DOI: 10.15439/2018F213
ISSN 2300-5963 ACSIS, Vol. 15

Autonomous Graph Partitioning for Multi-Agent Patrolling Problems

Bernát Wiandt and Vilmos Simon
Budapest University of Technology and Economics
in Budapest
Magyar tudósok krt 2., Hungary
Email: [bwiandt,svilmos]@hit.bme.hu

***Abstract*—Patrolling algorithms are coordinating multiple agents with the goal of visiting points of interest in a timely manner. These algorithms play a major role in efficient use of UAVs or other autonomous vehicles for precision agriculture, large area monitoring or security use cases. These algorithms are either centralized and in need of constant connection with the agents or solve NP-hard problems to plan the routes of individual agents on the graph. These requirements become unfeasible when the number of agents or points of interest grow or become dynamic. In this article we further elaborate the performance characteristics of the Partition Based Patrolling Strategy (PBPS) algorithm. The partitioning requires only local interactions between agents, therefore it is scalable to a large number of nodes and agents. On these subgraphs agents patrol independently from each other, therefore the approach eliminates interference between agents.**

I. Introduction

PATROLLING is the task of visiting points of interest in a timely manner. These points of interest can be selected for security purposes (monitoring entry points of an area) or points in space one would like to monitor[1] by making photographs or measurements. Every problem that involves points of interests distributed in space and limited possibilities to move between them (modeled by a graph) is a patrolling problem. When patrolling inside a building, points of interest can be intersections or doors needed to be inspected in a timely manner. When dealing with a large open area, like an agricultural crop, the points of interest are locations where one wants to make measurements, i.e. water level, infections, fruit maturity, etc.

Points of interest are usually modeled as nodes of a graph, where the edges represent the option to travel between them directly and the length of the edges represent the distance or cost of moving between them. Patrolling is usually performed by physical agents, such as Unmanned Autonomous Vehicles (UAVs) or robots[2], either on ground or in the air. In this article we refer to these entities, without loss of generality, as agents. Patrolling with a single agent is a matter of calculating a sequence of points of interest for the agent to visit. Instead, our focus is on multi-agent patrolling, where the task is to efficiently coordinate an arbitrary amount of agents in order to minimize some performance metric associated with the patrolling task.

Coordinating multiple agents to perform patrolling can be reduced to a multiple traveling salesman problem, therefore it is an NP-hard problem[3]. Patrolling algorithms either plan the routes for each agent ahead of time and feed those routes to the agents as a priori information or try to coordinate the agents in real time by allowing them to communicate through some wireless medium and employing various heuristics in the agents to decide on the next graph node to visit.

Patrolling on graphs is either implemented by allowing every agent to visit all nodes in the graph or the graph is partitioned into subgraphs[4], [5] and the agents work only on their own subgraph without interfering with each other. The first case involves interference between the agents that can result in agents blocking each other on the way from one node to another. When agents try to move in the same direction or meet at an intersection, they inhibit each other from carrying out their tasks as fast as possible. These interference events can have a negative impact on the performance and scalability of the patrolling algorithm. Since patrolling is usually a long running task, these interference events will have a continuous, most of the time unpredictable effect on the performance.

To counter this effect, researchers started investigating partition-based patrolling. Currently partitioning is usually done before patrolling starts, on a central entity, and the resulting partitions are assigned to the agents as a priori information. This method is not always feasible as environments can be dynamic: appearing or disappearing graph nodes and failing or newly introduced agents can trigger the recalculation of partitions. Therefore agents need the central entity throughout the patrolling task and need synchronization at times of partition changes. A self-organizing partition based strategy is more desirable, because agents need less synchronization, require only local information and it makes the whole system more robust against agent failures.

II. Related works

Patrolling strategies can be really simple heuristic based approaches or complex learning algorithms that try to find the optimal route for a multi-agent team. It is not always clear whether a complex approach or a simple heuristic can offer better performance. In [6] patrolling is cast into a multi-agent Markov Decision Process and optimization techniques are applied to efficiently find the optimal paths for agents on the graph. However in the same article this solution is compared to a simple reactive algorithm and performance differences with regards to the instantaneous idleness metric were negligible. This experiment and the conclusion of [7] suggest that computationally complex approaches are not always justified and simple reactive algorithms can match the performance of their more elaborate counterparts.

Agents can be restricted to their own subgraph or partition and perform patrolling alone in that region. An early theoretical study [3] gives some insight into the performance characteristics of the partition-based and cyclic approaches (for example the previously mentioned SingleCycle). Authors of this article claim that the optimal patrolling strategy for a single agent can be obtained by finding a shortest closed walk on the graph. This can be reduced to the Traveling Salesman Problem for which good approximations exist (for example the Christofides algorithm with $O(n^3)$ for a $\frac{3}{2}$ approximation). When multiple agents are considered, one strategy can be to traverse the closed path with evenly distributed agents along the path. The performance of such a strategy is bound by the edge length distribution of the graph [3]. For example on graphs with “long corridors" (edges connecting two distant nodes) a partition-based strategy could be a better fit, because those “long corridors" can be avoided by leaving them out of the partitions assigned to the agents.

