sensors

MDPI

An Integrated Approach to Goal Selection in Mobile Robot Exploration

Miroslav Kulich [ID]***, Jiří Kubalík** [ID] **and Libor Přeučil**

Czech Institute of Informatics, Robotics, and Cybernetics, Czech Technical University in Prague, 160 00 Prague, Czech Republic; kubalik@cvut.cz (J.K.); preucil@cvut.cz (L.P.)

* Correspondence: kulich@cvut.cz

check for updates

Abstract: This paper deals with the problem of autonomous navigation of a mobile robot in an unknown 2D environment to fully explore the environment as efficiently as possible. We assume a terrestrial mobile robot equipped with a ranging sensor with a limited range and 360° field of view. The key part of the exploration process is formulated as the d-Watchman Route Problem which consists of two coupled tasks—candidate goals generation and finding an optimal path through a subset of goals—which are solved in each exploration step. The latter has been defined as a constrained variant of the Generalized Traveling Salesman Problem and solved using an evolutionary algorithm. An evolutionary algorithm that uses an indirect representation and the nearest neighbor based constructive procedure was proposed to solve this problem. Individuals evolved in this evolutionary algorithm do not directly code the solutions to the problem. Instead, they represent sequences of instructions to construct a feasible solution. The problems with efficiently generating feasible solutions typically arising when applying traditional evolutionary algorithms to constrained optimization problems are eliminated this way. The proposed exploration framework was evaluated in a simulated environment on three maps and the time needed to explore the whole environment was compared to state-of-the-art exploration methods. Experimental results show that our method outperforms the compared ones in environments with a low density of obstacles by up to 12.5%, while it is slightly worse in office-like environments by 4.5% at maximum. The framework has also been deployed on a real robot to demonstrate the applicability of the proposed solution with real hardware.

Keywords: path planning; routing; autonomous navigation; generalized traveling salesman problem; evolutionary algorithm

1. Introduction

Autonomous mobile robots solve complex tasks in many application areas nowadays. One of the most challenging scenarios is search and rescue missions in disaster areas involving large urban areas after an earthquake, a terrorist attack, or a burning house [1–3]. Other examples include humanitarian demining, Antarctic, underwater, and space exploration [4–7], or navigation in crowded environments [8,9]. The map of the working environment in the majority of these scenarios is not known in advance, or the environment dynamically changes significantly so that a priori knowledge is not useful. For the robot to behave effectively, it has to build a model of the environment from scratch during the mission, determine its next goal, and navigate to it. This iterative process is called *exploration*, and it is terminated whenever

the complete map is built. A natural condition is to optimize the effort needed to perform the exploration, e.g., to minimize the exploration time or the length of the traveled trajectory.

A determination of a next robot goal in each exploration iteration (one exploration step) is called an exploration strategy, which typically consists of three steps. A set of perspective goal candidates is generated first, followed by evaluation of the candidates based on the actual robot position, the current knowledge of the environment, and a selected optimization criterion (e.g., cost). The candidate with the best cost is selected as the next goal finally.

A realization of the exploration strategy is the key part of the exploration process as it influences exploration quality significantly—an inappropriate determination of the next goal to be visited may lead to revisiting of already explored places which increases the time needed to finish exploration. Therefore, the design of an efficient strategy that determines goals aiming to perform exploration with minimal effort plays an important role.

The exploration problem, and particularly exploration strategies, have been intensively studied in the last twenty years. The frontier-based strategy introduced in a seminal work of Yamauchi [10] considers all points on the *frontier*, which is defined as a boundary between a free and an unexplored space and navigates the robot to the nearest one. This approach has become very popular as it is simple to implement and it produces reasonable trajectories [11,12]. Several improvements were suggested. Holz et al. [10] segment an already known map, detect rooms in office-like environments and reduce the number of multiple visits of a room by a full exploration of the room once it is entered. Direction-based selection is presented in [13], where the leftmost candidate with respect to the robot position and orientation which is within the current sensing region is selected. If no such candidate exists, the algorithm picks the closest one.

While the approaches mentioned above use a single objective to evaluate candidates, another combine multiple criteria. For example, Gonzalez-Banos and Latombe in [14] mix an effort needed to reach a candidate within an expected area potentially visible from it. Similarly, expected information gain expressed as a change of entropy after performing the action is weighted with the distance to the candidate (distance cost) in [15,16], while information gain measured according to the expected posterior map uncertainty is used in [17]. Makarenko et al. [18] furthermore introduce the localization utility, which gives preference to places where the robot position can be accurately determined. Specification of a mixture function and its parameters is a crucial bottleneck of these approaches, and these are typically set up ad hoc. Basilico and Amigoni [19,20] therefore introduce a multi-criteria decision-making framework reflecting dependency on particular criteria. A technique that learns an observation model of the world by finding paths with high information content together with several weight functions evaluating goal candidates is introduced in [21]. An exploration framework based on the use of multiple Rapidly-exploring Random Trees has been introduced recently [22]. The authors define a revenue of a goal as a weighted sum of the information gained from exploring the goal and the navigation cost. Moreover, a hysteresis gain is added to prefer goals in robot's vicinity. A similar approach is proposed in [23], where a steering angle to the goal is integrated into the utility.

