*1.3*	**32**	*1.0*	48
7	SR-500(25)	*0.9*	219	*1.0*	138	*1.1*	134	*3.0*	**92**	*1.0*	155
8	SC-500(25)	*0.9*	135	*0.9*	78	*1.0*	81	*2.1*	**76**	*1.0*	80
9	SM-500(25)	*0.9*	178	*0.9*	103	*1.0*	101	*2.5*	**85**	*1.0*	107
10	TR-5000(50)	*5.1*	222	*5.6*	106	*5.7*	97	*17.0*	**91**	*5.2*	124
11	TC-5000(50)	*4.8*	209	*5.3*	87	*5.4*	87	*12.3*	**77**	*5.0*	93
12	TM-5000(50)	*5.0*	217	*5.4*	95	*5.6*	93	*14.3*	**84**	*5.3*	137
13	BR-5000(50)	*5.3*	290	*6.0*	152	*6.2*	147	*21.4*	**120**	*5.8*	172
14	BC-5000(50)	*4.9*	231	*5.6*	107	*5.7*	115	*14.1*	**95**	*5.7*	142
15	BM-5000(50)	*5.3*	256	*5.9*	135	*6.1*	128	*16.6*	**109**	*5.7*	164
16	SR-5000(50)	*5.7*	1095	*9.1*	584	*9.2*	572	*113.9*	**475**	*7.0*	673
17	SC-5000(50)	*5.2*	779	*7.8*	377	*7.9*	394	*53.3*	**359**	*6.6*	414
18	SM-5000(50)	*5.5*	959	*8.5*	485	*8.6*	492	*83.3*	**395**	*7.0*	602
19	TR-10000(100)	*16.4*	472	*19.9*	189	*26.5*	184	*231.1*	**152**	*18.6*	229
20	TC-10000(100)	*15.1*	365	*18.1*	122	*18.6*	117	*89.8*	**102**	*17.7*	202
21	TM-10000(100)	*15.7*	433	*18.4*	139	*20.3*	140	*151.2*	**129**	*17.9*	206
22	BR-10000(100)	*17.0*	577	*21.2*	242	*28.4*	241	*248.7*	**173**	*19.6*	309
23	BC-10000(100)	*16.7*	409	*19.1*	142	*20.5*	164	*101.0*	**118**	*19.1*	228
24	BM-10000(100)	*17.5*	496	*20.4*	193	*21.6*	185	*141.0*	**139**	*19.6*	281
25	SR-10000(100)	*26.0*	2040	*41.4*	974	*43.8*	861	*1506.0*	**676**	*25.3*	970
26	SC-10000(100)	*19.8*	1430	*38.9*	636	*39.2*	661	*837.5*	**600**	*32.5*	727
27	SM-10000(100)	*17.0*	1297	*27.4*	586	*28.9*	494	*533.5*	**365**	*22.5*	754

Despite the apparent advantage of the smart greedy heuristic over the rest approaches, it should be noted once more, that the solution values f presented in the table 1 are average for the series of 100 launches. Superior results of the smart greedy heuristic are provided, in the first place, by its inability to generate low-quality solutions. For instance, both Branching and, especially, Near-and-far are able to outperform other methods on a single launch (as opposed to the simple greedy heuristic, which is always equal or worse than its smart analogue). Still they are also capable of delivering inferior solutions (figure 4). Nonetheless, we believe that more intelligent use of these methods' features may allow them to compete with smart greedy evenly.

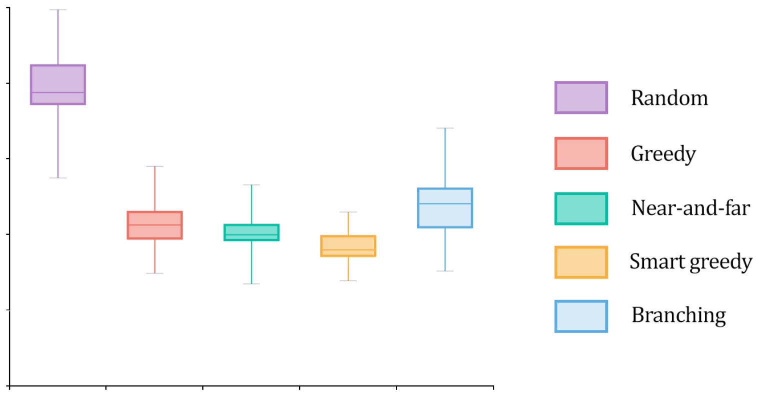

Figure 4. Solution distribution comparison from the Monte Carlo simulation (5000 launches) of the "TM-10000(50)"-instance with different initial group routes and starting agents.

Another characteristic of each solution is its activation timeline. It is a function that follows the dynamic of unaware agent number with time. Figure 5 displays general outlines of activation timelines for the proposed heuristics. Weak spots of some approaches can be clearly seen here (activation speed reduction for the latest agents in 5a and 5d) as the apparent room for further improvements.

