This is an accepted version of the article: Kulich, M., Miranda Bront, J. J., and Přeučil, L. (2017). **A meta-heuristic based goal-selection strategy for mobile robot search in an unknown environment**. *Computers \& Operations Research*, 84, which has been published in a final form at https://doi.org/10.1016/j.cor.2016.04.029.

A Meta-heuristic based Goal-Selection Strategy for Mobile Robot Search in an Unknown Environment

Miroslav Kulich[c], Juan José Miranda-Bront[a,b,*], Libor Preucil[c]

[a] *Departamento de Computación, FCEyN, Universidad de Buenos Aires, Pabellón I, Ciudad Universitaria, C1428EGA, CABA, Argentina*
[b] *Consejo Nacional de Investigaciones Científicas y Técnicas, Argentina*
[c] *Czech Institute of Informatics, Robotics, and Cybernetics, Czech Technical University in Prague, Czech Republic*

Abstract

The single-robot search problem in an unknown environment is defined as the problem of finding a stationary object in the environment whose map is not known a-priori. Compared to exploration, the only difference lies in goal selection as the objectives of search and exploration are dissimilar, i.e. a trajectory that is optimal in exploration does not necessarily minimize the expected value of the time to find an object along it. For this reason, in this paper we extend the preliminary ideas presented in Kulich et al. (2014) to a general framework that accounts for the particular characteristics of the search problem. Within this framework, an important decision involved in the determination of the trajectory can be formulated as an instance of the Graph Search Problem (GSP), a generalization of the well-known Traveling Deliveryman Problem (TDP) which has not received much attention in the literature. We developed a tailored Greedy Randomized Adaptive Search Procedure (GRASP) meta-heuristic for the GSP, which generates good quality solutions in very short computing times and is incorporated in the overall framework. The proposed approach produces very good results in a simulation environment, showing that it is feasible from a computational standpoint and the proposed strategy outperforms the standard approaches.

Keywords: Mobile Robotics, Robotic Search, Graph Search Problem, Traveling Deliveryman Problem, GRASP

1. Introduction and literature review

Single-robot search, similarly to exploration, can be understood as a process of autonomous navigation of a mobile robot in an unknown environment in order to find an object of interest. The search algorithm can be formulated as the iterative procedure consisting of model updating with actual sensory data, selection of a new goal for a robot based on the current knowledge of the environment, and subsequent navigation to this goal. A natural condition is to perform the search with an expected minimal usage of resources, e.g., trajectory length, time of search, or energy consumption. Robot search is an important task e.g. in the search and rescue scenario, where the goal is to find a black box flight recorder or debris after a plane crash, or victims/survivors after an accident or a catastrophe. In these situations, typically, the searched object does not move and a precise map of the environment is not available in advance.

While the research of exploration by a single or multiple mobile robots has been quite intensive (see e.g. [5, 32, 35], or our previous research [8, 20]), the search problem has been addressed marginally by the robotic community. On the other hand, the general structure of the search problem is the same as that of the exploration problem. The only difference lies in goal selection as the objectives of search and exploration are dissimilar, i.e. a trajectory that is optimal in exploration does not necessarily minimize the expected value of the time to find an object along it.

Some effort has been devoted to the single and multi-robot search problem for a-priori known environments which can be straightforwardly used as a goal selection strategy in the iterative procedure of the search problem in an unknown space. Sarmiento et al. [31] formulates the problem so that the time required to find an object is a random variable induced by a choice of search path and a uniform probability density function for the object's location. They propose two-stage process to solve the problem. Firstly, a set of locations (known as guards from the Art Gallery Problem [33]) to be visited is determined. An order of visiting those locations minimizing the expected time to find an object is found then. The optimal order is determined by a greedy algorithm in a reduced search space, which computes a utility function for several steps ahead. This approach is then used in Sarmiento et al. [30], where robot control is assumed with the aim to generate smooth and locally optimal trajectories. Hollinger et al. [16] utilize a Bayesian network for estimating the posterior distribution of target's position and present a graph

*Corresponding author
Email addresses: kulich@ciirc.cvut.cz (Miroslav Kulich), jmiranda@dc.uba.ar (Juan José Miranda-Bront), preucil@ciirc.cvut.cz (Libor Preucil)

search to minimize the expected time needed to capture a non-adversarial object.

In the operations research literature, the single-robot search problem in known environments is formulated as the Graph Search Problem (GSP), introduced in Kutsoupias et al. [18] and arising in applications to web searching problems. The GSP can be defined as follows: consider a complete graph $G = (V, E)$, with $V = \{0, 1, \ldots, n\}$, a distance function $d_{ij}, (i, j) \in E$, and a probability p_i that the required information is at vertex $i \in V$. The objective of the GSP is to find a tour that minimizes the expected time to find the required information. Besides some theoretical results regarding approximation schemes presented in Ausiello et al. [4], no further developments are present in the related literature.

The Traveling Deliveryman Problem (TDP) is a well known problem in the operational research community, which has received some attention in the last few years. It can be formulated under the same settings as the GSP, with the only difference that the probability of finding the information is the same for all the vertices in the graph. Integer Linear Programming (ILP) formulations and exact algorithms for the TDP are considered in, e.g., [10, 12, 23]. Recently, several ILP formulations and Branch-and-Cut (BC) and Branch-Cut-and-Price (BCP) algorithms have been developed for the Time-Dependent Traveling Salesman Problem (TDTSP), a generalization of the classical TSP which also generalizes the TDP. Some of these approaches are proposed in [1, 11, 24, 25]. Overall, the best exact algorithm is the BCP proposed in Abeledo et al. [1], being able to solve instances with up to 107 vertices to optimality in several hours of computing time.

In addition, several heuristics and meta-heuristics have been proposed for the TDP and some other variants. The approaches rely mostly on Greedy Randomized Adaptive Search Procedure (GRASP), introduced originally by Feo and Resende [9], and Variable Neighborhood Search (VNS), proposed by Hansen and Mladenovic [13]. Salehipour et al. [29] propose a GRASP, evaluating the impact of considering for the local search phase a Variable Neighborhood Descent (VND) as well as a VNS procedure. Mladenovic et al. [26] propose a General VNS (GVNS), which is able to improve the results obtained by Salehipour et al. [29]. Silva et al. [34] propose a simple multistart heuristic combined with an Iterated Local Search procedure. The method improves all the results reported in Salehipour et al. [29] and finds a new best known solution in two benchmark instances. To the best of our knowledge, the approach by Silva et al. [34] is the one producing the best results in the literature. Finally, regarding variants of the TDP, Dewilde et al. [6] tackle the TDP with profits and Heilporn et al. [14] the TDP with time windows.

The single-robot search problem in an unknown environment is formally formulated in our previous research Kulich et al. [21], where a criterion to be optimized is defined and several goal-selection strategies are considered. Two of these strategies are borrowed from the exploration problem, for which they were originally designed, namely the *greedy strategy* proposed in Yamauchi [35] and the *Traveling Salesman strategy* introduced in Kulich et al. [20], while the third one is based on the formulation and resolution of a TDP instance within the framework. The three strategies are evaluated in a simulation environment and the behavior is discussed. The results are somehow mixed, showing that none of these strategies clearly dominates the others. As the key problem was identified that the studied strategies do not use information gain of visiting a goal, which may be crucial for designing effective search strategies.