Contrary to the strategies mentioned above, which greedily select the candidate with best immediate cost, Tovar et al. [24] describe an approach where several exploration steps ahead are also considered. The algorithm goes through a tree structure representing all possible paths the robot may pursue in the given number of steps and searches for the best one employing the branch and bound algorithm. Although the experimental methodology is not clear, the results show that a greedy approach outperforms the proposed one. Moreover, the search state space is large and therefore there is a trade-off between the quality of the solution found and the time complexity and choice has to be made depending on the defined pruning depth.

An interesting research stream is a utilization of reinforcement learning. Zhu et al. [25] introduce Reinforcement Learning supervised Bayesian Optimization based on deep neural networks.

Similarly, Chen et al. [26] propose a learning-based approach and investigate different policy architectures, reward functions, and training paradigms.

A more sophisticated approach to next goal determination is presented in our previous paper [27]. It is based on the observation that the robot should pass or go nearby all the goal candidates and defines the goal selection problem as the Traveling Salesman Problem (TSP). That is, the cost of a candidate q is a minimal length of the path starting at the current robot position, continuing to the candidate q at first and then to all other candidates. It was shown that the introduced cost could reduce the exploration time significantly and leads to more feasible trajectories. The key part of this approach lies in the generation of goal candidates guaranteeing that all frontiers will be explored after visiting all the goal candidates. An ad hoc procedure is employed, which clusters frontier points by the k-means algorithm and generates candidates as centers of the clusters found. As k-means considers mutual distances of candidates only and does not take their visibility into account, it can rarely happen that frontiers are not fully covered due to occlusions (This does not influence completeness of the algorithm as it finishes when no unexplored area remains, and uncovered frontiers will be covered in the next exploration steps when occlusions disappear. On the other hand, the quality of the found solution can be degraded.). Moreover, the way how goal candidates are determined has no theoretical relation to the aim of exploration, i.e., traversing all the candidates does not lead to the shortest possible path that explores all frontiers. A similar approach was then used by Oßwald et al. [28], who run a TSP solver on a priori user-defined topological map. The authors, in consensus with our results, experimentally demonstrated that this method significantly reduces the exploration time.

Faigl and Kulich [29] formulate candidates' generation as a variant of the Art Gallery Problem with limited visibility, which aims to find a minimal number of locations covering all frontiers. The proposed iterative deterministic procedure called Complete Coverage follows the idea of a generation of samples covering free curves proposed by Gonzalez-Banos and Latombe [14] and guarantees full coverage of frontiers. Nevertheless, candidates generation and goal selection are still independent processes, i.e., candidates are not generated with respect to the cost of a path visiting all the candidates.

To the best of our knowledge, the only attempt to join these two processes into a single procedure is presented in Faigl et al. [30], where the goal selection task is formulated as the Traveling Salesmen Problem with Neighborhoods and a two-layered competitive neural network with a variable size to solve the problem is proposed. The presented results show that the approach is valid and it provides good results for open-space environments and longer visibility ranges and for office-like environments and small visibility ranges. On the other hand, the approach is very computationally demanding, which limits its deployment in real applications.

The research presented in this paper continues in the direction outlined in our previous works as it introduces an integrated solution to goal candidates' generation and goal selection. Novelty and contribution of the paper stand mainly in the following:

- We formulate the objective of the integrated approach to candidates generation and goal selection as the d-Watchman Route Problem, which enables a theoretically sound interpretation of the integrated approach to the goal determination problem. Our solution to the problem then leads to a definition of the objective of a goal selection itself as a variant of the Generalized Traveling Salesman Problem (GTSP).
- The introduced GTSP variant involves additional constraints to the original GTSP, which renders standard GTSP solvers inapplicable here. A novel evolutionary algorithm taking into account the added constraints is introduced. It uses an indirect representation and an extended nearest neighbor constructive procedure which circumvent the candidate solutions feasibility issue encountered when using traditional evolutionary algorithms with direct encodings.

- A novel approach for generating nodes/vertices for GTSP is presented as a mixture of the techniques for goal candidates generation mentioned above.
- The whole exploration framework is evaluated in a simulated environment and compared to the state-of-the-art methods. Moreover, we implemented the framework on a real robot to show that the proposed strategy is applicable in real conditions.

The fundamental single-robot exploration in a 2D environment, which is the main interest of the paper, has several extensions. One of these is the multi-robot case in which coordination of multiple robots is studied. Several goal selection strategies for multi-robot coordination based on different principles were introduced by many authors. A Markov process to model exploration using the transition probabilities to consider environment characteristics is proposed in [31], while a greedy approach is introduced in [32] and bio-inspired goal selection using a hybrid pheromone and anti-pheromone signaling mechanism is presented in [33]. A goal selection formulated as a multi-vehicle variant of the Travelling Salesman Problem is presented in [34], while several exploration strategies are compared in [29].

The previously mentioned exploration approaches assume that robot position is known, either provided by GPS or some SLAM (Simultaneous Localization and Mapping) algorithm. One of the first attempts to deal with localization uncertainty is presented in [35]. After that, several approaches employing Bayes filter were proposed [24,36,37]. With the increasing popularity of aerial robots and achievements in 3D sensing, exploration in three dimensions has been also studied (see [37–40]).

The rest of the paper is organized as follows. The problem definition is presented in Section 2. The proposed exploration framework is described in Section 3. The evolutionary algorithm designed for the constrained variant of the Generalized Traveling Salesman Problem and employed for goal determination is introduced in Section 4 and the experimental evaluation of the method is presented in Section 5. Finally, concluding remarks and future directions are summarized in 6.

