Computer Assisted Methods in Engineering and Science
29(4): 379–395, 2022, doi: 10.24423/cames.581

Multi-Robot Coverage with Reeb Graph Clustering and Optimized Sweeping Patterns

Jacek SZKLARSKI

Institute of Fundamental Technological Research, Polish Academy of Sciences, Warsaw, Poland; e-mail: jszklar@ippt.pan.pl

In order to perform mapping, inspecting, searching, painting, cleaning, and other similar tasks, mobile robots have to act according to a coverage plan. Finding a trajectory that a robot should follow requires an appropriate coverage path planning (CPP) algorithm and is a non-trivial problem, especially if a cooperating group of robots is considered. We propose that the multi-robot CPP can be solved by: decomposing the input occupancy grid map into cells, generating a corresponding Reeb graph, clustering the graph into N_r clusters, and solving the associated equality generalized traveling salesman problem in order to obtain optimal back-and-forth sweeping patterns on the clusters. This last step has been proven to be one of the most efficient ways to find trajectories for a single robot [5]. The discussed approach is motivated by a specific application: industrial cleaning of large warehouses by N_r autonomic mobile cleaners (the cleaning radius of a robot is much smaller than the area to be cleaned). The total time required for cleaning is to be minimized. By means of statistical analysis, using an extensive, realistic set of synthetic maps, it is shown that the proposed algorithm meets the criteria for applying it in the production process.

Keywords: coverage path planning, motion planning, multi-robot.

Copyright © 2022 J. Szklarski
Published by IPPT PAN. This work is licensed under the Creative Commons Attribution License CC BY 4.0 (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

One of the fundamental applications of autonomous robotic platforms is to perform coverage tasks: area inspection, surveillance, creating image mosaics, mowing, harvesting, planting, spraying, mapping, searching, painting, etc. Such problems require that a mobile robot or – more generally: a tool (e.g., CNC machines) – at some time will come upon a certain region of the environment, eventually visiting, or observing, the entire region of interest. In order to do so, the robot must act according to a CPP algorithm. The problem of CPP is well-known and well-studied in the field of robotics [1, 23]. Nevertheless, it remains

challenging and even the simplest variants, such as the lawnmower problem, are NP-hard.

In order to practically speed up the process of covering, a number N_r of robots may be performing the task in a cooperative manner. This leads to a specific version of the CPP problem: multi-robot CPP. There are many ways to approach this task, depending on a specific application. For example, whether planning is realized online or offline, if the environment map is available and known in advance to all the robots, what are the means of communication between the covering units, whether the environment dynamically changes during coverage. Additionally, it is important to consider scales involved in a particular scenario: the ratio of the area covered by a robot vs. the total area to be covered, and time constraints on the path generation, etc.

In this paper, we focus on an N-robot CPP problem of relevant cleaning large warehouses by a fleet of industrial cleaning robots. The main contribution of this work is the CPP algorithm itself, consisting of graph clustering and sweeping patterns optimization. The algorithm is designed for its real-world application having in mind the specific characteristics of the environment: the area to be cleaned is supposed to be much larger than the cleaning radius; the underlying static map of the environment is known in advance (this is done by performing three-dimensional (3D) scan prior to deploying devices), sparse dynamical obstacles are allowed (e.g., people moving in the warehouse), and real robots

Fig. 1. Photo of a prototype of the cleaning robot being developed by our industrial partner, the United Robots company. The robot is equipped with sensors such as LIDARs and RGBD cameras, which ensure that all safety measures are taken into account, and the accurate position in the global frame is known. The photo is for illustrative purposes only, and the details regarding robot control and navigation are outside of the scope of this work.

can localize themselves and follow a given path with the accuracy of the order of 10 cm. The details regarding sensors, robots and their navigation stack are outside of the scope of this work and will be presented elsewhere.

The paper is organized as follows: first, the CPP problem is defined, including its multi-robotic variant. Then it is discussed how it can be approached, bearing in mind the particular problem of warehouse cleaning. Essentially, this is done by decomposition of an environment grid map into polygons (cells), generating an associated Reeb graph, clustering it and then generating N_r path plans by one of the state-of-the-art methods [5]. Next, by means of numerical simulations, the method is evaluated on an actual obstacle map and a large realistic synthetic dataset [15]. It is demonstrated that the total coverage time t_{cov} scales on average as $t_{\mathrm{cov}} \propto N_r^{-0.86}$ which is satisfactory close to the theoretically perfect scaling $t_{\mathrm{cov}} \propto N_r^{-1}$. Finally, some conclusions are presented and directions for further work are pointed out.

2. The coverage path planning problem

The CPP problem is defined as finding a path for a mobile robot, which ensures that the device will visit/observe all points of a given area or volume. The robot should also avoid obstacles and collisions. The usual requirement is that the complete coverage is guaranteed and realized in the fastest possible way (which often means minimizing any path overlapping). CPP is closely related to other classical problems, such as the lawnmower problem, the watchman route problem, the gallery problem, the covering salesman problem, or even a wide class of search games on graphs such as cops & robber games [22]. Some of these problems – in their simplest variations – can be solved in a polynomial time, but generally, they are NP-hard (e.g., [2]).