The aim of this research goes in that direction. The contribution of this paper is threefold. Firstly, we build upon the preliminary results presented in Kulich [21] and provide a complete framework that accounts for the particular characteristics of the problem. Secondly, in order to provide a complete implementation of such framework, we develop a meta-heuristic to solve the GSP, tailored for the context of the search problem and employ it as a goal-selection strategy within the search framework. In the proposed framework, instances consider generally between 50 and 100 vertices, but the amount of time required to obtain a near-optimal solution should not exceed one second. This is a key factor since, opposed to the TSP, even the TDP requires several hours of execution to solve to optimality instances of moderate size. Finally, the whole framework is evaluated computationally in a simulation environment, where the proposed goal selection strategy outperforms the traditional one and showing that the proposed strategy has potential to be applied in practice.

The rest of the paper is organized as follows. The problem definition is presented in Section 2, while the frontier-based framework for search is described in Section 3. The GRASP approach developed for the GSP is described in Section 4. In Section 5 we present the computational results, including the evaluation of the GRASP meta-heuristic and the evaluation of the overall framework in a simulation environment. Finally, Section 6 is dedicated to concluding remarks and future directions.

2. Problem definition

Assume an autonomous mobile robot equipped with a ranging sensor with a fixed, limited range (e.g. laser rangefinder) and 360° field of view operating in a priory unknown environment. The search problem is defined as the process of navigation of the robot through this environment with the aim to find a stationary object of interest placed in the environment randomly and reachable by the robot[1]. By finding an object we understand the situation

[1] In general, a-priory information about object's position can be given in the form of a probability density function (PDF), but a uniform distribution is expected in the paper. On the other hand, incorporation of a-priory PDF is straightforward as it involves only determination of probabilities $p(t)$ in (2).

when it is firstly detected by robot's sensors. A natural condition is to minimize the time when this situation occurs. More formally, this condition can be expressed as minimization of the expected (mean) time T_f the object is firstly detected, when the robot follows the trajectory R:

$$T_f = \mathbb{E}(T|R) = \int_0^\infty tp(t)\ \mathrm{d}t, \tag{1}$$

where $p(t)$ is the probability of finding the object at time t.

Sensing as well as planning is performed in discrete times, therefore (1) can be rewritten as

$$T_f = \mathbb{E}(T|R) = \sum_{t=0}^{\infty} tp(t), \tag{2}$$

where $p(t) = \frac{A_t^R}{A_{total}}$ is the ratio of the area A_t^R newly sensed at time t when the robot follows the trajectory R and A_{total}, the area of the whole environment the robot operates. The objective is to find the trajectory R^{opt} minimizing (2):

$$R^{opt} = \arg\min_R \mathbb{E}(T|R) = \arg\min_R \sum_{t=0}^{\infty} tA_t^R. \tag{3}$$

Notice that A_{total} can be omitted as it is a constant.

3. Framework

3.1. General settings

The proposed framework for single-robot search is derived from seminal Yamauchi's frontier based approach (FBA) [35] successfully used for exploration. FBA utilizes an occupancy grid as the environment representation, which divides robot's working space into rectangular grid cells, where each cell stores information about the corresponding piece of the space in the form of a probabilistic estimate of its state. The grid is built incrementally as actual sensor measurements are gathered during search and serves as a model of the environment (map) for further exploration steps.

The search process, which is summarized in Algorithm 1, is an iterative procedure that is terminated once the search object is detected (it is assumed that the object lies entirely in one grid cell and it is detected when the cell is within a visibility range of the sensor) (line 1). After the map is updated with recent sensory information, occupancy grid cells are consequently segmented into three categories (by application of two thresholds on their probability values): free, occupied, and unknown (unsearched) (line 2). After that, *frontier cells*, i.e. grid cells representing free regions adjacent with at least one not yet searched cell, are detected (line 3). Each continuous set of frontier cells forms *frontier* (each frontier cell is thus a member of exactly one frontier), see Fig. 1 for visualization of the used terms. The most appropriate frontier cell is then selected as the new robot goal, which is done in two steps.

Algorithm 1: Frontier based search.

```
1 repeat
2 |  Get the updated map built from sensor
  |   readings;
3 |  Detect all frontiers from the actual map;
4 |  Determine goal candidates;
5 |  Select the next goal from the set of goal
  |   candidates;
6 |  Plan a path to the next goal;
7 |  Perform the plan;
8 until the object is found;
```

At first, frontier cells are filtered to determine a set of representatives approximating the frontier cells such that each frontier cell is detectable by the robot sensor from at least one representative (line 4). This is done by k-means clustering for each frontier, where the representative of each cluster is the closest frontier cell to the cluster's mean. The number of clusters a frontier is split into is defined similarly to [8] as

$$n_f = 1 + \left\lfloor \frac{N_f}{1.8\rho} + 0.5 \right\rfloor,$$

where N_f is the number of cells forming the frontier and ρ is a sensor range. This guarantees that all frontier cells will be explored after visiting all representatives (goal candidates) and thus the searched object will not be missed. In the second step, the most suitable goal candidate according to the defined criteria is determined and assigned as a new goal (line 5). Finally, the shortest path from the robot's current position to the goal is found (line 6) and the robot is navigated along that path (line 7).

The presented algorithm is in robotic applications typically implemented in two threads: sensor processing, map building, and robot control, which run frequently (with frequency in units of tens of Hz) in one thread, while goal selection incorporating steps 3–6 in the second one. Goal selection does not need to be run so frequently, conversely, higher frequencies lead to oscillations of robot movement causing longer trajectories and search time. Amigoni et al. [2] for example suggest to perform goal selection with frequency of 0.6 Hz. In other words, the goal selection process should take less than two seconds.

3.2. Search problem and the goal selection strategy

In this section we formulate the search strategy and provide some implementation details that incorporate key information regarding the search problem.

Recall that the overall objective is to find a trajectory that minimizes the expected time to detect the object by the robot's sensors, as stated in (2) and (3). We construct a complete graph $G = (V, E)$, with $V = \{0, 1, \ldots, n\}$ the set of goal candidates, with 0 a special vertex representing the position of the robot (depot), and E the set of

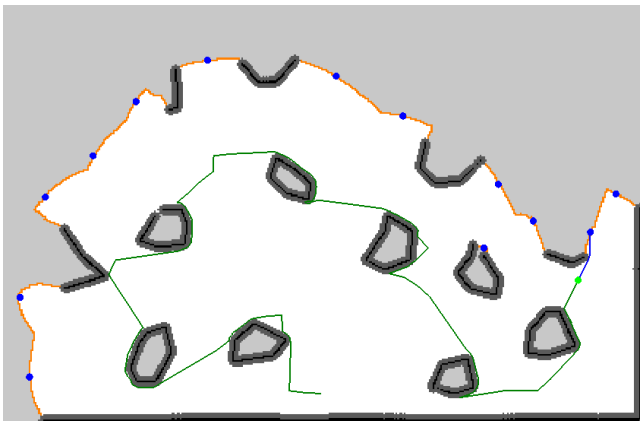

Figure 1: Illustration of the used terms at one particular step of search: empty space is in white, occupied in black, and unknown in light gray (Notice the dark gray areas around obstacles which represent obstacles "blown" with Minkowski sum. This enables to plan a path for a robot with a circular shape: path planning for a point-like robot in the space with blown obstacles equals to planning for a robot with a circular shape). Frontiers are represented by the orange strings, while the blue dots stand for goal candidates, and the blue one for the current robot position. Finally, the green curve represents already traversed path and the blue one a plan to the chosen candidate.

edges (i, j), $i, j \in V$, $i \neq j$, each of them having an associated distance d_{ij} representing the shortest path from i to j on the grid. We assume that the time needed to reach a candidate is proportional to the distance. We further consider for each vertex $i \in V$ a non-negative weight w_i, representing an approximation of the probability of finding the object when visiting candidate i.