2. Problem Definition

Assume a fully localized autonomous mobile robot equipped with a ranging sensor with a fixed, limited range (e.g., a laser range-finder) and 360° field of view operating in an unknown flat environment. Exploration is defined as the process in which the robot is navigated with the aim to build a map of the surrounding space to collect information about this space. The map is built incrementally as sensor measurements are gathered and it serves as a model of the environment for further exploration steps.

The whole exploration process is summarized in Figure 1. The algorithm consists of several steps that are repeated until no unexplored area remains. We assume that the environment is bounded. The exploration process is thus finished when there is no remaining unexplored area accessible to the robot (step 1). Accessibility is an essential condition as interiors of obstacles are inaccessible and thus remain unexplored.

The process starts with reading actual sensory information (step 2). The map is updated after some data processing and noise filtering (step 3). New goal candidates are determined afterward (step 4) and a next goal for the robot is assigned using a defined cost function (step 5). The shortest path from the robot's current position to the goal is found next (step 6) and the robot is navigated along the path (step 7).

While steps 2,3,6, and 7 belong to fundamental robotic tasks and many solutions to these already exist, the key part heavily influencing efficiency of the exploration process is the **next goal assignment problem** (step 5), formally defined as follows:

Given a current map $M \subset \mathcal{R}^2$, a robot position $p \in M$, and n goal candidates located at positions $G = \{g_1, \ldots, g_n\} \subset M$. The problem is to determine a goal $g \in G$ that minimizes the total required time (or the traveled distance) needed to explore the whole environment.

Goal assignment together with the determination of goal candidates in step 4 is called **exploration strategy**. More formally, exploration strategy aims to find a policy $\pi : M \times \mathcal{M} \rightarrow M$, where $\mathcal{M}$ is a set of all possible maps and $M \in \mathcal{M}$ is a map. Given the robot position $p \in M$ and the map $M \in \mathcal{M}$, the policy determines the next goal $g^* \in M$ so that the following cost is minimized:

$$\pi = \arg\min_a \sum_{t=0}^{T^a - 1} cost(p_t^a, p_{t+1}^a). \tag{1}$$

T^a is the time needed to explore the whole environment and p_{t+1}^a is robot position at time $t + 1$ when the policy a is followed. $cost(x, y)$ is a cost of movement from x to y. Notice that, if $cost(x, y)$ is set to $1 \quad \forall x, y$, then time is minimized, while setting $cost(x, y)$ as the distance of a path from x to y leads to optimization of a total travelled distance. Similarly to the exploration literature, we assume that a robot moves with a constant velocity and thus traveled distance linearly depends on a total traveling time so these two can be interchanged.
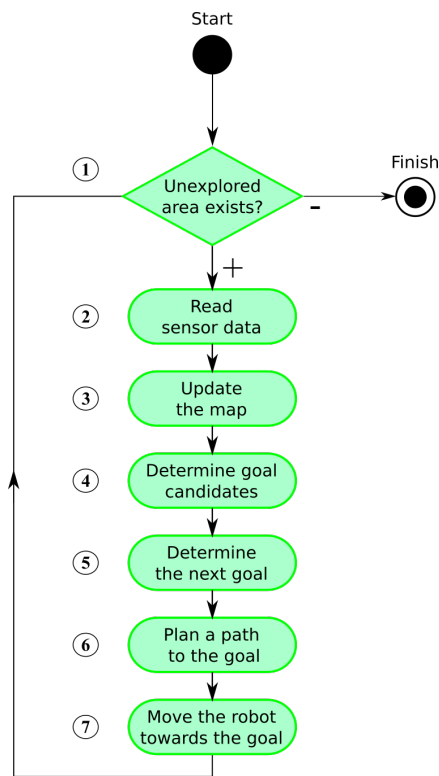
Start

① Unexplored area exists?   Finish   −

+

② Read sensor data

③ Update the map

④ Determine goal candidates

⑤ Determine the next goal

⑥ Plan a path to the goal

⑦ Move the robot towards the goal

Figure 1. The exploration process.

Determination of the optimal policy is not possible in general because *cost* and T^a cannot be exactly computed without knowledge of a structure of the unexplored environment. The aim is thus to find a strategy that leads to an exploration of the whole environment in the shortest possible time.

The objective of the exploration strategy is illustrated in Figure 2. Goal candidates (blue points) are generated on the border of an already explored area (white) and an unexplored space (grey) or in its vicinity. The goal assignment then selects the candidate minimizing some predefined penalty function as the next goal to which the robot is navigated (the red arrow points to the chosen goal). A typical penalty function is, for example, *distance*, i.e., the candidate nearest to the robot is selected.
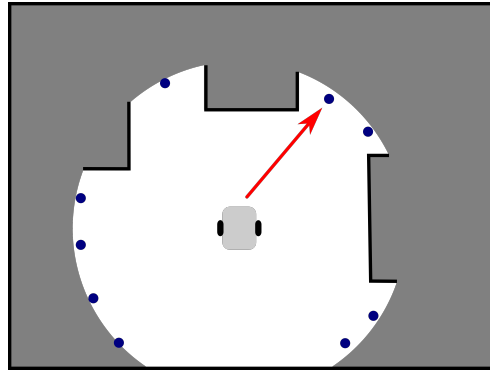
Figure 2. Illustration of the exploration strategy objective.

Realization of the particular steps of the exploration problem, especially the proposed exploration strategy, is detailed in Section 3.