CPP can be divided into two main classes: offline and online. This paper is focused on offline tasks that rely on stationary information and where the environment is assumed to be known in advance. Contrary to this, online tasks utilize real-time measurements without any prior knowledge of the environment (such a sensor-based coverage is closely related to the problem of simultaneous localization and mapping – SLAM – with exploration). There exist a large number of exact, approximate and heuristic algorithms to solve many variations of both types of CPP, see [7, 11, 21]. The vast majority of the research is focused on minimizing the time to complete coverage in deterministic systems with perfect information (for multi-robotic CPP this often comes down to the equal path-length distribution). Examples of a non-standard optimization goal include: minimizing energy consumption during autonomous landmine detection [8], and minimizing the coverage time by taking into account speed variations while turning or by considering travel times instead of distances [17].

Using multiple robots in the CPP task has some obvious advantages [23]: decreased time to complete coverage, improved robustness, failure tolerance, and possibly smaller localization errors. However, this comes at a cost: the need for communication, synchronization, handling mutual observations or possibly heterogeneous devices, and specialized algorithms for CPP (updating poses, maps, loop closing, etc). Possible applications range from two-dimensional (2D) boundary inspections of rotor blades inside a turbine [9], through efficient surveillance and mapping [19], to complex 3D coverage realized by a group of UAVs [20].

Bearing in mind the requirements outlined above, our CPP should meet the following conditions:

1. The mobile robots move on a 2D surface and their state is defined by the position and rotation, i.e., the vector $\mathbf{x} = (x, y, \phi)$.
2. The number of robots involved in the cleaning process is assumed to be between 1 and 5, $1 \leq N_r \leq 5$.
3. The cleaning radius of the machines is $r_{\text{cov}} = 1$ m.
4. The total area to be cleaned A is of the order of 10^5 m^2.
5. Each point of the area A must be covered by any robot at least once.
6. The robots move along preplanned plans, that is, trajectories being sets of consecutive vectors $\mathbf{x}_0, ..., \mathbf{x}_n$.
7. The maximum velocity, the maximum acceleration/deceleration and the maximum angular velocity are given, $v_{\max}$, $|a_{\max}|$, $\dot{\phi}_{\max}$, respectively.
8. Between any two consecutive points $\mathbf{x}_i$ and $\mathbf{x}_{i+1}$, a robot moves in a straight line accelerating until it reaches $v_{\max}$ (or decelerating to 0 if a turning point is being approached).
9. Should a robot account for an unexpected obstacle (static or dynamic), that is, an obstacle not present on the static map, it should attempt to avoid collision and go back to its trajectory as soon as possible.
10. If a robot should fail, all the remaining devices will be informed about this by means of Wi-Fi communication and will be able to follow preplanned path plans for smaller N_r.
11. Should the static environment change (rearrangements in a warehouse), the map and planned trajectories will be updated for all the devices.
12. The cost metric to be minimized is the total cleaning time.

It is assumed that the probability of encountering expected obstacles is spatially homogeneous for the entire environment. Therefore it is not taken into account in the path planning process since the average expected delays are the same for each robot and independent of its trajectories.

It should be noted that it is possible that the environmental circumstances enforce overlapping of the planned paths for different robots. This can lead to

a possible collision between the operating units. Such collisions are not taken into account during the path-planning stage itself. Should such a collision occur, the other robot is simply treated as any other dynamical obstacle, and the collision is avoided by means of a variant of a virtual force field based avoidance navigation algorithm (e.g., [24]).

Regarding communication, it is assumed that the units can communicate by means of a centralized all-to-all channel (a single Wi-Fi network) and an additional central server is available. Should a unit stop responding or should it report a failure, new path plans will be distributed among the units for the reduced number of robots. Such event is extraordinarily rare, therefore dynamic replanning (i.e., replanning that takes into account the total area already covered in a cleaning cycle) is not considered and not taken into account in the CPP algorithm.

3. Solving multi-robot CPP

Input to the proposed algorithm is a 2D grid map M representing the environment with obstacles and the number of robots N_r together with their initial locations $\mathbf{x}_{0,N=1,\ldots,N_r}$. Throughout the paper, the starting point is assumed to be the same for all the robots: $(0, 0)$. Inevitably, this leads to some insignificant path overlapping. However, searching for optimal starting points (different for each robot) would lead to a different problem formulation. The choice of starting positions is justified since, in reality, all the units start operating from the same base station (which also serves as a clean water reservoir and power charging station).

From our assumptions, it follows that $r_{\text{cov}} \ll A$. This is not always the case for CPP problems. For example, photo mosaicking or visual searching may have a much larger r_{cov} than the size of the robots, e.g., UAVs equipped with cameras. The assumption of $r_{\text{cov}} \ll A$ basically precludes usage of any grid-based approach, since for most grid-based methods, the time for finding a solution grows significantly with the grid size (for many methods even exponentially [23]). For the discussed problem, the number of grid cells would be too large to come up with a feasible solution.