Let $\pi = (\pi_0 = 0, \pi_1, \ldots, \pi_n)$ be a Hamiltonian path, starting from the depot. We define

$$t_k(\pi) = \sum_{i=1}^{k} d_{\pi_{i-1}\pi_i} \tag{4}$$

as the time to reach vertex k given that the vertices are visited following path π. With this information, at each iteration of the overall scheme we approximate the problem defined in (3) by selecting as the next goal the first candidate in the optimal path π^* minimizing the weighted sum of the times t_k

$$\begin{aligned} \pi^* &= \arg\min_{\pi} \sum_{k=1}^{n} w_k t_k(\pi) \\ &= \arg\min_{\pi} \sum_{k=1}^{n} w_k \sum_{i=1}^{k} d_{\pi_{i-1}\pi_i}. \end{aligned} \tag{5}$$

As described in the introduction, this problem is the GSP and has received little attention in the literature. It generalizes the TDP, which can be considered by setting $w_i = 1$ for $i \in V$. Given the context under consideration for the search problem, as mentioned previously we need to resort to a heuristic algorithm, which should provide near-optimal solution in very short computing times for instances containing about 100 vertices.

Compared to the exploration problem as well as with the previous version considered in Kulich et al. [21], the formulation of (5) considers explicitely the probabilities of finding the object when visiting a candidate goal. From a practical standpoint, our proposal is to approximate this values using the information known by the robot. Assuming the height and width of the whole rectangular environment to be known, we compute the Voronoi diagram over the unknown area using the goal candidates as seeds. For each candidate $i \in V \backslash \{0\}$, let m_i be the number of cell grids in its neighborhood and let $m_t = \sum_{i \in V \backslash \{0\}} m_i$ the total number cells in the unknown area. Then, we define $w_i = \frac{m_i}{m_t}$ as the approximation for the probability of finding the object when visiting goal candidate i.

4. Solution approach for the GSP

In this section we provide the details regarding the metaheuristic approach considered for solving the GSP in the context of the search problem. We first describe two greedy heuristics and the local search operators developed, and then explain how they are combined within a GRASP scheme.

4.1. Constructive heuristics

We consider two greedy heuristics, which differ in the criterion used to expand the current (partial) solution since one of them incorporates the weights w_i in the decision. These two simple heuristics are then extended to their corresponding randomized version to be incorporated into the GRASP, and used as a baseline for comparison as well. We show in Algorithm 2 the scheme considered, with a general cost function $f : V \times V \to \mathbb{R}$.

Algorithm 2: General greedy scheme.

1 Set $P = (0)$, i.e., the path having only the depot.
2 **while** *there are unassigned vertices* **do**
3 Let $P = (0, v_1, \ldots, v_k)$.
4 Set $v = \arg\min_{u \in V, u \notin P} f(v_k, u)$ and append v to the end of P.
5 Return path P.

The difference between the two heuristics lies in the definition of f, which affects the selection of the next vertex to be visited (line 4). We consider the standard distance-based function, $f_{\text{dist}}(u, v) = d_{uv}$, and we name the resulting heuristic as G_{dist}. In order to incorporate the weights for each candidate, we further consider the function $f_{\text{ratio}}(u, v) = d_{uv}/w_v$. The resulting heuristic is referred as G_{ratio}.

4.2. Improvement phase

After obtaining an initial feasible solution, we perform an improvement phase by means of a Variable Neighborhood Descent (VND) composed of different neighborhoods. We consider three different local search operators, two of them are the following two classical ones:

- **Swap**: Select two vertices in the tour and interchange them.
- **2-opt**: Select two non-adjacent arcs and replace them by two new arcs, obtaining as a result a new path.

We consider also an adaptation of the well-known Lin-Kernighan (LK) operator designed for the TSP in Lin and Kernighan [22]. The main idea behind this heuristic is to improve a feasible solution by applying a sequence of 2-opt moves, and is one of the most effective heuristics for the classical TSP (see, e.g., Applegate et al. [3, Chapter 15]). The idea is to consider a more time-consuming operator, aiming to find very good quality solutions in a reasonable time by exploring a larger neighborhood. Karapetyan and Gutin [17] provide a nice explanation of the method, excluding some of the technical aspects which obscure the basic ideas, as well as an adaptation of the procedure for the Generalized Traveling Salesman Problem (GTSP), obtaining very good computational results. We name this LK-based operator for the GSP as **LK-op**, which is described in the following.

We point out that the presence of a depot and the order of the vertices have a direct impact on the value of the objective function. For instance, given an ordering of the vertices, in the TSP the value of the solution following such an order or its reverse is exactly the same. On the contrary, this is not the case in the GSP given the cumulative structure of the objective function. In this sense, the overall heuristic differs from the traditional versions for the TSP and similar problems. In addition, before starting the optimization, for each vertex $v \in V$ we sort its neighbors $w \in V,\ w \neq v$ according to the distances d_{vw} ascendingly. We name $neighb(v)$ to the list of neighbors for vertex $v \in V$.

The overall approach of LK-op starts from a feasible path $P = (0 = v_0, v_1, \ldots, v_n)$. Each edge (v_i, v_{i+1}), $i = 0, \ldots, n-1$, is considered sequentially as a seed for an improvement procedure, $improvePath_{\text{GSP}}$, which will attempt to obtain an improved path by applying a sequence of 2-opt moves, not necessarily improving the current solution, with a limited backtracking. If such a better path is found, we accept it as the new (initial) feasible solution and re-start the procedure again from the first edge. Otherwise, we move to the next edge, until all edges have been considered as a seed.

The sketch of the $improvePath_{\text{GSP}}$ is shown in Algorithm 3, where we follow mostly the notation in Karapetyan and Gutin [17]. To reduce the computation time, a length of performed sequences of 2-opt moves is limited to a maximum depth of α moves[2] (line 2). In addition, for each vertex considered only a subset of its closest vertices are expanded, and the size of this subset depends on the current depth within the backtracking (line 3). Following the notation in Applegate et al. [3], we denote this parameter as $breadth[depth]$ and consider it as an input parameter. Finally, the procedure follows a first-improvement strategy (lines 7-8). As reported by Helsgaun [15] and noted by Karapetyan and Gutin [17], for the TSP this strategy simplifies the routine without affecting the quality of the solutions. In our case, we also adopted this strategy aiming to reduce the computation times, given the context described for the search problem.

Algorithm 3: $improvePath_{\text{GSP}}(P, depth, R, v_i)$.

```
1  Let P = (0 = v_0, v_1, ..., v_n) be a feasible solution,
     depth the recursion depth and R a set of restricted
     vertices, v_i the vertex used as seed.
2  if depth < α then
3      for each of the first breadth[depth] vertices in
         neighb(v_i) do
4          Let v_k be the current neighbor of v_i, and
             (v_k, v_{k+1}) the edge to be removed.
5          If v_k ∈ R or the edge (v_k, v_{k+1}) is next to
             (v_i, v_{i+1}), then skip v_k and move to the
             next vertex in the list.
6          Otherwise, let P' be the path obtained by
             applying the 2-opt move induced by
             (v_i, v_{i+1}) and (v_k, v_{k+1}), and g the
             corresponding gain.
7          if g > 0 then
8              Accept the path P' as the new solution
                 and terminate.
9              return P'
10         else
11             Call the recursive step aiming to improve
                 P' using v_k as a seed.
12             return improvePath_GSP(P', depth +
                 1, R ∪ {v_k}, v_k)
13 return P.
```

Compared to the TSP, the cumulative nature of the objective function requires some further developments in order to explore certain neighborhoods efficiently. Mladenovic et al. [26] and Silva et al. [34] consider two different strategies in this direction, in both cases they add extra data structures in order to evaluate moves in (amortized) constant $O(1)$ operations. In our implementation, we extend the ideas proposed in Mladenovic et al. [26] by including an extra sequence to store the accumulated weights of a given path, in addition to the structures considered to

[2] A special case of the 2-opt move is considered when one of the vertices involved is the last vertex in the path.

store information regarding the (partial) cumulative objective function at each vertex. In this fashion, we can explore both the Swap and the 2-opt neighborhoods in $O(n^2)$ operations. It is important to remark also that each move applied by the LK-op operator (line 6) requires $O(n)$ operations, since either the objective function has to be recomputed or the structures need to be updated. This is an important difference compared the original procedure for the TSP and some adaptations to other similar problems.