So far the solutions enumerated were dealing with the multi-agent patrolling problem based on the assumption that agents cannot interfere with each other. However, interference can occur when two agents are too close to each other and cannot proceed in the direction of their targets. Interference is important when the algorithm is designed to be deployed on physical agents and can render theoretically better algorithms inferior in the real world. Accounting for interference results in a much harder problem, because the theoretically best results are no longer optimal if agents have to stop before colliding with each other or possibly choose different targets. Several new heuristic approaches are taking into account this phenomena, for example in [8] authors implement Greedy Bayesian Strategy (GBS) and State Exchange Bayesian Strategy (SEBS). Both strategies require global communication meaning that agents can communicate with all other agents involved in the patrolling task. They employ reactive agents (so the agents do not form plans ahead of time) and the decision making algorithm that chooses the next graph node to visit is based on Bayes' rule. The latter strategy (State Exchange Bayesian Strategy (SEBS)) aims to minimize the interference between agents while patrolling, by communicating the agent's next node it is intending to visit. SEBS is able to achieve better performance than any other heuristic studied in the article. Interference minimalization and graphs with non uniform edge length distributions contributed to the interest in partition based strategies.

Numerous other approaches have been tried to address the multi-agent patrolling problem, such as spanning tree[9], [10] or graph partitioning based approaches[11], [12]. Others involve task allocation based strategies[13] and evolutionary algorithms[14] or linear programming[15]. Auction based strategies involve agents placing bids on nodes to patrol on the graph. These kind of approaches implement a decentralized task allocation, where the agents decide on the outcome and able to reassign nodes or partitions of poorly performing agents to others[16], [17].

III. The Multi-Agent Patrolling Problem

In this article our goal is to further elaborate the performance characteristics of our proposed PBPS algorithm [18]. In our model the only global knowledge the agents have is the map they are patrolling on. They do not have the capability to communicate globally with each other. Agents' behavior will result in solving the patrolling problem efficiently together, without third party coordination.

In the model, the connected graph $G(V, E)$ consists of nodes $v_1, v_2, ..., v_n \in V$ and undirected edges $e_{i,j} \in E$. Nodes are the points of interest, they are visited by the agents periodically. Each node has a counter that measures time between the last visit to that node and the current time. This counter is usually referred to as the *instantaneous idleness* of the node at time t and is defined as $I_{v_i}(t) = t - I'_{v_i}$, where I'_{v_i} is the time at the last visit to the node and $v_i \in V$. In some use cases patrolling is done to measure some physical quantity at the point of interest periodically and as frequently as possible. Let us suppose that the measurement at each point of interest takes the same amount of time for each agent.

Agents' movement is restricted to the edges between connected nodes. Agents have a priori knowledge of the graph nodes and edges, and are capable of moving between nodes along the edges with constant and uniform speed. Agents are able to communicate with each other, therefore influence each other's decisions by broadcasting messages via a wireless medium.

Patrolling algorithms have very different characteristics with respect to the routes they take, the order in which they visit certain nodes, etc. To be able to compare their performances, we have to define metrics that represent the goodness of an algorithm from a certain point of view. The metric used for this task is usually the *average idleness* or the *worst idleness*. Both of these metrics are based on the *instantaneous idleness* defined before as $I_{v_i}(t)$. This basic metric can be averaged in a window or throughout the simulation to obtain the *average idleness*, defined as:

$$\overline{I_{v_i}}(t) = \frac{\overline{I_{v_i}}(t_{v_i}) \cdot C_i + I_{v_i}(t)}{C_i + 1} \tag{1}$$

where C_i is the number of visits to v_i. The *average graph idleness* is the average of *average idleness* values, such that:

$$\overline{I_G}(t) = \frac{1}{|V|} \sum_{i=1}^{|V|} \overline{I_{v_i}}(t) \tag{2}$$

A problem with the *average graph idleness* metric is that there can be two patrolling algorithms with the same *average graph idleness* but significantly different behaviors. *Average graph idleness* masks important characteristics of patrolling algorithms and as a consequence, its value can be misleading when comparing different algorithms. The idleness times between visits are samples from an unknown distribution and to reason about that unknown distribution based on the mean of a limited amount of its samples can be misleading. Therefore in this article we use the distribution of the *worst idleness* values to compare performances. The *worst idleness* associated with a node between times t_a and t_b is defined as the maximum *instantaneous idleness* value:

$$WI_{v_i} = max\{I_{v_i}(t_a), ..., I_{v_i}(t_b)\} \tag{3}$$

The worst idleness times are the worst case scenario, therefore by gathering enough samples, one can be reasonably sure of an upper bound of the idleness times that can be observed during patrolling. Patrolling use cases like reading measurements from sensors or taking photographs of certain physical locations usually benefit from a known maximum time between measurements. The *worst idleness* distribution makes it possible to evaluate patrolling algorithms based on this criteria. It also shows the fairness of the algorithm: the difference between the minimum and maximum *worst idleness* values on the graph over time. Moreover the distribution of *worst idleness* can be easily visualized using box plots.

IV. Partition Based Patrolling Strategy (PBPS)

The patrolling strategy we use in this article follows the partition based approach. Based on [3], [19], the performance of such a patrolling algorithm is in theory inferior compared to a cycle-based strategy on graphs without “long corridors". However, if the model accounts for the effects of interference between agents, this may no longer be true. Partitioning the graph between the agents is not a new idea, authors in [8] and [5] improved their patrolling algorithms by limiting the amount of interference between agents. Partition based strategies take this idea to the extreme as there is no interference between agents, they patrol their own partitions independently.