Consequently, we implement a solution based on the geometric decomposition of the grid map into a set of polygons – called *cells* – which will be covered optimally. This can be realized by decomposing the grid map into trapezoids (i.e., *trapezoidal decomposition*), after which simple back-and-forth movements are generated for each trapezoid. The main drawback of this simple decomposition is the fact that it generates only convex polygons and therefore results in a large number of polygons and suboptimal sweeping patterns, see Fig. 2. A possible well-known alternative is the so-called *boustrophedon cellular decomposition – BCD*, which also generates a non-convex cells, that can be also covered only by

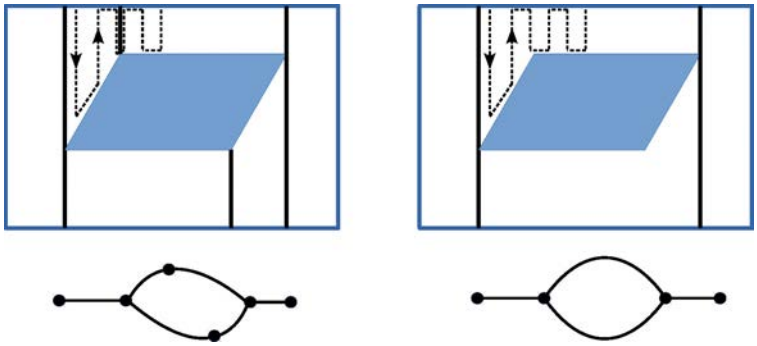

Fig. 2. An example of trapezoidal and boustrophedon decompositions into cells with their Reeb graphs (bottom). The latter decomposition allows for a better sweeping pattern fit (the dashed line).

simple zig-zag-like motions. This results in the better fitting of the zig-zag like motions. It should be noted that more sophisticated decompositions have been proposed, e.g., [16]. Nevertheless, since in the discussed case, the optimization is done at a later stage, we use the straightforward trapezoidal or BCD. All the results presented here are with BCD, and our research shows that it generally performs slightly better than the trapezoidal decomposition. After decomposition, a set of cells is obtained where the geometrical relation between them may be represented by a graph (a Reeb graph, in particular). Consequently, the problem may be solved with the help of existing solutions known from graph theory.

In the literature, there exist a number of proposed solutions for multi-robot CPP with cellular decomposition [1]. For example, [18] minimizes repeated coverage by using a dedicated auctioning mechanism and utilizing communication (restricted) between the units. The authors of [13] reformulate the CPP into a problem of finding minimal flow in the network where nodes correspond to trapezoids. Another solution was proposed by [4], where the domain is divided into hexagons that are then assigned to each robot (UAVs in this case) and each hexagon is covered by means of predefined patterns. In [3], the authors present a two-stage coverage approach in which a graph, based on decomposition, is constructed together with associated sweeping patterns. Afterward, a vehicle routing problem is solved in order to equally distribute the task between UAVs. However, this solution works only for maps without obstacles (i.e., grid maps that correspond to a general polygon but without holes). Another load-distributing solution with BCD was developed by [14]. The authors compare two distinct approaches: 1) solving the min-max k-Chinese postman problem on the entire (Reeb) graph with $k = N_r$, and 2) clustering the graph into N_r sets with equal expected coverage times, then finding the Euler cycle for each cluster.

The solution proposed here utilizes load balancing by clustering of the graph into N_r sets $C_{1,\ldots,N_r}$ of cells, and then searching for trajectories minimizing the coverage time for each cluster, see Algorithm 1. The coverage time for each cluster of cells depends on the order of cell visiting and entry point of the robot into each cell. Minimizing the time is an NP-hard problem and the result is found by solving the associated variant of a traveling salesman problem (TSP) first proposed by [5] for a single UAV. This works for general maps that may be represented as a general polygon with holes (PWH).

Algorithm 1. Outline of the path planning algorithm for coverage with N_r robots.

```
function FINDPLANS(M, N_r)        ▷ M is the 2D raster grid map of the environment
  G ← DECOMPOSE(M)                          ▷ Decompose map into cells (with BCD)
  C_{1,...,N_r} ← CLUSTERIZE(G)             ▷ clusterize G into N_r disjoint sets of cells
  for i ← 1, N_r do
    P_i = FINDOPTIMUMCOVERAGE(C_i)          ▷ Path for each cluster, minimize time
  end for
  return P
end function
```

3.1. Area clustering

As stated above, the input grid map with obstacles is transformed into a set of cells (convex polygons) by means of a chosen decomposition method (BCD). This decomposition can be represented in a particular type of adjacency graph called a Reeb graph $G = (V, E)$. In such a graph each **edge** $e \in E$ corresponds to **a cell** and has an associated cost that is the time required to cover the polygon. Nodes V represent critical points where connectivity changes, see Fig. 2, bottom.