Finally, we describe how we combine these local search operators into two VND procedures considered:

- VND_1: apply 2-opt and Swap operators until no further improvements are found.
- VND_2: execute the VND_1 procedure, and then LK-op to the resulting solution.

4.3. GRASP scheme

The algorithms described previously are combined in this section and incorporated within a GRASP scheme meta-heuristic. As mentioned in Section 4.1, both constructive algorithms G_{dist} and G_{ratio} are slightly modified including a straightforward randomization step. At each iteration, a *Restricted Candidate List* (RCL) with the best *rclsize* candidates according to the corresponding function are considered, and the next vertex to be added to the path is randomly selected among them. These randomized versions are noted as $G^{\text{rand}}_{\text{dist}}$ and $G^{\text{rand}}_{\text{ratio}}$, respectively.

The general scheme of the GRASP is shown in Algorithm 4. We firstly remark that for the constructive step we consider both greedy heuristics (lines 3 and 8, respectively), running *ngit* iterations of each of them. For each solution obtained in the constructive step, a local search phase is applied (lines 4 and 9). Finally, the best solution found during the process is returned (line 12).

Algorithm 4: GRASP scheme.

```
1  Let P_best be the best solution found, z_best = ∞.
2  for k = 1, ..., ngit do
3  |  Obtain a feasible path P using G^rand_dist.
4  |  Improve path P by applying a local search step,
   |    and z its cost.
5  |  if z < z_best then
6  |  |  Set P_best = P, z_best = z.
7  for k = 1, ..., ngit do
8  |  Obtain a feasible path P using G^rand_ratio.
9  |  Improve path P by applying a local search step,
   |    and z its cost.
10 |  if z < z_best then
11 |  |  Set P_best = P, z_best = z.
12 return P_best
```

For the local search phase we evaluate different alternatives, i.e., VND_1, VND_2, and a mix them to trade-off between quality and computing times. The details and the results are presented in the next section.

5. Computational results

In this section we present the computational results obtained for the search problem. Firstly, we conducted an experimental study in order to evaluate the quality and the computation effort required to solve the GSP on literature benchmark instances. Then, the selected meta-heuristic has been embed within the framework described in Section 3 and evaluated in a simulation environment, comparing the results obtained with the usual strategy adopted in the exploration problem.

5.1. Solution of the GSP

The methods described in Section 4 are coded in C++, using `g++ 4.8.2` and an Ubuntu Linux 14.04 LTS as operating system. The experiments are run on a Lenovo Thinkpad T530 with an Intel Core i5-3320M CPU and 4Gb of RAM. Although some of the methods can be easily parallelized, our program runs in a single thread in order to account for contexts with limited resources.

Regarding the parameters for the different components of the algorithms, based on limited preliminary experiments for the operator LK-op we set $\alpha = 20$, limiting the size of the sequences of 2-opt moves. In the same direction, the backtracking is limited by setting $breadth[depth] = 5$, for $k = 1, \ldots, 4$, and $breadth[depth] = 1$ for $k > 4$. Regarding the GRASP scheme, we set $ngit = 50$, generating a total of 100 initial solutions, and $rclsize = 3$. Under these settings, we consider the following variants::

- ***H*-dist**: Greedy heuristic G_{dist} followed by VND_2 as a local search step.
- ***H*-ratio**: Greedy heuristic G_{ratio} followed by VND_2 as a local search step.
- **GRASP-S**: GRASP scheme, using VND_1 as the local search step.
- **GRASP-F**: GRASP scheme, using VND_2 as the local search step.
- **GRASP-Int**: GRASP-S, and apply the LK-op to solutions that are promising, i.e., those whose cost is within the 10% of the best solution found so far. The intuition is to apply an intensification step to good quality solutions, aiming to obtain the same results as in GRASP-F in shorter computing times.

With respect to the experiments, to the best of our knowledge there are no benchmark instances for the GSP in the related literature. Therefore, we decided to evaluate the methods using 21 instances from TSPLIB [28] with size between 42 and 107 vertices. Firstly, the instances are interpreted as TDP ones, for which optimal solutions are

known for almost all of them. Recall that TDP stands for the GSP setting $w_i = 1$, $i \in V$. Furthermore, on this instances we are able to compare our results with the best method so far in the literature, proposed by Silva et al. [34]. In addition, for each of these instances we generate 10 GSP instances by taking $w_i \in Unif[1, 100]$, $i \in V$. As a result, 210 GSP instances representative of our context of application are considered.[3]

Table 1 shows the results for the TDP instances. In order to be able to compare our results with the previous ones in the literature, the objective function is slightly modified in order to account for the *cycle* version of the problem. For each instance, we report the value of the Best Known Solution (BKS), obtained in most cases from the results reported in Abeledo et al. [1] and in a few cases by the improved solutions reported by Silva et al. [34]. Regarding the methods, we report the average gap (%G) obtained by the heuristic H-dist. For the three variants of the GRASP scheme, namely GRASP-S, GRASP-F, and GRASP-Int, we performed 10 independent runs with different seeds. Therefore, we report the percentage gap (%bG) of the best solution found, the percentage average gap (%aG) and the average computation times (Time) in seconds. In all cases, gaps are computed as $100 \times (Sol - BKS)/BKS$.

The main message of Table 1 is, as expected, that the best results are provided by GRASP-F. The best solution found is in all cases below 1% of the optimum solution, and on average is below 1.5% in all cases except for instance eil101. However, the computing time required for the instances with approximately 100 vertices are larger than the limits imposed by the context. In this sense, the best results are produced by GRASP-Int, which in terms of quality are very close (both the best solution and on average) to the ones obtained by GRASP-F, but the computing times exceed one second only in two of the 21 instances. Both H-dist and GRASP-S show to be feasible from a computational side and provide reasonable results on most of the instances, although they are in general far from the 1-2% of gap obtained consistently by GRASP-F and GRASP-Int. We remark the special result on instance pr107, where H-dist is able to find the best solution of all heuristics.

Finally, although not included explicitly in the table, we make a comparison with the best heuristic in the literature, the one proposed by Silva et al. [34]. Regarding the quality, they are able to obtain the best know solutions in all runs. However, although some details regarding the processor are not detailed, it seems that the approach may exceed the time restrictions for the execution desirable in our context.[4]

[3]Instances can be downloaded from `goo.gl/2aiBno`. Last access: September 30, 2015.

[4]Assuming their processor is an Intel i7-870, 2.93 Ghz, a comparison at `cpubenchmark.net` retrieves that it has a better overall ranking compared to our processor, and comparable when running on single thread.