PBPS was published in [18] along with the first results obtained on well-known graphs. Here we give only a short introduction to the algorithm. PBPS has two important assumptions about the environment: agents can localize themselves and they know the structure of the graph a priori. It should be noted, that these assumptions are not typical in self-organized systems, because they require agents with greater knowledge than usual. However the information requirement of PBPS is the same as other patrolling algorithms, such as SEBS or Greedy Bayesian Strategy (GBS). The problem of discovering the environment and forming consensus about it among the agents is not discussed in this article.

The goal of the algorithm is to partition the graph into non-overlapping connected subgraphs and in the process create a global partitioning that enables the efficient patrolling of the graph. All agent's partitions are empty in the beginning. A partitioning of G is defined as $P = \{P_1...P_k\}$, such that $P_1 \cup ... \cup P_k = V$ and $P_i \cap P_j = \emptyset$. $\{G_1...G_k\}$ refer to subgraphs induced by the partitioning, thus $G_i = (V \cap P_i, E \cap (P_i \times P_i))$. The partition growing process is self-organized, such that there is no global coordination between agents and no global

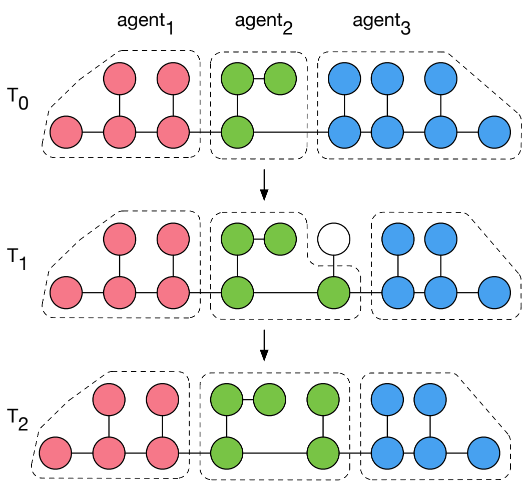

Fig. 1: Claiming of nodes between agents in order to achieve a balanced partitioning.

state shared among them. Agents grow their own partitions by claiming nodes from each other's partitions and broadcasting their new partition after claiming a new node. In the process (see for example Figure 1) of growing partitions, there can be local violations to the strict non-overlapping partitioning introduced above, but these are temporary and will resolve over time. The algorithm always reaches a state, where the partitions are stable, i.e. no more changes made by the agents. When agents arrive at a stable configuration, each partition will belong to one and only one agent and each partition will be a connected subgraph with no overlapping nodes with other partitions. The resulting partitioning assigns a subgraph to each agent to patrol with roughly equal number of nodes.

V. Simulation environment

We have conducted numerous simulations to investigate the performance characteristics of PBPS and compared patrolling on the formed partitions to well-known patrolling algorithms. First we describe the environment in which the experiments were carried out as it was designed to resemble possible real-world usage scenarios. We used the osmnx python library[20] to download maps of two Hungarian cities: Budapest and Komárom. We chose Budapest, since it is the capital of Hungary and like large cities it has a lot of intersections and shorter edges. Komárom on the other hand tends to have differently distributed edge lengths and node degrees (as can be seen on Fig. 2 and 3) as it is a smaller city on the countryside. Node degrees tend to be lower for Komárom but edge lengths are significantly longer than the ones for Budapest. We chose two different real maps to model possible usage scenarios and also to have environments with different characteristics. Our partitioning strategy is based on the assumption that the edge length distribution of the graph is more or less even. The map excerpt from Komárom has a longer tail in its edge length distribution, therefore more variety than the map excerpt from Budapest. The downloaded maps were simplified as the

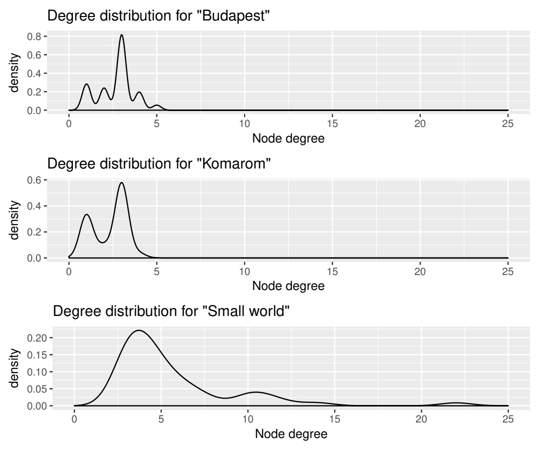

Fig. 2: Degree distributions of the maps used for the experiments.

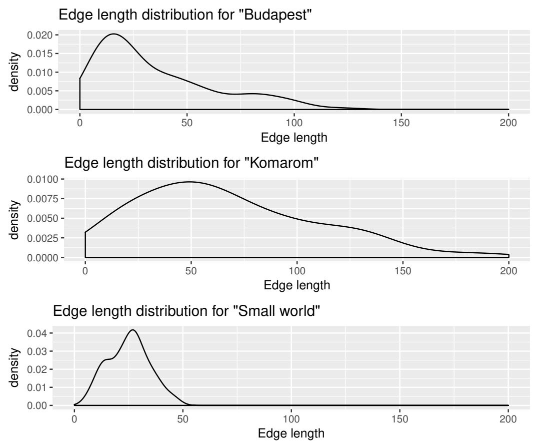

Fig. 3: Edge length distributions of the maps used for the experiments.

original Open Street Map data contains a lot of short edges and unnecessary nodes. The simplification step was done using a built-in functionality in osmnx. A third map was introduced, called Small World, which is a generated small world type graph (Barabási-Albert graph) with a completely different node degree distribution than the other two maps. This graph has 50 nodes and was grown with 3 edges preferentially attached to existing nodes using the networkx python library's built in graph generator. This third environment has some really high degree nodes and claiming those nodes in the partitioning phase can effect other partitions, as these high degree nodes act as access points to the rest of the graph.