Having the Reeb graph G, it is next divided into N_r disjoint clusters $C_{1,\ldots,N_r}$ by means of a variant of breadth-first-search (BFS) traversal. The cluster coverage time $t(C_n)$ is defined as the sum of the *expected* coverage times $t(e_i)$ associated with each cell (i.e., graph edge), and the graph coverage time $t(G)$ is defined as the sum of times required to cover all clusters. The clustering schema is summarized in Algorithm 2.

Starting with the robot $i = 1$, set time cost for robot i: $t_i = 0$. If a robot i is not in an unvisited edge (that is, in a cell), move it to the nearest edge k respecting obstacles and add travel time to the cost t_i. If the expected coverage time, including e_k, for robot i is larger than T/N_r (i.e., longer than the equal time balancing), the cell is divided into two parts such as $t(C_i) = T/N_r$. Otherwise, graph G is BFS traversed until each robot gets assigned an equal expected coverage time. The above procedure is repeated until changes in T are smaller than

Algorithm 2. The Reeb graph clustering algorithm.

function CLUSTERIZE($G = (V, E), N_r$)
  ▷ G is the Reeb graph representing the environment
  $T \leftarrow t(G)$ ▷ T is the expected coverage time for the entire G
  $C_{1,\ldots,N_r} \leftarrow \emptyset$ ▷ Start with empty clusters
  `visited`$(e_i \in E) \leftarrow$ `False`
  $t_{1,\ldots,N_r} = 0$ ▷ Set time costs to 0
  repeat
    for $i \leftarrow 1, N_r$ **do**
      $e_k \leftarrow$ edge where robot i is located
      if `visited`(e_k) **then**
        $e_k \leftarrow$ the nearest unvisited edge
        $t_i \leftarrow t_i +$ `travel time to` e_k
      end if
      repeat
        if $t_i + t(e_k) < T/N_r$ **then**
          $C[i] \leftarrow C[i] \cup e_k$
          `visited`$(e_k) \leftarrow$ `True`
        else
          Divide $e_k = e_k^1 + e_k^2$ such as $t_i + t(e_k^1) = T/N_r$, modify G accordingly
          `visited`$(e_k^1) \leftarrow$ `True`
        end if
        $e_k \leftarrow$ BFSTRAVERSE_NEXTEDGE(G)
      until $t_i < T/N_r$ `and` $e_k \neq \emptyset$
    end for
    $T \leftarrow \sum_{i=1,\ldots,N_r} t_i$
  until $\Delta(T) < \epsilon$ ▷ Stop if converged
  return C
end function

a given threshold ϵ (here 0.1%). This accounts for necessary travel times between the initial robot poses and travel times between non-adjacent edges. This simple procedure quickly converges and makes the work-load more balanced between the units.

It should be noted that the above procedure also modifies the Reeb graph by dividing the cells in a way that makes the clustering better balanced in terms of equal t_i. Here, the expected times $t(e_i)$ denote the theoretical time that is

required to clean a square with an area equal to that of the cell associated with the edge i.

Having a set set of N_r clusters, the trajectory for each robot is found by solving a single robot i CPP on the set C_i. The union of all cells in C_i forms a set of – possibly disjoint – general polygons with holes. If PWHs are disjoint (i.e., there is more than one PWH for a single robot), trajectories between PWH are connected by means of the shortest distance routes between them while taking into account obstacles from the input map. While traveling between PWH, trajectories of some robots might overlap. This, however, cannot be avoided in a general case.

3.2. Generating trajectories for single robots

The goal of this procedure is to find a trajectory, i.e., a set of points which a robot i will visit sequentially covering all cells from the set C_i. Between the points, the robot will try to travel as fast as possible in a straight line while maintaining any safety measures (including possible dynamical obstacles). After arriving at its next point, the device may turn into another direction (i.e., change ϕ).

Due to the fact that in order to perform a turn, a robot must slow down, stop, then turn and again accelerate afterward, it is intuitive to seek for plans that minimize the number of turns. More formally, the cost function for a trajectory with N points is the total time:

$$T' = \sum_{i=1,\ldots,N} t'_i,$$

where

$$t'_i = \begin{cases} \sqrt{4d/a_{\max}}, & \text{for} \quad d < v_{\max}^2/a_{\max}, \\ 2v_{\max}/a_{\max} + \dfrac{d - v_{\max}^2/a_{\max}}{v_{\max}}, & \text{for} \quad d \geq v_{\max}^2/a_{\max}, \end{cases}$$

d being the 2D distance between $\mathbf{x}_i$ and $\mathbf{x}_{(i-1)}$. T' is going to be minimized. This is realized by choosing the appropriate order for visiting cells while considering combinations of sweeping directions for each cell on a graph. Practically it is done on a new graph, where each node holds information about all possible start and end vertices and all possible sweeping directions, i.e., directions parallel to the cell's walls. Figure 3 depicts some possible sweeping directions for a sample cell.