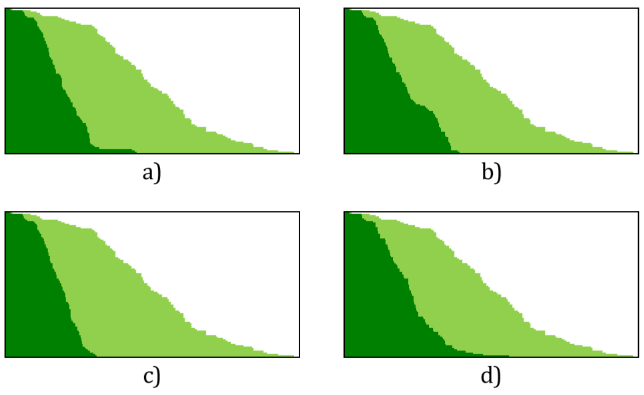

Figure 5. Group activation timeline for the different heuristics comparing to the random method (figure on the background): a – greedy; b – near-and-far; c – smart greedy; d – branching.

6. Conclusion

This paper presents a new problem of distributed emergency broadcasting for autonomous mobile robot teams under limited communication. The presented problem is to find the shortest route for the starting aware agent to share some emergent information with the rest unaware members of the group and is formulated as the original variation of the vehicle routing problem. As opposed to the classical travelling salesman problem, clients (unaware robots) here are not static as they travel through the mission area along the known route on a discrete-time. The main feature of the problem is that unaware agents after being informed by any aware agent leave their prescribed routes to join the broadcasting task.

Several simple ruled-based heuristics were designed and proposed to treat the problem. A series of computational experiments and comparisons among the proposed constructive heuristics showed that the addition of even simple problem-oriented rules allows standard greedy heuristic to performs efficiently and ambitious, in terms of both solution quality and computational efficiency. Nonetheless, it is still an interesting and open problem to find out possible ways to further improve these solutions in an efficient manner.

Among the future developments we intend to undertake, there is the development of more advanced meta-heuristic approaches and local search methods to deal with the problem. We also plan to embed more realistic environment into the problem statement, moving from the discrete formulation towards continuous time, complex weighted and directed graphs, and heterogeneous (by speed and communication capabilities) agents. Another extension of this work is to adjust the proposed constructive heuristics to the alternative problem statement with the final rendezvous point.

Acknowledgments

This work was supported in part by the RFBF, project no. 20-07-00397 (time-dependent routing problem for distributed mobile robot teams with data sharing), and by the Russian Science Foundation, project no. 16-11-00053-П (algorithm simulation framework for the dynamic multi-objective mission planning).

References

[1] Dunbabin M and Marques L 2012 Robots for environmental monitoring: significant advancements and applications *IEEE Robotics & Automation Magazine* **19(1)** 24–39

[2] Gan S K, Xu Z and Sukkarieh S 2016 Distributed situational awareness and control (*Encyclopedia of Aerospace Engineering*) eds Blockley and W Shyy pp 1—11

[3] Kenzin M, Bychkov I and Maksimkin N 2018 Task allocation and path planning for network of autonomous underwater vehicles *Int. J. of Computer Networks & Communications* **10(2)** 33—42

[4] Kenzin M, Bychkov I and Maksimkin N 2019 Two-level evolutionary approach to persistent surveillance for multiple underwater vehicles with energy constraints *SPIIRAS Proceedings* **18(2)** 267—301

[5] Papp Z 2012 Situational awareness in intelligent vehicles (*Handbook of Intelligent Vehicles*) ed A Eskandarian (London: Springer London) pp 61—80

[6] Arvin F, Murray J, Shi L, Zhang C and Yue S 2014 Development of an autonomous micro robot for swarm robotics *2014 IEEE International Conference on Mechatronics and Automation* 635–640

[7] Elsasser R, Lorenz U and Sauerwald T 2004 Agent-based information handling in large networks (*Mathematical Foundations of Computer Science 2004. Lecture Notes in Computer Science. Vol 3153*) ed J Fiala, V Koubek and J Kratochvil (Berlin: Springer)

[8] Simon M, Huraj L, Shi L, Dirgova-Luptakova I and Pospichal J 2019 Heuristics for spreading alarm throughout a network *Applied Sciences* **9(16) 3269**

[9] Pastor-Satorras R and Vespignani A 2001 Epidemic dynamics and endemic states in complex networks *Physical Review* **E63.6**

[10] Riley S 2007 Large-scale spatial-transmission models of infectious disease *Science* **316(5829)** 1298-–1301

[11] Farokh Z, Tahmasbi M, Tehrani Z and Buali Y 2020 New heuristics for burning graphs *Preprint* cs-dm/2003.09314

[12] Punnen A P 2007 The traveling salesman problem: applications, formulations and variations (*The Traveling Salesman Problem and Its Variations*) eds G Gutin and A P Punnen (Boston: Springer US) pp 1—28