We next show in Table 2 the results for the GSP instances. In this case, each row stands for the average results over the 10 GSP instances generated based on the original one. Since we do not have the optimal solutions for these instances, or at least good lower bounds, we follow the same strategy as in Silva et al. [34] and compute the improvement percentages with respect to the best solution obtained by G_{dist} and G_{ratio}. We report the average improvement percentage (%aImp) for H-dist and H-ratio. For the GRASP-based heuristics, again we report the average of improvements percentages over 10 independent runs, averaged over the 10 different instances considered. Therefore, we report the average best improvement percentage (%bImp), the average improvement percentage (%aImp) and the computing times required (Time) in seconds. Percentage improvements are computed as $100 \times (Sol - UB)/UB$.

The results for the GSP instances follow the same tendency as for the TDP ones. GRASP-F produces the best results, but the best trade-off between quality and computing times is obtained by GRASP-Int. However, we can observe in general an increase in the computing times, meaning that the routines spend more time improving the solutions compared to the TDP case. For this instances we also present the results for the two heuristics: H-dist and H-ratio. Recall that the only difference between these two heuristics lies in the constructive step. H-ratio outperforms H-dist, obtaining larger improvements in 18 out of 21 instances and being comparable in the remaining three.

Overall, and based on the results presented in this section, GRASP-Int appears as the best choice to be included in the framework since it is feasible from a computational standpoint and provides good quality solutions.

5.2. Simulation results

Performance of the proposed GSP solver (the GRASP-Int variant) as a goal selection strategy in the search framework was evaluated and compared to the standard greedy approach widely used for exploration. To study a theoretical behavior of the method not influenced by sensor noise, localization impression, or motion control, the first pillar (Level-0) described in Faigl [7] was followed: both strategies were deployed in the search framework and run a simulator we developed[5].

The comparison was done in four environments from Motion Planning Maps Dataset [27], which were scaled to dimensions of 20 $m \times 20$ m: *empty*, *potholes*, *jari-huge*, *warehouse*, see Fig. 2. The *empty* map simulates a trivial case of a big area without obstacles, while the *potholes* map represents an unstructured environment with many small obstacles. *jari-huge* is a map of a real administrative building with separated rooms and corridors between

[5]The software library containing the simulator as well as the search framework is intended to be available at the time of publication of the paper.

Instance	n	BKS	H-dist	GRASP-S			GRASP-F			GRASP-Int		
			%G	%bG	%aG	Time	%bG	%aG	Time	%bG	%aG	Time
dantzig42	42	12528	1.80	0.00	0.50	0.03	0.00	0.01	0.22	0.00	0.01	0.11
swiss42	42	22327	4.07	0.06	1.93	0.03	0.00	0.00	0.21	0.00	0.02	0.06
att48	48	209320	0.20	0.33	1.01	0.05	0.00	0.20	0.35	0.00	0.20	0.18
gr48	48	102378	2.89	1.53	3.38	0.04	0.00	0.77	0.35	0.00	0.85	0.16
hk48	48	247926	2.69	0.94	2.44	0.04	0.00	0.12	0.44	0.00	0.13	0.16
eil51	51	10178	1.71	1.23	3.23	0.05	0.00	0.21	0.53	0.00	0.25	0.28
berlin52	52	143721	2.72	1.65	3.33	0.06	0.25	0.85	0.39	0.25	0.85	0.17
brazil58	58	512361	20.96	0.39	1.89	0.08	0.00	0.34	0.55	0.00	0.47	0.23
st70	70	20557	2.50	2.68	5.02	0.14	0.42	1.20	0.84	0.42	1.40	0.31
eil76	76	17976	2.38	3.33	6.33	0.16	0.18	1.30	1.76	0.18	1.40	0.49
pr76	76	3455242	4.05	1.25	4.12	0.19	0.19	1.03	1.35	0.19	1.15	0.48
gr96	96	2097170	5.41	1.12	4.16	0.40	0.42	0.99	1.94	0.42	1.02	0.80
rat99	99	57896	2.75	4.14	5.11	0.36	0.99	1.38	2.83	0.99	1.43	1.30
kroA100	100	983128	6.89	4.73	6.51	0.44	0.46	0.96	2.76	0.46	1.22	0.81
kroB100	100	986008	3.09	2.72	4.20	0.43	0.46	0.93	2.16	0.46	1.02	0.89
kroC100	100	961324	4.55	3.66	4.97	0.44	0.55	0.93	2.40	0.69	1.07	0.79
kroD100	100	976965	0.82	3.94	6.02	0.42	0.41	1.45	2.45	0.41	1.69	0.89
rd100	100	340047	8.39	3.35	6.51	0.43	0.59	0.96	2.75	0.59	1.04	0.77
eil101	101	27513	5.34	5.27	7.08	0.35	0.83	2.12	4.55	1.85	2.56	1.64
lin105	105	603910	8.32	3.86	5.46	0.55	0.54	0.91	2.19	0.57	1.08	0.74
pr107	107	2026626	0.16	1.77	3.07	0.49	0.27	0.72	1.72	0.27	0.73	0.88

Table 1: Computational results for TDP instances.

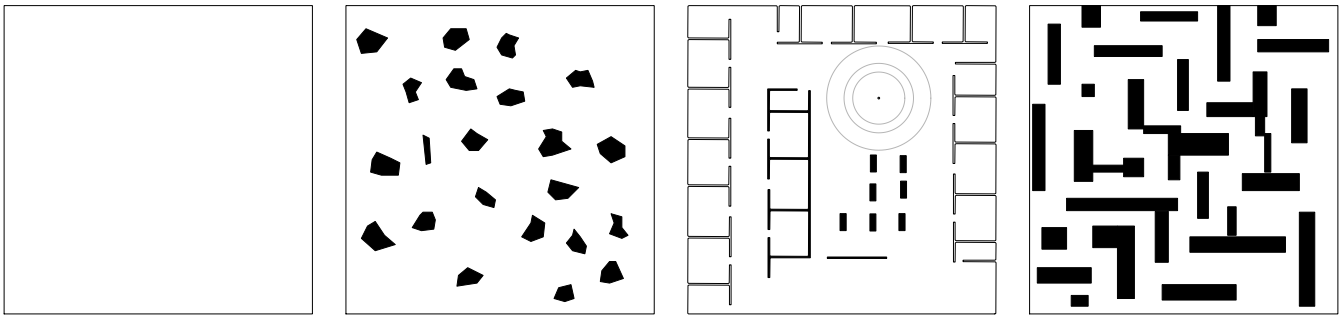

Figure 2: The maps used for evaluation (from left to right): *empty*, *potholes*, *jari-huge*, *warehouse*. The circle in the *potholes* map represent visibility ranges of 1.5, 2.0, and 3.0 meters.

them. Finally, *warehouse* is a space with a number of rectangular obstacles. For each map, three different sensor ranges were considered: 1.5 m, 2 m, and 3 m, and 50 trials were run for each combination $\langle map, range, method \rangle$, which gives 1200 trials in total.

The obtained results are statistically summarized in Table 3. *avg* stands for the expected time of finding the object T_f as defined in (2) averaged over all 50 runs, *min* and *max* are minimal and maximal values of T_f over these runs and *stdev* stands for their standard deviation. *ratio* expresses a ratio of the averages of the methods determined as $avg_{GRASP-based}/avg_{greedy}$.

Generally, the proposed method outperforms the greedy approach in all cases. The biggest difference (up to 16%) is for the *empty* and *potholes* maps, which force the goal selection strategy to make hard decisions (i.e. to choose between a number of goal candidates which are at similar distances from a robot) frequently. The progress of both methods at various time steps is illustrated in Fig. 3. It can be seen that greedy frequently moves around a nearest obstacle (steps 169, 284, and 689), which is inefficient and critical especially in the first stages of search, because amount of newly search space is minimal. GRASP, in opposite, guides the proposed approach to large unsearched areas leaving small islands of space unsearched. This is the desired behavior according to the objective of search. On the other hand, the time needed for the whole environment is higher for the GRASP-based method. Evolution of relative amount of searched area in time is demonstrated in Fig. 4, which support the previous findings. The proposed approach searches the environment fast in the first stages, while visiting all small missed places takes time at the end of the process.