It can be shown that the problem defined in this way is equivalent to the equality generalized traveling salesman problem (E-GTSP), which is a variant

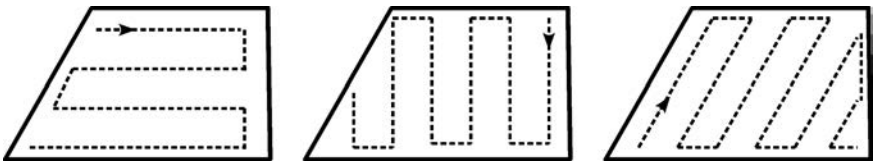

FIG. 3. For each cell, the coverage time is determined by sweeping direction and entry point into the cell (i.e., start and end vertex). The figure depicts some possible entry/exit points and directions for a sample trapezoid.

of the TSP, see [5]. E-GTSP is an NP-hard and exact solvers work only for small problems [10]. In our case, in order to solve E-GTSP the memetic solver proposed by [12] is used in accordance with the solution by [5] (the relevant code is publicly available). This is an efficient solver and is considered to be the state-of-the-art solution for E-GTSP.

3.3. An alternative approach: single trajectory splitting

In order to make a comparison, an alternative approach has been implemented. It is based on a single trajectory generated for the input map as would be the case for $N_r = 1$. Next, this trajectory is simply divided into N_r equal parts (equal in the sense of the time required to travel along each path). This trivial treatment of the multi-robot assumption is expected to work well particularly in simple environments with a small number of cells (thus, cells being large).

4. Results

The proposed multi-robot CPP algorithm described above is validated on a real world map and on an extensive, realistic synthetic dataset.

4.1. An underground garage

Figure 4, top, depicts a top-down view of a 3D point cloud composed of a set of LIDAR measurements performed in an underground garage. The point cloud is transformed appropriately in order to obtain a 2D obstacle map suitable for mobile robot planning and navigation. The mapping system and algorithms for point cloud processing have been presented in [6]. In the middle of the figure, the result of decomposition is depicted. In this particular case, the cellular decomposition is rather simple due to the regular geometry of the environment. Initially all the robots are placed at an unoccupied point closest to the lower-left corner of the gridmap. For the robot, a realistic parameters for the kinematic model were used: $a_{\max} = 0.3$ ms^{-2}, $v_{\max} = 1$ ms^{-1}. This corresponds to real

devices used for the cleaning tasks, see Fig. 1. A trajectory for a single robot is shown at the bottom of Fig. 4.

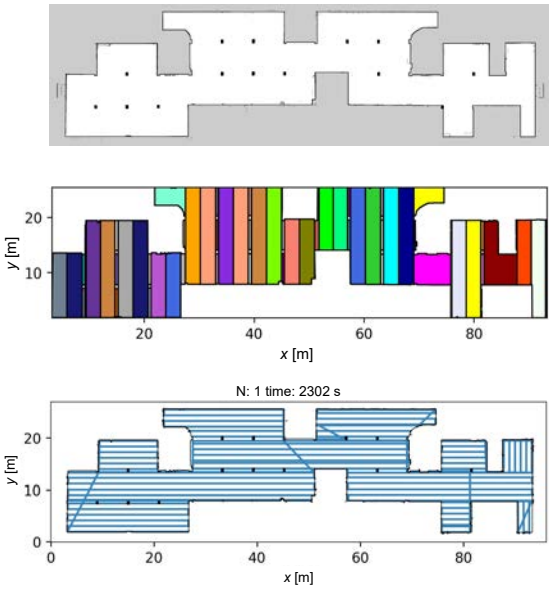

FIG. 4. Top: a gridmap representing an underground garage for robot motion planning and navigation. The pixel size is 3×3 cm. Middle: the corresponding cellular decomposition. Bottom: a trajectory for a single robot for the given decomposition and the E-GTSP optimization as described in Subsec. 3.2.

Figures 5 and 6 depict trajectories for $N_r = 2, 3, 5$ for both: the area clustering approach and the alternative one with simple single-path division. A typical problem with the splitting approach can be seen for $N_r = 3, 5$. Even for this simple map, clearly there are small separate areas to be covered by some robots. While the coverage time is only slightly worse, such fragmented coverage plans might be undesirable due to other practical means. This is even more evident for complex maps.

In order to measure effectiveness, we use utilization $U(T, N_r) = t_1/(T * N_r)$ with T being the total coverage time for N_r robots involved. For the trajectories in Figs. 5 and 6, the utilization is 0.94, 0.89, 0.81 and 0.85, 0.75, 0.72, respectively. It should be stressed that in any non-trivial obstacle map, the utilization cannot be perfect (i.e., equal to 1) since usually there are regions where trajectories for different robots must overlap. On the other hand, for specific maps, it may happen (although rarely) that the utilization is larger than 1. This is due to the fact that a single robot trajectory overlaps with itself which, in turn, may be avoided for the case with $N_r > 1$.