We carried out experiments in these graphs with three different patrolling algorithms with parameters given in Table I. First, Conscientious Reactive[21] was used as a baseline as it is a greedy heuristic with no communication requirements among the agents. The second algorithm tested was State Exchange Bayesian Strategy[8], which is based on the idea that agents can perform better if they exchange their intentions and

Parameter	**Value**
Number of agents	$1, 2, 4, \ldots, 32$
Double number of agents every	$200000s$
Number of iterations	10
Agent speed	$1m/s$

TABLE I: Simulation parameters for the local vs. global metric experiment.

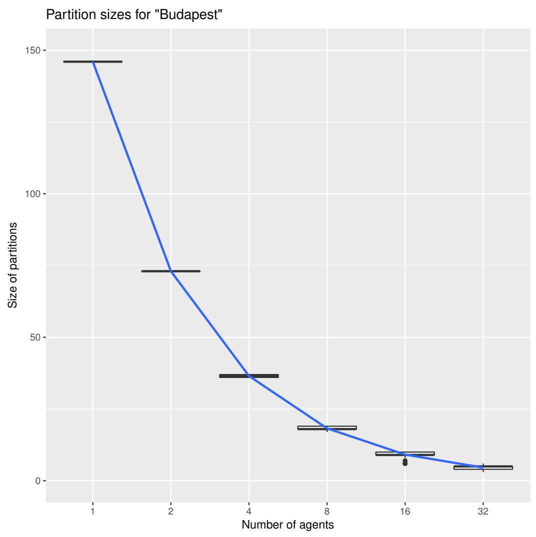

Fig. 4: Partition sizes for Budapest

factor the intentions of other agents in when making a decision which node they visit next. The third was PBPS, which forms partitions on the map in a self-organized manner and agents perform patrolling on their own partition only.

VI. Partitioning algorithm results

The goal of this experiment was to gain insight into the performance characteristics of the partition forming part of the PBPS algorithm. To this end we have disabled the mobility of the agents and tracked the partitioning events throughout the simulation. Partitioning events are either node free or nonfree node claims, or partition resets. In the beginning of every simulation run, only one agent is on the map at a randomly chosen node and it starts forming its partition. After this initial agent claims the whole map as its partition there are no more partitioning events occurring. The simulator waits until there are no more partitioning events for 5000 time steps and then doubles the number of agents on the map. The agents already on the map retain their partitions and the new agents are assigned a random starting node as their initial partition. This mechanism for adding agents sometimes cause overlaps in the partitioning, because the previous generation of agents claimed the whole map together (every node belongs to one and only one agent). However this is not a problem as the first time a new agent broadcasts its initial partition, the overlaps will be resolved by the old agent releasing that node from its partition.

The number of agents simulated on each map are exponentially increasing $(1, 2, \ldots, 16)$. The exponential increment in

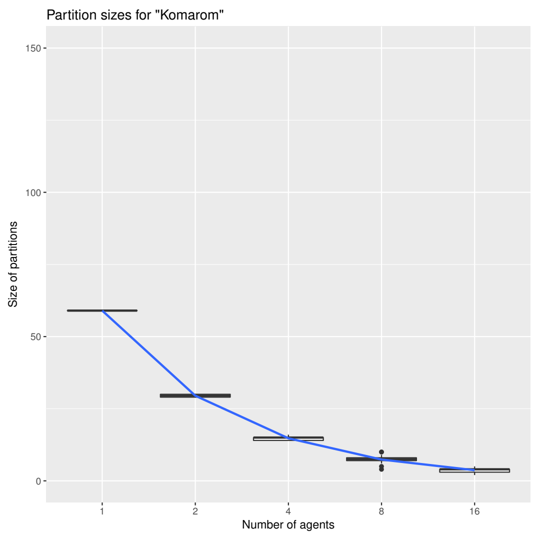

Fig. 5: Partition sizes for Komárom

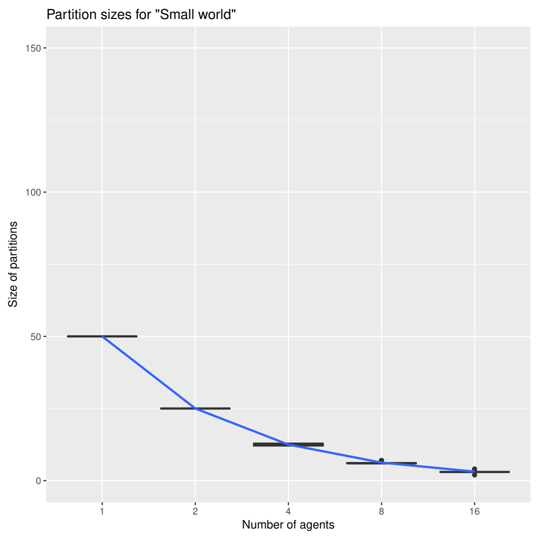

Fig. 6: Partition sizes for Small World

the number of agents can reveal the scalability characteristics of the partition forming part of PBPS with respect to the different maps. As a performance metric, the time to arrive at a stable partitioning was measured for different number of agents $(1, 2, 4, \ldots, 16)$ and at the same time the number of nodes in each partition were tracked.