On the other hand, *warehouse* is relatively easy for both approaches, as a structure of a map built during search looks like a corridor in majority of cases and the strategies thus recommend to continue in the current corridor in these cases. Moreover, the penalty of choosing non-

Instance	n	H-dist	H-ratio	GRASP-S			GRASP-F			GRASP-Int		
		%aImp	%aImp	%bImp	%aImp	Time	%bImp	%aImp	Time	%bImp	%aImp	Time
dantzig42	42	-9.77	-13.48	-14.62	-13.59	0.03	-15.28	-15.09	0.33	-15.27	-15.06	0.13
swiss42	42	-10.62	-11.32	-14.03	-12.86	0.03	-14.72	-14.63	0.30	-14.72	-14.59	0.12
att48	48	-9.17	-11.19	-12.90	-11.86	0.05	-13.67	-13.48	0.54	-13.67	-13.44	0.23
gr48	48	-17.41	-18.21	-20.15	-18.72	0.05	-20.86	-20.67	0.47	-20.86	-20.60	0.17
hk48	48	-11.97	-16.45	-18.21	-16.95	0.05	-19.29	-18.96	0.64	-19.24	-18.92	0.25
eil51	51	-7.82	-10.99	-11.82	-10.48	0.06	-13.23	-12.80	0.88	-13.23	-12.75	0.38
berlin52	52	-9.21	-8.49	-12.41	-11.14	0.06	-13.88	-13.39	0.55	-13.84	-13.32	0.19
brazil58	58	-2.04	-15.19	-20.15	-18.63	0.09	-21.35	-20.71	0.74	-21.34	-20.63	0.25
st70	70	-8.30	-10.73	-12.75	-11.06	0.16	-14.66	-14.10	1.34	-14.58	-13.99	0.46
eil76	76	-10.97	-11.69	-12.87	-10.46	0.19	-15.56	-14.86	2.75	-15.51	-14.65	0.69
pr76	76	-9.51	-9.79	-12.18	-10.07	0.20	-13.83	-13.35	1.94	-13.82	-13.24	0.49
gr96	96	-13.18	-13.11	-15.33	-14.03	0.41	-18.18	-17.51	2.87	-18.09	-17.35	0.94
rat99	99	-11.41	-13.79	-14.24	-12.52	0.42	-17.43	-16.88	4.60	-17.43	-16.72	1.47
kroA100	100	-13.32	-16.62	-17.15	-15.32	0.47	-20.02	-19.54	3.90	-19.96	-19.31	0.93
kroB100	100	-12.67	-15.21	-15.39	-13.83	0.47	-17.72	-17.17	3.48	-17.64	-17.11	1.18
kroC100	100	-9.17	-9.49	-10.30	-9.08	0.46	-13.11	-12.64	3.47	-13.09	-12.57	1.24
kroD100	100	-10.57	-13.58	-14.36	-12.91	0.47	-16.96	-16.34	3.34	-16.96	-16.25	1.17
rd100	100	-11.43	-13.31	-13.92	-12.28	0.46	-16.45	-15.84	3.78	-16.37	-15.66	0.99
eil101	101	-13.95	-15.64	-16.60	-14.60	0.44	-19.97	-19.27	6.13	-19.92	-19.06	1.51
lin105	105	-10.07	-14.55	-16.22	-14.39	0.55	-18.80	-18.12	3.44	-18.63	-17.90	0.85
pr107	107	-4.62	-4.81	-6.34	-5.19	0.55	-7.61	-7.09	2.25	-7.61	-7.08	1.23

Table 2: Computational results for GSP instances.

optimal candidate is not so big. Fig. 5 demonstrates the progress of greedy and shows that this strategy leads a robot to new areas in first stages of search. Moreover, a number of crossings indicating visiting of already searched places is not big.

The strategies can behave differently at "crossroads" (i.e. places where selection of different candidates nearest to a robot leads to topologically different paths as shown in the second leftest picture of Fig. 5), where the GRASP-based strategy prefers candidates beyond whose large unsearched areas are expected. This gives the strategy a slight advantage leading to a little better results in search. Notice that the worst solutions found by GRASP in *empty* and *potholes* are even better than the best ones found by greedy and the worst solutions found by GRASP in *jari-huge* are better than the average solutions found by greedy.

Another property worth to mention is variance (standard deviation more precisely) of the results, which is significantly lower (3 times for some cases) for the GRASP-based strategy making its solutions more consistent. The explanation is straightforward. Assume several candidates with almost equal distances to a robot, which is a frequent case: when the robot searches a new area, new candidates appear on a newly discovered circularly-shaped frontier at the distance of a sensor range from the robot. Even small perturbations in positions of these candidates cause that the greedy strategy selects different candidates. This is not a case of GRASP as it assumes also farther candidates, which make a shape of a found solution more stable to position perturbations.

6. Conclusions and future research

This article presents a general frontier-based framework for the single-robot search problem in an unknown environment, where the goal selection strategy involves the solution of a GSP instance. We provide a complete implementation of such a framework, including a tailored GRASP meta-heuristic for the GSP which is able to find near-optimal solutions in about one second of computing times. The overall framework is evaluated in a simulation environment on four different maps and 1200 trials in total. The results show that the proposed strategy outperforms the strategies previously considered in Kulich et al. [21] and that it is feasible to be applied in practice.

As future research, several lines remain open that would be interesting to consider. From the mobile robotics side, the next step would be to evaluate the framework and the GSP strategy according to the second and third pillar of the methodology [7], especially to perform experiments on real robots. Regarding the framework, a natural extension could be to consider the multi-robot scenario, and to evaluate the impact both from the computational feasibility and the quality of the results in simulations. Another interesting stream is to consider a moving object to be found which leads to spatio-temporal search. Preliminary results of utilization of a simple depth-first search were presented in Krajnik et al. [19]. A natural continuation is to employ generalization of the proposed meta-heuristics for the spatio-temporal case.

Regarding the GSP, it appears as a very difficult and challenging combinatorial optimization problem. It would be interesting to explore further meta-heuristics schemes both for the GSP in general and to be included in the framework as well. In addition, developing exact algo-