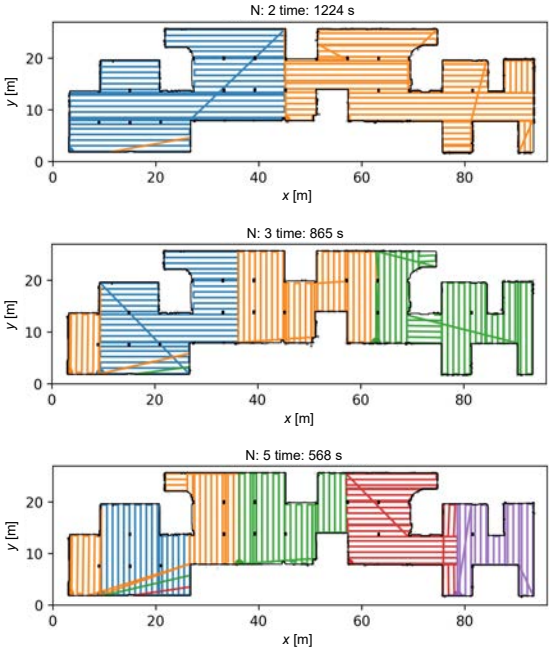

FIG. 5. Trajectories for robots with $N_r = 2, 3, 5$ for the garage environment as in Fig. 4 for the area clustering with the E-GTSP optimization approach. The total coverage time is also shown.

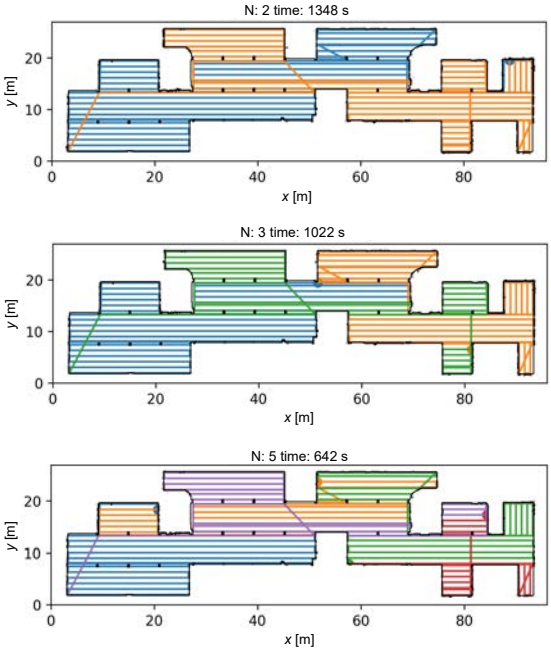

FIG. 6. Similar as Fig. 5 but for the alternative method with single path splitting, Subsec. 3.3.

4.2. Tests on a large synthetic map dataset

The authors of [15] have presented a way to generate realistic maps for testing algorithms in the field of mobile robotics. These synthetic layouts are thought to mainly represent indoor environments and – after proper scaling – can be used to statistically measure the effectiveness of the algorithm discussed in this paper. Figure 7 depicts some sample maps.

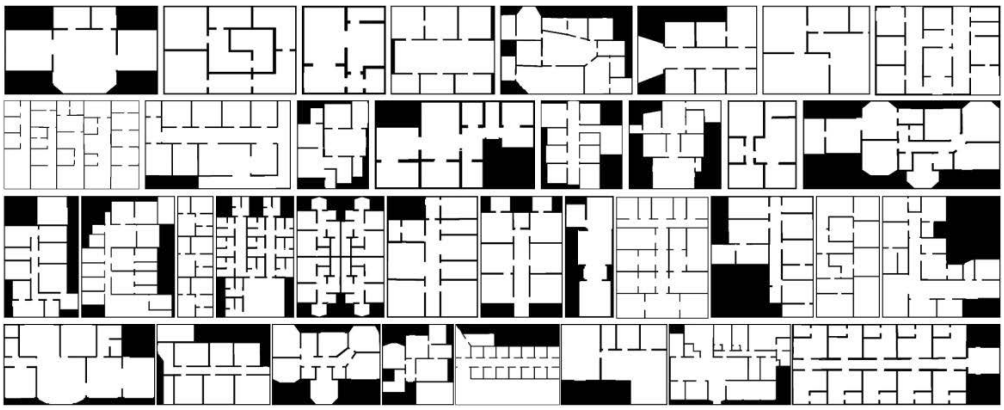

FIG. 7. Samples from a large set of 60k realistic layouts for testing robotic algorithms, after [15].

In order to obtain some valuable insights, 10^3 maps were selected randomly from the entire dataset and the multi-robot CPP run for $N_r = 1, ..., 5$ with the area clustering and the path splitting approach. The results are summarized in Table 1 and Figs. 8 and 9. The scales are chosen so that the cleaning time for a single robot is of the order of hours (between 1 and 20).

TABLE 1. Statistical values for utilization for various N_r and the two discussed methods for multi-robot CPP. Area decomposition gives better and more consistent (in the sense of lower σ) results.