This was done in order to confirm whether a stable partitioning is also a balanced one. A balanced partitioning in this terminology is where the number of nodes in each agent's partition is close to the $\frac{number\_of\_nodes}{number\_of\_agents}$ number. In other words, a balanced partitioning is where each partition has a roughly equal amount of nodes. Depending on the structure of the map this balanced property might not be achievable, since the tested maps are not full graphs with an edge connecting every node. Figures 4, 5 and 6 report the distribution of

Num. of agents	**1**	**2**	**4**	**8**	**16**	**32**
			Budapest belváros			
Mean	146.0	73.0	36.5	18.25	9.125	4.5625
Var	0.0	0.0	0.2564	0.3670	0.6383	0.3534
IDI	0.0	0.0	0.0070	0.0201	0.0699	0.0774
			Komárom			
Mean	59.0	29.5	14.75	7.3750	3.6875	N/A
Var	0.0	0.2631	0.5000	1.1487	0.3671	N/A
IDI	0.0	0.0089	0.0338	0.1557	0.0995	N/A
			Small world			
Mean	50	25	12.5	6.25	3.125	N/A
Var	0.0	0.0	0.2564	0.1898	0.1729	N/A
IDI	0.0	0.0	0.0205	0.0303	0.0553	N/A

TABLE II: Mean, variance and index of dispersion values of resulting partition sizes for all three tested maps and different number of agents.

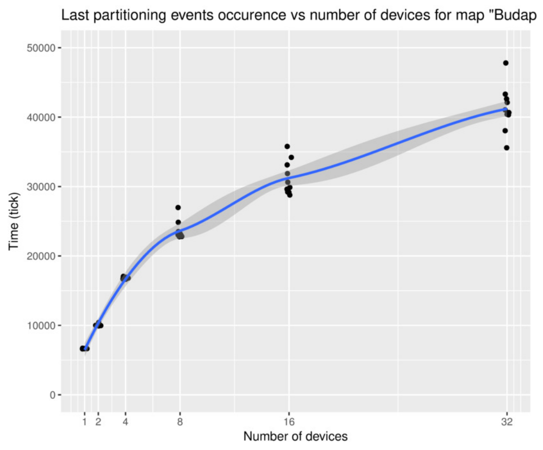

Fig. 7: Times of last partitioning events for versus number of agents for Budapest

partition sizes versus the number of agents. These are box plots[22] where the box contains 75% of the observed values. The blue curve shows the balanced partition sizes for different number of agents, its value is $\frac{number\_of\_nodes}{number\_of\_agents}$, where $number\_of\_nodes$ is constant and a property of the map. The results here show that in all simulated experiments the partitioning algorithm was able to arrive at solutions close to the optimal. Some exceptions show up as outliers on the box plots and indicate that the initial conditions have some influence on the final partitioning. All simulations were done 10 times with different starting positions for the agents, therefore each box represents the distribution of $10 * number\_of\_agents$ different resulting partition sizes.

Additionally the mean, variance and index of dispersion[23] metrics were calculated for the three different maps and for the different agent numbers. These results are summarized in Table II.

The next set of results discuss the time requirements of the partitioning algorithm, paying attention to its scaling properties. The question here is how does adding more agents to a map influence the time needed to arrive at a stable configuration? A stable configuration is one, where no more

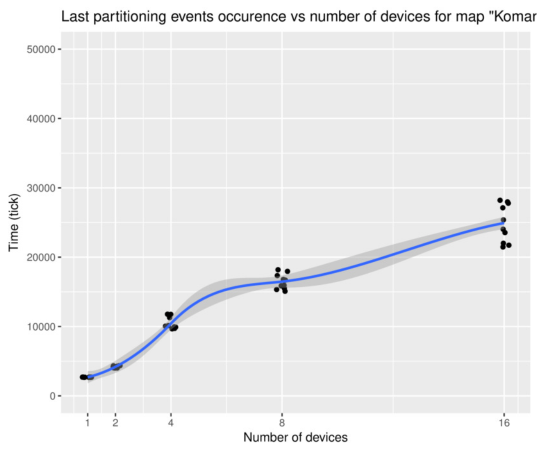

Fig. 8: Times of last partitioning events for versus number of agents for Komárom

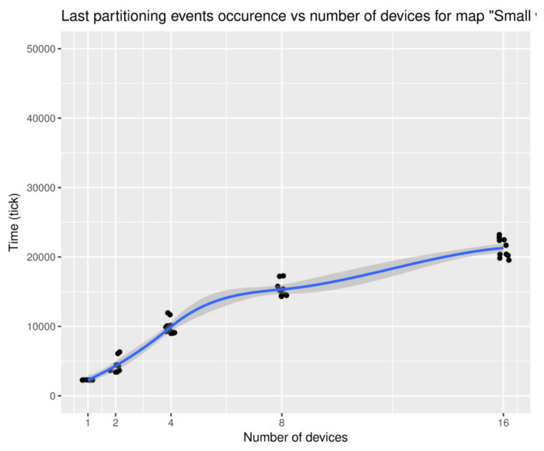

Fig. 9: Times of last partitioning events for versus number of agents for Small World