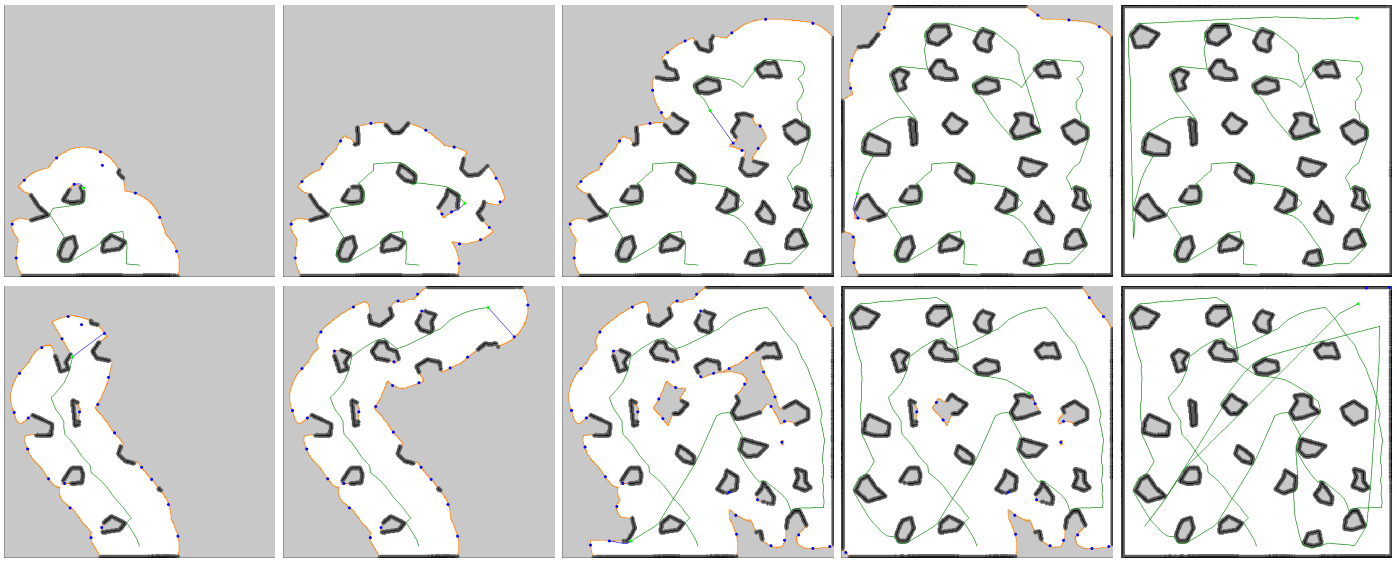

Figure 3: Search progress for the *potholes* map and a sensor range=3 m. The top row depicts a currently searched area by a random run of the greedy method at times 169, 284, 689, 1074, and 1434. The second row shows progress of the GRASP-based approach at the same times (except the last one which is 1879. The colors have the same meaning as in Fig. 1.

Map	Range	Greedy				GRASP-based				Ratio
		avg	min	max	stdev	avg	min	max	stdev	%
empty	1.5	913.12	855.78	995.34	25.74	817.75	787.68	857.92	16.78	89.55
	2.0	693.01	630.50	770.99	34.17	616.20	586.55	648.70	11.41	88.92
	3.0	452.53	403.98	504.55	23.31	415.54	385.77	444.14	14.00	91.83
potholes	1.5	943.65	880.91	1008.98	34.05	833.58	806.22	866.55	13.66	88.34
	2.0	735.18	668.96	811.13	39.39	631.18	583.72	668.48	13.75	85.85
	3.0	519.05	468.06	581.48	22.24	432.57	411.82	477.55	9.68	83.34
jari-huge	1.5	1039.03	982.64	1109.83	27.00	989.32	915.26	1024.49	25.54	95.22
	2.0	768.14	699.20	825.53	29.27	729.96	651.65	767.51	24.89	95.03
	3.0	482.81	439.45	537.68	20.50	450.74	405.82	475.48	14.67	93.36
warehouse	1.5	955.55	867.45	1054.84	50.50	933.95	861.72	1034.04	37.07	97.74
	2.0	773.22	726.74	855.34	33.74	762.96	732.83	815.42	19.11	98.67
	3.0	654.06	639.20	680.04	9.22	638.91	610.09	670.27	12.67	97.68

Table 3: Comparison of the greedy approach and the proposed algorithm.

rithms providing optimal solutions could be another interesting research line, and the results could be applied to the search problem in a-priori known environments.

Acknowledgments

The work of J.J. Miranda-Bront is partially supported by grants PICT-2011-0817, PICT-2013-2460 and project MEYS 2013 ARC/13/13. The work of M. Kulich is supported by the Czech Science Foundation (GAČR) under research project No. GA15-22731S and the Czech Ministry of Education, Youth and Sports under the project no. 7AMB14AR015 "Multi-Robot Autonomous Systems".

Access to computing and storage facilities owned by parties and projects contributing to the National Grid Infrastructure MetaCentrum, provided under the programme "Projects of Large Infrastructure for Research, Development, and Innovations" (LM2010005), is greatly appreciated.

References

[1] Hernán Abeledo, Ricardo Fukasawa, Artur Pessoa, and Eduardo Uchoa, *The time dependent traveling salesman problem: polyhedra and algorithm*, Mathematical Programming Computation **5** (September 2012), no. 1, 27–55.

[2] Francesco Amigoni, Alberto Quattrini Li, and Dirk Holz, *Evaluating the impact of perception and decision timing on autonomous robotic exploration*, Ecmr, 2013, pp. 68–73.

[3] David L. Applegate, Robert E. Bixby, Vasek Chvatal, and William J. Cook, *The traveling salesman problem: A computational study (princeton series in applied mathematics)*, Princeton University Press, Princeton, NJ, USA, 2007.

[4] Giorgio Ausiello, Stefano Leonardi, and Alberto Marchetti-Spaccamela, *On salesmen, repairmen, spiders, and other traveling agents*, Algorithms and complexity, 2000, pp. 1–16.

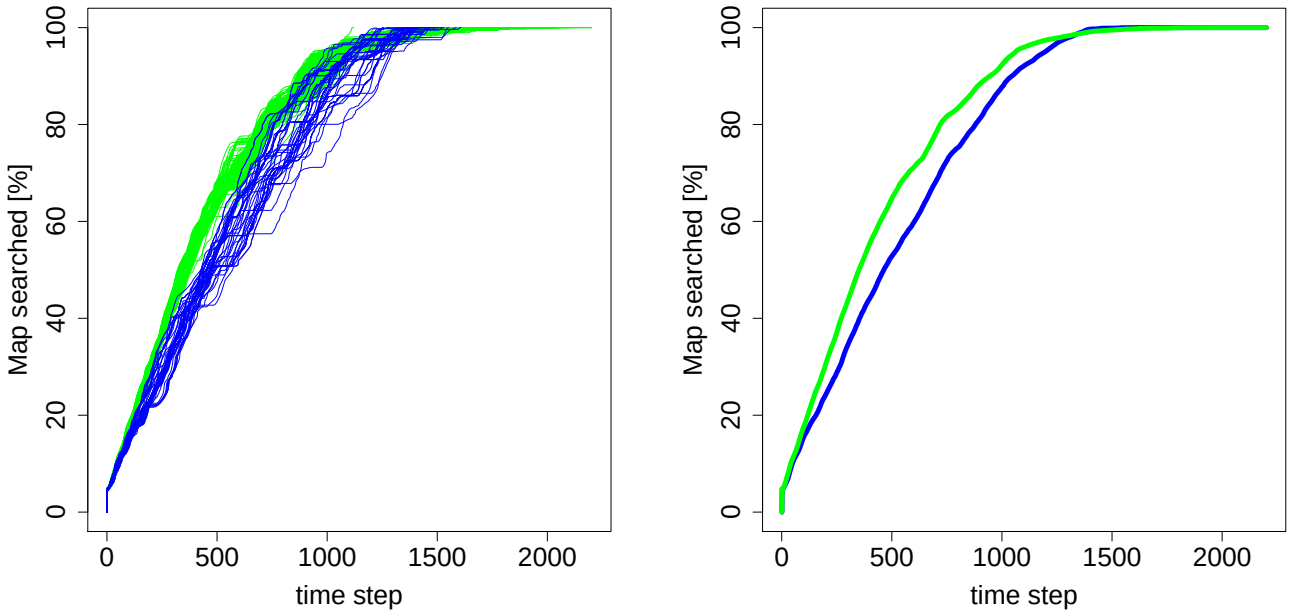

Figure 4: Relative amount of searched space during time for the *potholes* map and sensor range range=3 m. Left: progress of all 50 runs of the greedy approach (blue) and all 50 runs of the proposed method (green). Right: the average values over all 50 runs.