N_r	Path splitting				Area decomposition			
	$\overline{U}$	$\widehat{\sigma}_2^2$	$U_{\min}$	$U_{\max}$	$\overline{U}$	$\widehat{\sigma}_2^2$	$U_{\min}$	$U_{\max}$
2	0.90	0.08	0.69	1.00	0.94	0.05	0.81	1.04
3	0.82	0.09	0.57	0.99	0.87	0.06	0.73	0.99
4	0.78	0.10	0.50	0.98	0.83	0.06	0.67	0.96
5	0.75	0.10	0.47	0.97	0.79	0.06	0.61	0.93

Additionally, a fitting of mean coverage time $t_{\text{cov}}(n) = n^{\alpha}$ has been performed. This gives $\alpha = -0.87$ for the area clustering method and $\alpha = -0.81$ for the one with splitting. These results are quite good, bearing in mind that with the larger number of robots, there must be some overlapping. Additionally,

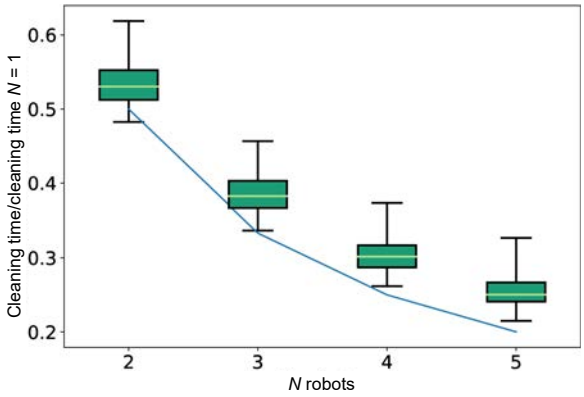

Fig. 8. Times for full coverage using area decomposition, 1 being the time required to clean by a single robot. The boxes show min, max, median and quartiles Q1 and Q3 for the tested set. The line denotes the perfect $1/N_r$ scaling.

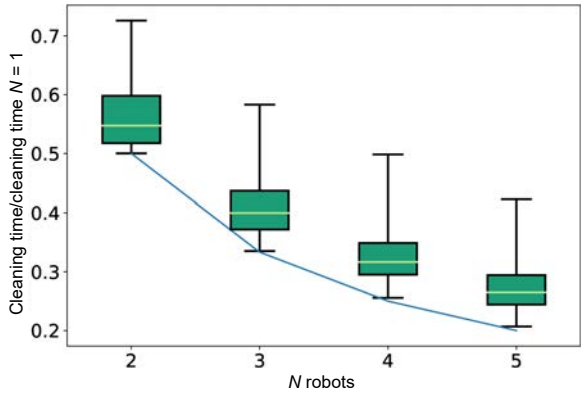

Fig. 9. As in Fig. 8 but for the alternative path-splitting method.

all the units start from the same base station, which inevitably leads to time inefficiency.

Considering the required CPU time to find the plans: it heavily depends on the map, i.e., its shape, number of small elements, holes, and of course, the number of cells after the decomposition. For the testing dataset, a typical desktop computer with i5 requires between 10 seconds up to about 1 minute for a map to complete the calculations.

5. Conclusions and future work

In this paper, a novel method for efficient multi-robot coverage path planning has been proposed. It is based on the clustering of the Reeb graph arising from

the cellular decomposition of the input grid-map with obstacles. Afterward, the optimization problem is solved, which looks for trajectories minimizing the total coverage time by analyzing permutations of sweeping patterns consisting of back-and-forth motions for all the cells from a set assigned to the robot i. This is realized by means of a memetic solver for the equivalent problem, E-GTSP. This approach has already been proven to be very reliable and efficient [5]. The real-world application is currently being tested by our industrial partner with real cleaning robots.

Future work will be focused on the inclusion of the possibility of multiple covering of certain regions of the map. This feature is important for the practical cleaning process since some areas may require multiple passes of the cleaner. Another extension that will be incorporated into the algorithms is a non-homogeneous fleet, i.e., robots with different covering radii and different kinematic properties.

Acknowledgements

The authors acknowledge the financial support of the National Centre for Research and Development (Poland), project POIR.01.01.01-00-0206/17 "Designing an autonomous platform which operates in an industrial production environment".

References

1. R. Almadhoun, T. Taha, L. Seneviratne, Y. Zweiri, A survey on multi-robot coverage path planning for model reconstruction and mapping, *SN Applied Sciences*, **1**(8): 1–24, 2019, doi: 10.1007/s42452-019-0872-y.

2. E.M. Arkin, S.P. Fekete, J.S.B. Mitchell, Approximation algorithms for lawn mowing and milling, *Computational Geometry*, **17**(1–2): 25–50, 2000, doi: 10.1016/S0925-7721(00)00015-8.

3. G.S.C. Avellar, G.A.S. Pereira, L.C.A. Pimenta, P. Iscold, Multi-UAV routing for area coverage and remote sensing with minimum time, *Sensors*, **15**(11): 27783–27803, 2015, doi: 10.3390/s151127783.

4. H. Azpúrua, G.M. Freitas, D.G. Macharet, M.F.M. Campos, Multi-robot coverage path planning using hexagonal segmentation for geophysical surveys, *Robotica*, **36**(8): 1144–1166, 2018, doi: 10.1017/S0263574718000292.

5. R. Bähnemann, N. Lawrance, J.J. Chung, M. Pantic, R. Siegwart, J. Nieto, Revisiting boustrophedon coverage path planning as a generalized traveling salesman problem, [in:] *Field and Service Robotics*, pp. 277–290, Springer, 2021.