Note that it is inefficient to invoke goal determination after the whole path is traversed and the current goal is reached. The experimental evaluation made by Amigoni et al. [41] indicates that it is better to run this decision-making process continuously, at some fixed frequency. The authors show that increasing decision frequency generally increases performance, but when the frequency is too high, performance degrades due to increased computational effort. Moreover, higher frequencies lead to oscillations of a robot's movement causing longer trajectories and exploration time. Best results in their paper were achieved with the frequency of 0.6 Hz, while the authors of [42] use the frequency of 0.25 Hz.

The exploration algorithm can be implemented in two threads: the first one communicating with the robot hardware and containing sensor processing, map building, and robot control with high frequency of units or tens of Hz and the decision-making process incorporating goal candidates generation, goal selection and path planning in the second one, which is triggered with a lower frequency.

3. Exploration Framework

The proposed exploration framework described in Algorithm 1 is derived from Yamauchi's frontier based approach [10]. The approach employs an occupancy grid [43] for map representation, which divides the working space into small rectangular cells. Each cell stores information about the corresponding piece of the environment in the form of a probabilistic estimate of its state, see Figure 4. Assuming that the map is static and individual cells are independent, a cell can be updated using a Bayes rule as described in [43]:

$$Bel(m_t^{[xy]}) = \eta p(z_t|m_t^{[xy]})Bel(m_{t-1}^{[xy]}),$$

where η is a normalization constant ensuring that probabilities of all possible states of $m_t^{[xy]}$ sum to 1, $p(z_t|m_t^{[xy]})$ is *a sensor model*, and $Bel(m_{t-1}^{[xy]})$ is the current believe in the state of $m^{[xy]}$ determined in the previous step. No a priori information about the environment is provided, therefore, $Bel(m_0^{[xy]})$ is set to $\frac{1}{2}$ for all cells.

A precision of contemporary ranging sensors is in order of centimeters, which is lower than or equal to the size of a grid cell. Therefore, a simple sensor model is used: $p(z_t|C) = O$ for the grid cell C corresponding to the sensor measurement, while $p(z_t|m^{[xy]}) = E$ for cells $m^{[xy]}$ lying on an abscissa between the current robot position S and C, see Figure 3. For a simulated or ideal sensor, we set $O = 1$, and $E = 0$. For real sensors, where a fusion of measurements during time suppresses an influence of sensor noise, the values are set closer to $\frac{1}{2}$, e.g., $O = 0.3$, and $E = 0.7$.
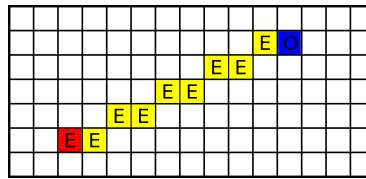
Figure 3. Sensor model. The robot (the red cell) measures an obstacle at the blue cell. The numbers in the cells represent sensor model values.

Occupancy grid cells are consequently segmented into three categories by application of two thresholds on their probability values: free (represented by a white color in Figure 4), occupied (black), and unexplored (light gray). The occupied cells are inflated by Minkowski sum (the dark gray areas in Figure 4) enabling to plan non-colliding paths for a circular-shaped robot (the robot does not collide with obstacles if its center lies outside inflated areas). The frontier based approach detects *frontier cells* (represented by a brown color in Figure 4), i.e., the reachable free grid cells that are adjacent with at least one cell that has not been explored yet (line 2). The *frontier* is a continuous set of frontier cells such that each frontier cell is a member of exactly one frontier.

Given determined frontiers, the exploration strategy determines a next goal the robot is navigated to as detailed in Section 3.1. The shortest path to this goal is determined by application of some standard planning algorithm like Dijkstra', A*, Voronoi Diagram or wave-front propagation techniques like Distance Transform [44]. Finally, navigation of the robot to the goal is realized by a local collision avoidance algorithm, e.g., Smooth Nearness-Diagram Navigation [45].
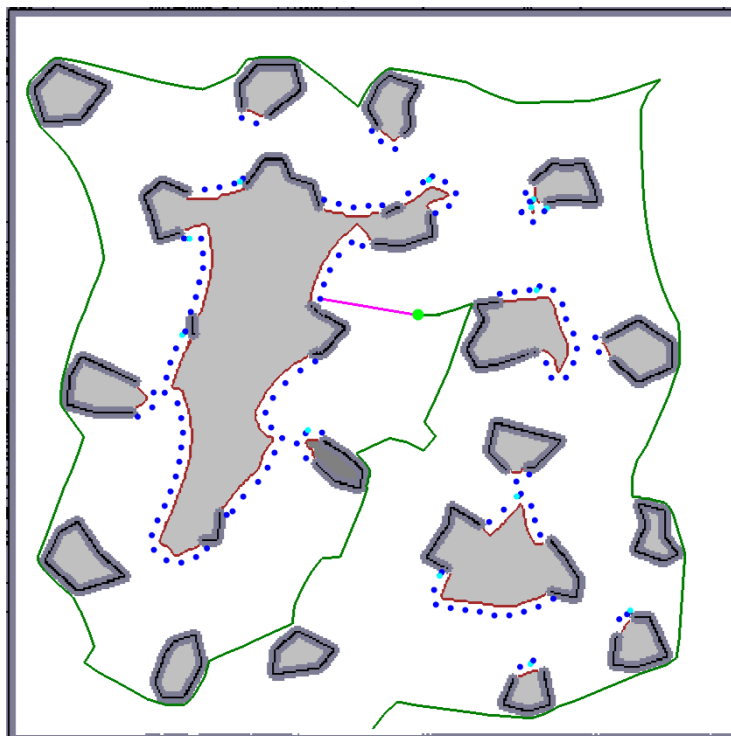

Figure 4. An exploration step: the robot (the green circle) selects one of the goal candidates (the blue dots; note that two different hues are used to visually distinguish overlapping dots.) as a next goal and plans a path to it (the pink curve). The green curve represents the already traversed trajectory.