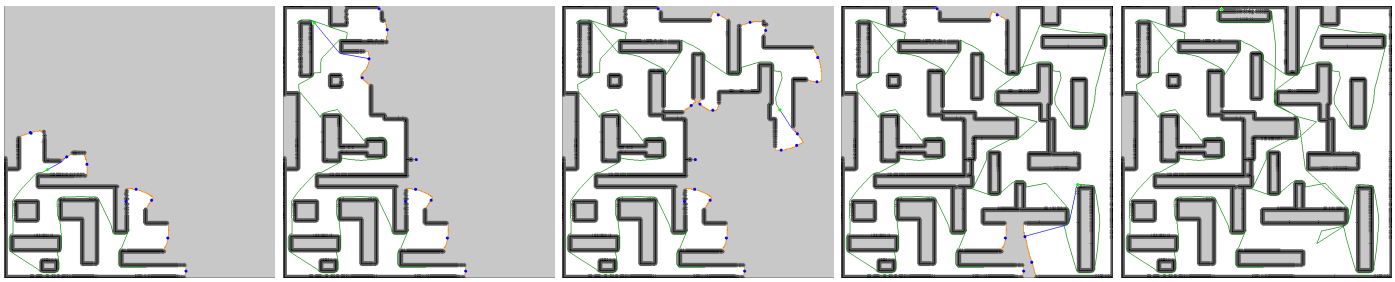

Figure 5: A typical progress of search by the greedy method in the *warehouse* environment and a sensor range=3 m.

[5] Nicola Basilico and Francesco Amigoni, *Exploration strategies based on multicriteria decision making for searching environments in rescue operations*, Autonomous Robots **31** (2011), no. 4, 401–417.

[6] T. Dewilde, D. Cattrysse, S. Coene, F.C.R. Spieksma, and P. Vansteenwegen, *Heuristics for the traveling repairman problem with profits*, Computers & Operations Research **40** (July 2013), no. 7, 1700–1707.

[7] Jan Faigl and Miroslav Kulich, *On benchmarking of frontier-based multi-robot exploration strategies*, European conference on mobile robots, 2015.

[8] Jan Faigl, Miroslav Kulich, and Libor Preucil, *Goal assignment using distance cost in multi-robot exploration*, Intelligent robots and systems (iros), 2012 ieee/rsj int. conf. on, 2012oct., pp. 3741 –3746.

[9] Thomas A. Feo and Mauricio G.C. Resende, *Greedy randomized adaptive search procedures*, Journal of Global Optimization **6** (1995), no. 2, 109–133.

[10] Matteo Fischetti, Gilbert Laporte, and Silvano Martello, *The delivery man problem and cumulative matroids*, Operations Research **March 2015** (1993), 1055–1064.

[11] Maria Teresa Godinho, Luis Gouveia, and Pierre Pesneau, *Natural and extended formulations for the Time-Dependent Traveling Salesman Problem*, Discrete Applied Mathematics **164** (February 2014), 138–153.

[12] Luis Gouveia and S Voß, *A classification of formulations for the (time-dependent) traveling salesman problem*, European Journal of Operational Research **2217** (1995), no. 93.

[13] Pierre Hansen and N Mladenović, *Variable Neighborhood Search*, Computers & Operations Research **24** (1997), no. 1, 1097–1100.

[14] Géraldine Heilporn, Jean-François Cordeau, and Gilbert Laporte, *The Delivery Man Problem with time windows*, Discrete Optimization **7** (November 2010), no. 4, 269–282.

[15] Keld Helsgaun, *An effective implementation of the Lin-Kernighan traveling salesman heuristic*, European Journal of Operational Research **126** (2000), 106–130.

[16] Geoffrey Hollinger, Joseph Djugash, and Sanjiv Singh, *Coordinated search in cluttered environments using range from multiple robots*, Field and service robotics, 2008, pp. 433–442.

[17] D. Karapetyan and G. Gutin, *LinKernighan heuristic adaptations for the generalized traveling salesman problem*, European

Journal of Operational Research **208** (February 2011), no. 3, 221–232.

[18] Elias Koutsoupias, Christos Papadimitriou, and Mihalis Yannakakis, *Searching a fixed graph*, Automata, languages and programming, 1996, pp. 280–289.

[19] Tomás Krajník, Miroslav Kulich, Lenka Mudrová, Rares Ambrus, and Tom Duckett, *Where's waldo at time t? Using spatio-temporal models for mobile robot search*, IEEE international conference on robotics and automation, ICRA 2015, seattle, wa, usa, 26-30 may, 2015, 2015, pp. 2140–2146.

[20] Miroslav Kulich, Jan Faigl, and Libor Preucil, *On distance utility in the exploration task*, Robotics and automation (icra), 2011 ieee int. conf. on, 2011, pp. 4455–4460.

[21] Miroslav Kulich, Libor Preucil, and Juan José Miranda-Bront, *Single Robot Search for a Stationary Object in an Unknown Environment*, 2014 IEEE International Conference on Robotics and Automation (2014), 5830–5835.

[22] S Lin and BW Kernighan, *An effective heuristic algorithm for the traveling-salesman problem*, Operations research (1973).

[23] Isabel Méndez-Díaz, Paula Zabala, and Abilio Lucena, *A new formulation for the traveling deliveryman problem*, Discrete Appl. Math. **156** (October 2008), no. 17, 3223–3237.

[24] Juan José Miranda-Bront, Isabel Méndez-Díaz, and Paula Zabala, *An integer programming approach for the time-dependent TSP*, Electronic Notes in Discrete Mathematics **36** (August 2010), 351–358.

[25] ———, *Facets and valid inequalities for the time-dependent travelling salesman problem*, European Journal of Operational Research (May 2013).

[26] Nenad Mladenović, Dragan Urošević, and Saïd Hanafi, *Variable neighborhood search for the travelling deliveryman problem*, 4OR (2012), 1–17.

[27] Motion planning maps. Accessed: September 30, 2015.

[28] Gerhard Reinelt, *Tsplib - a traveling salesman problem library.*, INFORMS Journal on Computing **3** (1991), no. 4, 376–384.

[29] Amir Salehipour, Kenneth Sörensen, Peter Goos, and Olli Bräysy, *Efficient GRASP+VND and GRASP+VNS metaheuristics for the traveling repairman problem*, 4OR **9** (2011), 189–209.

[30] A. Sarmiento, R. Murrieta, and S. Hutchinson, *An efficient motion strategy to compute expected-time locally optimal continuous search paths in known environments*, Advanced Robotics **23** (2009), no. 12-13, 1533–1560.

[31] Alejandro Sarmiento, Rafael Murrieta-Cid, and Seth Hutchinson, *A multi-robot strategy for rapidly searching a polygonal environment*, Advances in artificial intelligence - iberamia 2004, 9th ibero-american conference on ai, puebla, méxico, november 22-26, 2004, proceedings, 2004, pp. 484–493.

[32] K. S. Senthilkumar and K. K. Bharadwaj, *Multi-robot exploration and terrain coverage in an unknown environment*, Robotics and Autonomous Systems **60** (2012), no. 1, 123–132.

[33] T.C. Shermer, *Recent results in art galleries [geometry]*, Proceedings of the IEEE **80** (1992sep), no. 9, 1384 –1399.

[34] Marcos Melo Silva, Anand Subramanian, Thibaut Vidal, and Luiz Satoru Ochi, *A simple and effective metaheuristic for the Minimum Latency Problem*, European Journal of Operational Research **221** (September 2012), no. 3, 513–520.

[35] Brian Yamauchi, *Frontier-based exploration using multiple robots*, Proc. of the second international conference on autonomous agents, 1998, pp. 47–53.