6. J. Bedkowski, K. Majek, P. Majek, P. Musialik, M. Pełka, A. Nüchter, Intelligent mobile system for improving spatial design support and security inside buildings, *Mobile Networks and Applications*, **21**(2): 313–326, 2016, doi: 10.1007/s11036-015-0654-8.

7. H. Choset, Coverage for robotics – A survey of recent results, *Annals of Mathematics and Artificial Intelligence*, **31**(1): 113–126, 2001, doi: 10.1023/A:1016639210559.

8. P. Dasgupta, A. Muñoz-Meléndez, K.R. Guruprasad, Multi-robot terrain coverage and task allocation for autonomous detection of landmines, [in:] *Sensors, and Command, Control, Communications, and Intelligence (C3I) Technologies for Homeland Security and Homeland Defense XI*, Vol. 8359, pp. 98–111, SPIE, 2012.

9. K. Easton, J. Burdick, A coverage algorithm for multi-robot boundary inspection, [in:] *Proceedings of the 2005 IEEE International Conference on Robotics and Automation, (ICRA 2005)*, pp. 727–734, IEEE, 2005.

10. M. Fischetti, J.J.S. González, P. Toth, A branch-and-cut algorithm for the symmetric generalized traveling salesman problem, *Operations Research*, **45**(3): 378–394, 1997, doi: 10.1287/opre.45.3.378.

11. E. Galceran, M. Carreras, A survey on coverage path planning for robotics, *Robotics and Autonomous Systems*, **61**(12): 1258–1276, 2013, doi: 10.1016/j.robot.2013.09.004.

12. G. Gutin, D. Karapetyan, A memetic algorithm for the generalized traveling salesman problem, *Natural Computing*, **9**(1): 47–60, 2010.

13. A. Janchiv, D. Batsaikhan, B. Kim, W.G. Lee, S.-G. Lee, Time-efficient and complete coverage path planning based on ow networks for multi-robots, *International Journal of Control, Automation and Systems*, **11**(2): 369–376, 2013, doi: 10.1007/s12555-011-0184-5.

14. N. Karapetyan, K. Benson, C. McKinney, P. Taslakian, I. Rekleitis, Efficient multi-robot coverage of a known environment, [in:] *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 1846–1852, IEEE, 2017.

15. T. Li, D. Ho, C. Li, D. Zhu, C. Wang, M.Q.-H. Meng, HouseExpo: A large-scale 2D indoor layout dataset for learning-based algorithms on mobile robots, [in:] *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 5839–5846, IEEE, 2020.

16. L.D. Nielsen, I. Sung, P. Nielsen, Convex decomposition for a coverage path planning for autonomous vehicles: Interior extension of edges, *Sensors*, **19**(19): 4165, 2019, doi: 10.3390/s19194165.

17. M. Ozkan, A. Yazici, M. Kapanoglu, O. Parlaktuna, A genetic algorithm for task completion time minimization for multi-robot sensor-based coverage, [in:] *Control Applications (CCA) & Intelligent Control (ISIC)*, pp. 1164–1169, IEEE, 2009.

18. I. Rekleitis, A.P. New, E.S. Rankin, H. Choset, Efficient boustrophedon multi-robot coverage: An algorithmic approach, *Annals of Mathematics and Artificial Intelligence*, **52**(2): 109–142, 2008, doi: 10.1007/s10472-009-9120-2.

19. A. Renzaglia, L. Doitsidis, S.A. Chatzichristofis, A. Martinelli, E.B. Kosmatopoulos, Distributed multi-robot coverage using micro aerial vehicles, [in:] *2013 21st Mediterranean Conference on Control & Automation (MED)*, pp. 963–968, IEEE, 2013.

20. A. Renzaglia, L. Doitsidis, A. Martinelli, E.B. Kosmatopoulos, Multi-robot three-dimensional coverage of unknown areas, *The International Journal of Robotics Research*, **31**(6): 738–752, 2012, doi: 10.1177/0278364912439332.

21. S. Saeedi, M. Trentini, M. Seto, H. Li, Multiple-robot simultaneous localization and mapping: A review, *Journal of Field Robotics*, **33**(1): 3–46, 2016, doi: 10.1002/rob.21620.

22. J. Szklarski, Ł. Białek, A. Szałs, Paraconsistent reasoning in cops and robber game with uncertain information: A simulation-based analysis, *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, **27**(03): 429–455, 2019, doi: 10.1142/S021848851950020X.

23. Z. Yan, N. Jouandeau, A.A. Cherif, A survey and analysis of multi-robot coordination, *International Journal of Advanced Robotic Systems*, **10**(12), 2013, doi: 10.5772/57313.

24. L. Zeng, G.M. Bone, Mobile robot collision avoidance in human environments, *International Journal of Advanced Robotic Systems*, **10**(1): 41, 2013, doi: 10.5772/54933.

Received April 6, 2022; revised version October 20, 2022;
accepted October 21, 2022.