3.1. Proposed Exploration Strategy

While Yamauchi [10] greedily selects the frontier cell nearest to the current robot position, the proposed approach is more sophisticated as depicted in Algorithm 1. The idea behind it is motivated by the fact that it is not necessary to visit a frontier to explore a new area surrounding it. Instead, visiting places from which all frontier cells are visible is sufficient. In fact, seeing slightly "behind" frontiers is requested to get new information. We ensure this by generating possible goals closer to frontiers than at visibility range as will be described below.

The realization of this idea leads to the formulation of next goal determination as a variant of the Watchman Route Problem [46]: Given a connected domain P, the Watchman Route Problem is to compute a shortest path or tour for a mobile guard that is required to see every point of P. In our case, seeing only all frontier cells (i.e., boundary of a domain containing all free cells) assuming limited visibility is requested only. Nevertheless, the problem, called as d-Watchman Route Problem, remains NP-hard [47] and therefore its solution is split into a solution of two sub-problems:

- **Generation**: of a set of goal candidates so that union of their visibility regions (i.e., areas visible from the particular goal candidates) contains all frontier cells, and
- **Construction** of a route connecting a subset of the goal candidates, whose visibility regions cover all frontier cells and a path traversing all of them from a current robot position is minimal.

An example of a set of goal candidates is given in Figure 4, where the candidates are represented as blue points. The purpose of the Generation phase is to construct a huge number of goal candidates which is then used as an input to the Construction phase and which guarantees that a feasible solution is constructed. This huge number of candidates, which is an order of magnitude higher than in [27], gives big flexibility to the Construction algorithm to produce good solutions of the superior d-Watchman Route Problem as it can choose from many possible candidates. Given an optimal solver in the Construction phase, a solution found by it converges with an increasing number of candidates to an optimal solution of the d-Watchman Route Problem.

3.1.1. Goal Candidates Generation

A naïve approach to the Generation problem takes all frontier cells as goal candidates. This leads to hundreds or thousands of goal candidates and high computational burden of determination of a path connecting them. Instead, the proposed algorithm generates an order of magnitude lower number of candidates. It starts with segmentation of an occupancy grid (Algorithm 1, line 1) and detection of all frontier cells (line 2). A connected string of frontier cells of each particular frontier is created then (line 3). This can be done by one of boundary tracking algorithms known from image processing for extracting boundaries for images. Namely, Moore-neighbor tracing algorithm [48] is employed in our case.

The ordering of frontier cells helps to efficiently determine goal candidates (lines 4–20), which is done for each frontier contour F in two stages.

In the first stage, candidates are generated uniformly. Note that the shape of a frontier is not smooth in general due to the discretization of the world into grid cells and sensory data noise. This often causes occlusions of frontier cells lying in a visibility range. Therefore, candidates are not generated on the frontier, but at the distance d from F, which is a fraction of a visibility range: F is inflated by a disk with a radius d using Minkowski sum (line 7), a contour of the inflated frontier is detected by the boundary tracking algorithm (line 8) and every k-th frontier cell of the contour is taken as a goal candidate (line 10). Coverage, i.e., frontier cells visible from the candidate, is computed (lines 11–12) and subtracted from the set of uncovered frontier cells (line 13), which is initially set to F (line 6).

It can happen that a frontier is, due to occlusions, not fully covered by visibility regions of candidates generated uniformly. Therefore, a dual sampling scheme is utilized to cover uncovered frontier cells: a random not yet covered cell is selected from the frontier F (line 15), a set of free cells visible from it is determined (line 16), and a goal candidate is generated randomly from these free cells (lines 16 and 17). Finally, the set of uncovered frontier cells in updated in the same way as in the first stage (line 20). The dual sampling is repeated until no uncovered frontier cell remains (line 14). Candidates generated from a single frontier F form *a cluster* G_F.

Algorithm 1: Proposed exploration strategy

Input: M – occupancy grid
δ – visibility range
r – current robot position
Output: g^* – next goal

---

1   $\left\langle M_{free}, M_{unknown}, M_{empty} \right\rangle \leftarrow$ segmented M
2   $E \leftarrow$ frontier cells of M
3   $\mathcal{F} \leftarrow$ contours of E
4   **foreach** $F \in \mathcal{F}$ **do**
5      $G_F \leftarrow \emptyset$ `// goal generated by F`
6      $U \leftarrow F$ `// uncovered frontier cells of F`
7      $I \leftarrow$ inflate F
8      $T \leftarrow$ contours of I
9   **foreach** *k-th* $g \in T$ **do**
10         $G_F \leftarrow G_F \cup \{g\}$
11         $V \leftarrow M_{free} \cap R(g)$, where
            $R(g)$ is a set of cells visible from g
12         $g_{coverage} \leftarrow F \cap R(g)$
13         $U \leftarrow U \setminus g_{coverage}$
14   **while** $U \neq \emptyset$ **do**
15         $s \leftarrow$ random cell from U
16         $V \leftarrow M_{free} \cap R(s)$, where
            $R(s)$ is a set of cells visible from s
17         $g \leftarrow$ random cell from V
18         $G_F \leftarrow G_F \cup \{g\}$
19         $g_{coverage} \leftarrow F \cap R(g)$
20         $U \leftarrow U \setminus g_{coverage}$
21      $\mathcal{G} \leftarrow \mathcal{G} \cup G_F$
22   $D \leftarrow$ distance matrix of goals in $\mathcal{G}$
23   $tour \leftarrow f(\mathcal{G}, D)$
24   $g^* \leftarrow$ second node of *tour*
25   **return** g^*