partitioning events occur. One of the advantages of using a self-organized approach is that in theory it is scalable and can enable multi-agent patrolling on larger graphs for a larger number of agents, than centralized algorithms. Figures 7, 8 and 9 show the distribution of time of the last partitioning event for each different simulated agent population versus the number of agents across all 10 iterations. Each group of values consists of 10 measurements and a curve is fitted to show the observed general trend of the values. It is important to note that the 5000 time step period between agent additions is subtracted from the values on the figures to show an ideal situation where one could determine the exact moment when the partitioning stops. The figures show that although the number of agents increase exponentially, the time needed to partition the graph increases only linearly. This indicates, that for the tested maps, the algorithm behaves well with a large number of agents and might be suitable to handle very large number of agents partitioning the same graph simultaneously. The variance of the values increase for larger number of devices. This indicates that the intermediate resulting partitionings and randomly assigned starting nodes for added agents (initial condition for each addition step) has an increasingly large effect when the number of agents are high. It is possible that by carefully choosing the insertion point for new agents, the observed variance might be decreased. However in this analysis we were interested in the possible outcomes and was aiming of getting a full picture of the dynamic behavior PBPS, rather than optimizing its parameters for ideal performance.

VII. Patrolling results

After investigating the different aspects of the partitioning, we compared the PBPS algorithm with two other algorithms: Conscientious Reactive (CR) and State Exchange Bayesian Strategy (SEBS). Conscientious Reactive (CR) is a simple greedy reactive algorithm, where the agent makes decisions based only on the visits made by itself. Other agents are not taken into account, therefore no communication is needed by this algorithm. Upon arriving to a node it resets the node's idle timer in its own local data structure and evaluates the node's neighbors in the graph. For the next node to visit the agent chooses the one with the largest instantaneous idle time value. CR is a trivial algorithm that requires only an agent with the capability to move and store the graph and timers in its internal memory, therefore the algorithm is an ideal baseline to compare other, more sophisticated algorithms to.

The State Exchange Bayesian Strategy (SEBS) strategy takes into account the interference caused by other agents in the same area. Agents can inhibit each others' movements by standing in the way or forcing other agents to move slower to avoid collision. This effect can cause strategies like CR and GBS to perform poorly in situations, where the number of agents in an area becomes high. GBS and SEBS builds on the idea of Bayesian decisions, where the a priori probability to visit a neighbor node is offset by the instantaneous idleness of the node known to the agent. This way GBS can take into account priorities input by the designer and the actual state of the system (instantaneous idleness of the nodes). SEBS extends GBS by having agents broadcast their intentions (the node they are intend to visit next) for their immediate neighbors. This information is used in all agents when evaluating the agent's next node to visit. If one or more agents intend to visit the same node, it is not beneficial to let another one go in that same direction, even if the instantaneous idleness value of the node is high. SEBS aims to lower the amount of time an agent loses by executing collision avoidance behaviors, causing them to slow down or chose a different target node altogether.

In these simulation runs, each group size (number of agents) ran for 200000 time steps, and after the number of agents were doubled. This process was repeated until there were 16 agents for Komárom and Small World, and 32 agents for Budapest on the map. We ran 10 iterations for each algorithm-map pair with random starting positions for every agent for every iteration. The structure of the graphs and the starting position of the agents can influence their measured performance, therefore in

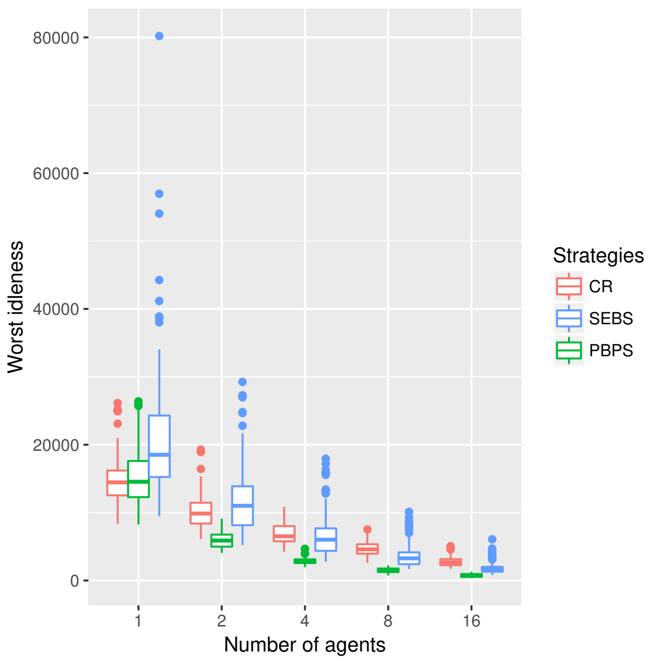

Fig. 10: Worst idleness time distributions for different strategies versus number of agents for Budapest

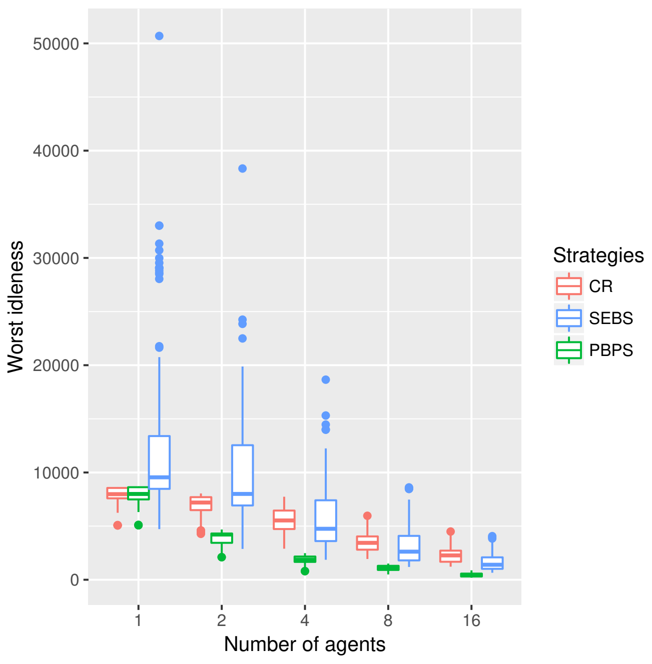

Fig. 11: Worst idleness time distributions for different strategies versus number of agents for Komárom