3.1.2. Route Construction

The Route construction phase starts with the computation of a distance matrix of all generated goal candidates by running Dijkstra's algorithm [49] on an adjacency graph of the free cells for each candidate (line 22). Note that utilization of all-pairs shortest path algorithms (e.g., Johnson's [50] with $O(VE \log V)$, where V is a number of vertices and E is a number of edges) is not effective, as these compute distances

among all free cells, while complexity of K runs of Dijkstra's algorithm is $O(KE + KV \log V)$, where a number of goal candidates $K << N$.

Finally, the determined set of clusters, each containing a set of corresponding goals with their coverages, and the distance matrix of the goals are used to find the best candidate as the next goal to move to in the next iteration of the exploration process (line 23–25). The problem of finding such goal involves the solution of an optimization problem of constructing a shortest possible tour starting from the current robot position and leading through a subset of goals subject to the constraint that all frontier cells are covered by the selected goals (line 23). The next goal g^* is then determined as the second node (Note that the first node of the tour is robot's position.) of *tour* (line 24), which is returned (line 25).

The optimization problem at line 23 can be formulated as a variant of the GTSP where the formulation of the general GTSP is: Given a set G of nodes partitioned into m non-empty clusters and a matrix D of their mutual distances, the objective is to find a subset S of G so that (a) S contains *exactly one* node from each cluster and (b) a tour connecting all nodes from S is the shortest possible. The GTSP is known to be NP-hard as it reduces to the NP-hard TSP when each cluster consists of exactly one node.

Here, we solve a constrained variant of the GTSP where the final tour contains *at least one node from each cluster* and must satisfy a constraint that all frontiers are fully covered by goals visited within the tour. To formulate the problem formally, assume

- a set of m frontiers $\mathbf{F} = \{F^i\}_{i=1}^{m}$, where each frontier F^i is a set of frontier cells $F^i = \{f_j^i\}$,
- a set of n goal candidates $\mathbf{G}$ clustered into m clusters. Each cluster $G^i = \{g_j^i\}_{j=1}^{n_i}$ associated to a frontier F^i is represented by n_i candidate goals,
- coverage $\mathbf{R}(g_j^i) \subset F^i$ for each g_j^i,
- a current position r of the robot,
- a matrix D of mutual distances between all goal candidates.

The aim is to find a tour t such that

$$t = \arg\min_{\mathbf{T}} \sum_{k=0}^{p-1} D(g_{j_k}^{i_k}, g_{j_{k+1}}^{i_{k+1}}), \tag{2}$$

where $\mathbf{T}$ is a set of all tours $\tau = \left\langle g_{j_0}^{i_0}, g_{j_1}^{i_1}, g_{j_2}^{i_2}, \ldots, g_{j_p}^{i_p} \right\rangle$ satisfying

$$g_{j_0}^{i_0} = r \quad \text{and} \tag{3}$$

$$\bigcup_{k=0}^{n} \mathbf{R}(g_{j_k}^{i_k}) = \bigcup_{i=1}^{m} F^i \tag{4}$$

We call this problem Generalized Travelling Salesman Problem with Coverage and denote it GTSPC.

A lot of attention has been paid by researchers to solve the GTSP. There are approaches transforming the GTSP to the TSP and solving the related TSP [51,52]. Other approaches formulate the GTSP as an integer linear program (ILP) and employ exact algorithms for solving the ILP [53–55]. Due to the NP-hardness of the GTSP problem, many heuristic, metaheuristic and hybrid approaches have been developed in the past decade as well. For example, a metaheuristic global search based on imperialist competitive algorithm inspired by a socio-political scheme and hybridized with a local search procedure is introduced in [56]. El Kraki et al. [57] present a simple heuristic that clusters input cities, finds their barycenters to determine an order of the clusters and determines the best city of each cluster. [58] provides a survey of neighborhoods and local search algorithms for the GTSP.

A hybrid genetic algorithm with the random-key indirect representation that enforces only feasible solutions created by genetic operators has been proposed in [59]. A memetic algorithm with a powerful local search procedure making use of six different local search heuristics was proposed in [60]. Various versions of Ant Optimization Algorithms were used to solve the GTSP as well [61–63].

In this work, we assume the GTSPC where the tour can contain one or more cities from each cluster. This is implied by the constraint imposed on the selected subset of goal candidates that they must completely cover all frontiers. All published approaches to the GTSP consider the unconstrained case, where the tour contains exactly one city from each cluster. These approaches thus do not provide valid solutions for the considered GTSPC and it is not possible to adapt them straightforwardly to this problem. We, therefore, propose an evolutionary algorithm, described in the following section, to efficiently solve the constrained variant of the GTSP–GTSPC.

4. Proposed Approach to Solve the GTSPC

The exploration strategy aims to determine a goal to which the robot will move next, which leads to finding a shortest possible tour through a properly selected subset of goals as discussed in Section 3.1.1 (see lines 23–24 of Algorithm 1). We defined this optimization problem as GTSPC, i.e., the next goal is the second node of a tour starting from the current position and minimizing Equation (2) while satisfying Equations (3) and (4).

The proposed approach to solve GTSPC is based on the evolutionary algorithm with indirect representation and extended nearest neighbor constructive procedure (IREANN), originally proposed for a symmetric TSP [64]. IREANN is an evolutionary algorithm particularly suited for solving routing and sequencing problems. When searching for a good solution to the routing problem, it directly exploits the short or low-cost links. Section 4.1 describes the main idea behind IREANN and basic components of the algorithm on an example of the TSP. The proposed adaptation of the algorithm to the GTSPC variant considered in this work is introduced in Section 4.2.

4.1. Original IREANN Algorithm

IREANN can be considered an extension of the *nearest neighbor* (NN) constructive algorithm used for solving the routing problems such as the TSP. The main idea is to explore a larger set of candidate solutions than the NN algorithm while making use of the shortest links as much as possible.

The standard NN algorithm starts the tour in a randomly chosen working city, s. Then, it repeatedly connects city s with its nearest neighbor v chosen from the set of available nodes at the moment **A**

$$v \leftarrow \arg\min_{a \in \mathbf{A}} dist(s, a) \tag{5}$$

and the city v becomes the current working city s for the next iteration. The process stops when all cities have been visited. We can generate up to N different tours, where N is the number of cities considered, and take the best tour as the final solution. This algorithm is easy to implement, quickly yields a short tour, but it produces suboptimal tours very often due to its greedy nature. An example of such a case is illustrated in Figure 5 with an optimal solution (Figure 5a) and two suboptimal solutions generated by the NN algorithm started from node B (Figure 5b) and H (Figure 5c), respectively. Note that the algorithm initialized in either of the ten starting nodes is not able to produce the optimal solution.

The NN algorithm can be extended so that in each step of the tour construction process multiple choices will be considered for selecting the current working city, s. In particular, one city out of the set of not yet fully connected cities is chosen to be linked within the developed solution using the nearest neighbor heuristic. By the term *fully connected node*, we mean the node that already has two edges—the

incoming and outgoing one—assigned. This way, multiple subtours can be developed simultaneously, which are gradually connected together resulting in the final single tour. The order in which cities are processed by such an **extended nearest neighbor** constructive procedure (ENN) determines the tour generated. When the cities are presented to (ENN) procedure in the right order, an optimal solution can be produced. This is illustrated in Figure 6 where the optimal solution is produced by processing cities in the order AJCEIFBHGD.
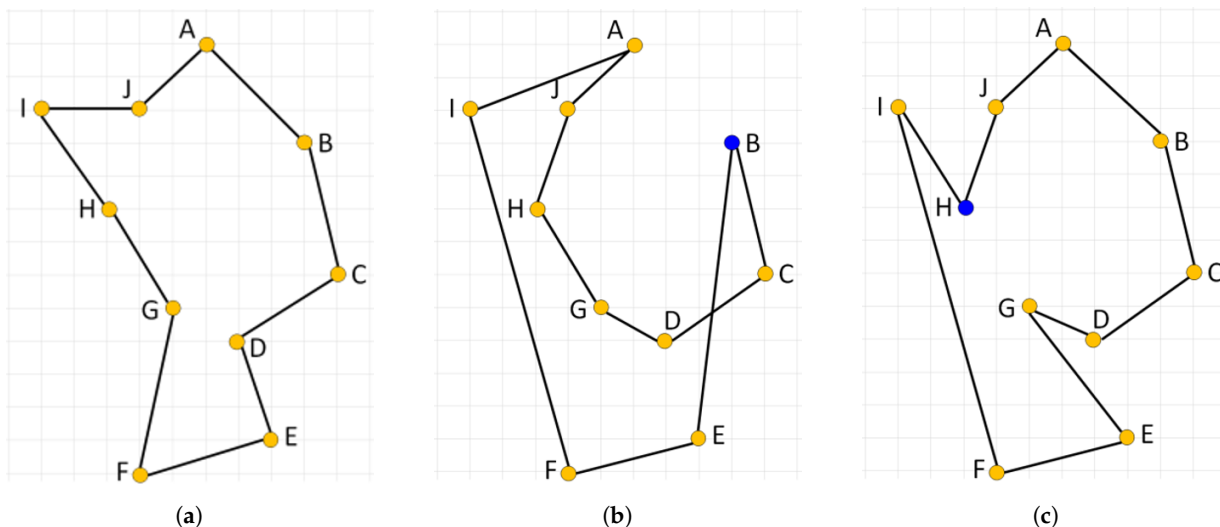