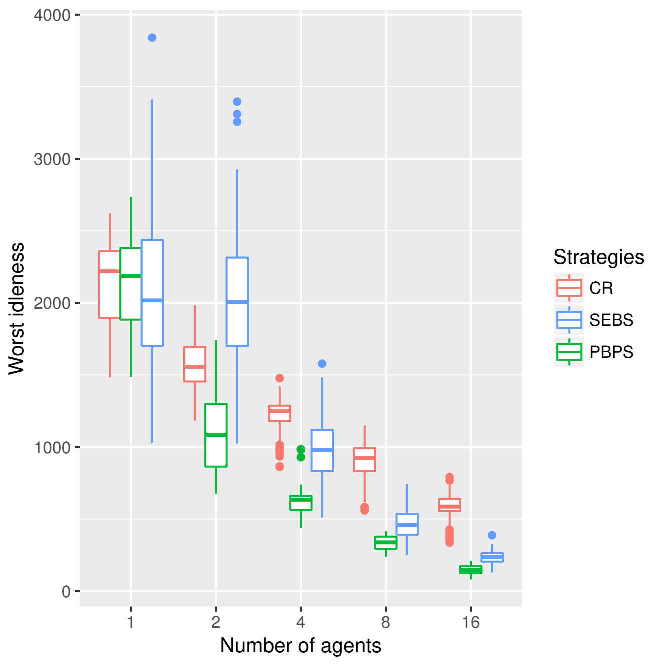

Fig. 12: Worst idleness time distributions for different strategies versus number of agents for Small World

order to gain as much information about the performances of different algorithms as possible, random non-overlapping starting points were chosen for the agents when they were added to the team. In the case of CR and SEBS strategies this means a random node from the graph and in the case of PBPS, this results in a random node from the agent's partition.

Partitioning strategies have a tendency to perform worse when they start on a graph, because they base their decisions on idleness times (and other agents' intentions in the case of SEBS), but in the beginning this data is not available for them. Therefore the first 100000 time steps for each different configuration were dropped in order to let the agents "warm up" and the worst idleness times for each node were collected for the last 100000 of each configuration. Figures 10, 11 and 12 report the worst idleness time distribution measured on the whole map for different strategies and different number of agents.

For one agent, CR and PBPS are almost identical, this is caused by the fact that PBPS employs CR as the patrolling strategy in the agents' partitions. In all tested configurations the performance of PBPS were better than the other tested algorithms. Comparing worst idleness distributions between SEBS and PBPS reveals that PBPS requires on average half the amount of agents to achieve the same performance as SEBS. In the case of map Komárom this effect is even more pronounced as the required number of agents for PBPS is one fourth of that for SEBS.

Another interesting facet is that SEBS tends to have a distribution with a longer tail than CR and PBPS. This means that the majority of values are larger than the mean for maps

Budapest and Komárom (see in Figures 10 and 11), which indicates that judging the performance of SEBS by the mean of its worst idleness times can be misleading, by indicating that the average time between visits to nodes in the map is smaller than what can be interpreted from the distribution of the values. This property means that performance metrics reported in articles like [8] can be only part of the whole picture as one cannot judge every aspect of the performance of the algorithms accurately from the average idleness values.

VIII. Conclusion

We have elaborated on the performance characteristics of the PBPS algorithm first published in [18]. The result of the partitioning algorithm is a set of subgraphs formed by the individual agents cooperating with each other that is used by patrolling agents as their patrolling tasks. These subgraphs are formed cooperatively by the agents on the graph without any central help or control. Using subgraphs of the original graph allows to completely avoid interference between agents, therefore have them performing in a parallel and deterministic manner. The cost function used in this article was the difference in number of nodes between partitions, but changing this cost function opens up possibilities for designers of patrolling systems to tailor the resulting partitions to their use case.

Simulations were performed on three different maps, representing different environments: the inner city of Budapest with shorter edges and more neighbors, part of the small city Komárom with longer edges and fewer neighbors and a random graph, which is a generated small world type graph (Barabási Albert graph). PBPS arrived at a balanced partitioning in every simulation run with low variance in the number of nodes assigned to an individual agent. Moreover PBPS scales well with the number of agents, meaning that the time needed to finish partitioning increased sublinearly with the number of agents tested. This property is in contrast with planning approaches, that usually have worse scaling properties, making it viable to employ this strategy on larger graphs and with a larger amount of agents.

We compared the performance of agents using different patrolling strategies, such as CR, SEBS and PBPS. Patrolling performance was judged by the distribution of worse idleness times for all nodes in the graph, represented as box plots. In every tested case PBPS came out ahead, and on two maps the necessary number of agents for PBPS were half of what was required by CR and SEBS for the same performance. This results in a tradeoff decided by the designer or implementor later: either employ the same amount of agents and get lower worst idleness times throughout the graph, or in a resource constrained scenario, change the patrolling strategy and get the same performance level with only half the agents required by other strategies.

References

[1] Gonçalo Cabrita, Pedro Sousa, Lino Marques, and A de Almeida. Infrastructure monitoring with multi-robot teams. In *Proceedings of the 2010 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS*, pages 18–22, 2010.

[2] David Portugal and Rui P Rocha. Cooperative multi-robot patrol in an indoor infrastructure. In *Human Behavior Understanding in Networked Sensing*, pages 339–358. Springer, 2014.

[3] Yann Chevaleyre. Theoretical analysis of the multi-agent patrolling problem. In *Intelligent Agent Technology, 2004.(IAT 2004). Proceedings. IEEE/WIC/ACM International Conference on*, pages 302–308. IEEE, 2004.

[4] Talita Menezes, Patrícia Tedesco, and Geber Ramalho. Negotiator agents for the patrolling task. In *Advances in Artificial Intelligence-IBERAMIA-SBIA 2006*, pages 48–57. Springer, 2006.

[5] David Portugal and Rui Rocha. Msp algorithm: multi-robot patrolling based on territory allocation using balanced graph partitioning. In *Proceedings of the 2010 ACM Symposium on Applied Computing*, pages 1271–1276. ACM, 2010.

[6] Jean-Samuel Marier, Camille Besse, and Brahim Chaib-Draa. Solving the continuous time multiagent patrol problem. In *ICRA*, pages 941–946. Citeseer, 2010.

[7] Alessandro Almeida, Geber Ramalho, Hugo Santana, Patrícia Tedesco, Talita Menezes, Vincent Corruble, and Yann Chevaleyre. Recent advances on multi-agent patrolling. In *Advances in Artificial Intelligence–SBIA 2004*, pages 474–483. Springer, 2004.

[8] David Portugal and Rui P Rocha. Distributed multi-robot patrol: A scalable and fault-tolerant framework. *Robotics and Autonomous Systems*, 61(12):1572–1587, 2013.

[9] Pooyan Fazli, Alireza Davoodi, and Alan K Mackworth. Multi-robot repeated area coverage. *Autonomous robots*, 34(4):251–276, 2013.

[10] Yoav Gabriely and Elon Rimon. Spanning-tree based coverage of continuous areas by a mobile robot. *Annals of mathematics and artificial intelligence*, 31(1-4):77–98, 2001.

[11] Tiago Sak, Jacques Wainer, and Siome Klein Goldenstein. Probabilistic multiagent patrolling. In *Brazilian Symposium on Artificial Intelligence*, pages 124–133. Springer, 2008.

[12] Ruben Stranders, E Munoz De Cote, Alex Rogers, and Nicholas R Jennings. Near-optimal continuous patrolling with teams of mobile information gathering agents. *Artificial intelligence*, 195:63–105, 2013.

[13] François Sempé and Alexis Drogoul. Adaptive patrol for a group of robots. In *Intelligent Robots and Systems, 2003.(IROS 2003). Proceedings. 2003 IEEE/RSJ International Conference on*, volume 3, pages 2865–2869. IEEE, 2003.

[14] Oswaldo Aguirre and Heidi Taboada. An evolutionary game theory approach for intelligent patrolling. *Procedia computer science*, 12:140–145, 2012.

[15] Burcu B Keskin, Shirley Rong Li, Dana Steil, and Sarah Spiller. Analysis of an integrated maximum covering and patrol routing problem. *Transportation Research Part E: Logistics and Transportation Review*, 48(1):215–232, 2012.

[16] Charles Pippin, Henrik Christensen, and Lora Weiss. Performance based task assignment in multi-robot patrolling. In *Proceedings of the 28th annual ACM symposium on applied computing*, pages 70–76. ACM, 2013.

[17] Cyril Poulet, Vincent Corruble, and Amal El Fallah Seghrouchni. Working as a team: using social criteria in the timed patrolling problem. In *Tools with Artificial Intelligence (ICTAI), 2012 IEEE 24th International Conference on*, volume 1, pages 933–938. IEEE, 2012.

[18] Bernát Wiandt, Vilmos Simon, and András Kőkuti. Self-organized graph partitioning approach for multi-agent patrolling in generic graphs. In *Smart Technologies, IEEE EUROCON 2017-17th International Conference on*, pages 605–610. IEEE, 2017.

[19] David Portugal, Charles Pippin, Rui P Rocha, and Helen Christensen. Finding optimal routes for multi-robot patrolling in generic graphs. In *Intelligent Robots and Systems (IROS 2014), 2014 IEEE/RSJ International Conference on*, pages 363–369. IEEE, 2014.

[20] Geoff Boeing. Osmnx: New methods for acquiring, constructing, analyzing, and visualizing complex street networks. *Browser Download This Paper*, 2016.

[21] Aydano Machado, Geber Ramalho, Jean-Daniel Zucker, and Alexis Drogoul. Multi-agent patrolling: An empirical analysis of alternative architectures. In *Multi-Agent-Based Simulation II*, pages 155–170. Springer, 2002.

[22] Robert McGill, John W Tukey, and Wayne A Larsen. Variations of box plots. *The American Statistician*, 32(1):12–16, 1978.

[23] DR Cox and PAWL Lewis. The statistical analysis of series of events. 1966.