Figure 5. Illustration of an ineffectiveness of the standard nearest neighbor constructive procedure on a simple instance of the TSP. (**a**) shows an optimal solution; (**b**) shows a solution constructed by the nearest neighbor when started from node B; (**c**) shows a solution constructed when the procedure starts from node H.
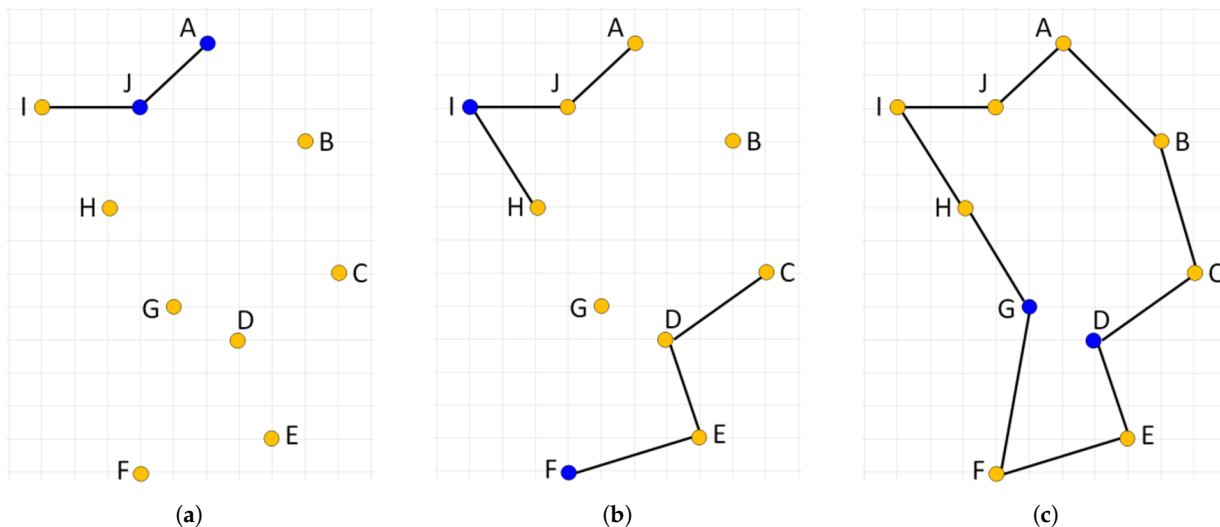

Figure 6. Extended nearest neighbor constructive procedure applied to cities in the following order AJCEIFBHGD. (**a,b**) show a partial solution after processing nodes {A, J} and {A, J, C, E, I, F}, respectively. (**c**) shows the final solution. The last two nodes linked into the solution in the given stage of the construction process are shown in blue.

IREANN is an ordinary evolutionary algorithm using a fixed-length linear representation that exploits the idea of the ENN procedure in the following way. Individuals evolved in the population do not represent tours directly as particular sequences of cities. Instead, each individual represents an input sequence to the ENN procedure called *priority list* that determines the order or priorities with which the cities will be processed by the ENN procedure. An important aspect related to this indirect representation is that a single tour can be represented by many different priority lists. This means there are multiple attractors to which the evolutionary algorithm can converge. On the other hand, some tours, including the optimal one, might get unreachable with the indirect representation. Nevertheless, any solution reachable by the standard NN procedure is reachable with the ENN procedure as well.

IREANN algorithm. The pseudo-code of IREANN is shown in Algorithm 2. It starts with a random initialization of the population of candidate priority lists (line 1). Then, each individual is evaluated (line 2), i.e., a tour is constructed using the ENN procedure applied to its priority list (described in Algorithm 3) and its length is used as the individual's fitness value. After the population has been evaluated, the counter of fitness function evaluations is set to the population size (line 3). Then, the algorithm runs until the number of calculated fitness evaluations reaches the maximum number of fitness evaluations (lines 4–16). The first parental individual is selected in each iteration using a tournament selection (line 5) and a new individual is created using either crossover (lines 7–10) or mutation (lines 12–13) operator. When the crossover operator is chosen with the crossover rate P_C, the second parental individual is selected (line 3) and the offspring produced undergoes the mutation operation (lines 9–10). The newly created individual is evaluated and the worst individual in the population is replaced by it (line 16).

Algorithm 2: Evolutionary algorithm for the optimal tour generation problem.

Input: *POPSIZE* – population size
      MAXEVALS – maximum number of fitness function evaluations
      P_C – crossover rate
      P_M – mutation rate
Output: *tour* – the shortest tour found

---

1   init(*population*)
2   evaluate(*population*)
3   *evaluations* ← *POPSIZE*
4   **while** *evaluations* < *MAXEVALS* **do**
5   *par*1 ← select(*population*)
6   **if** rand < P_C **then**
7   *par*2 ← select(*population*)
8   *offspring* ← crossover(*par*1, *par*2)
9   **if** rand() < P_M **then**
10   mutate(*offspring*)
11   **else**
12   *offspring* ← *par*1
13   mutate(*offspring*)
14   evaluate(*offspring*)
15   replace(*population*, *offspring*)
16   *evaluations* ← *evaluations* +1
17   *tour* ← getBest(*population*)
18   **return** *tour*

In the end, the tour generated by the best-fit individual in the population is returned (line 18). Note that the ultimate output of the algorithm, when used to solve the GTSPC, is the candidate goal to which the robot is going to move in the next step of the environment exploration. Thus, just the first goal in the tour of the best fit individual will be returned in the end.

IREANN evolutionary operators. The evolutionary algorithm uses the *order-based crossover* defined in [65] and a simple *point mutation*. The crossover works so that a set of goals randomly chosen from the priority list of the first parent are copied to the offspring into the same positions as they appear in the first parent. Remaining positions are filled in with goals taken in the same order as they appear in the priority list of the second parent.

The crossover is illustrated in Figure 7, where two priority lists parent1 $= [7, 6, 8, 2, 1, 3, 5, 4]$ and parent2 $= [5, 4, 1, 3, 2, 8, 7, 6]$ are crossed over. First, a group of goals $\{6, 2, 3, 5\}$ is inherited from parent1. Then, the remaining positions of the offspring are filled in with goals $\{4, 1, 8, 7\}$ while respecting their relative order in parent2.

IREANN uses a simple point mutation operator that randomly chooses one city in the priority list and moves it to an arbitrary position.
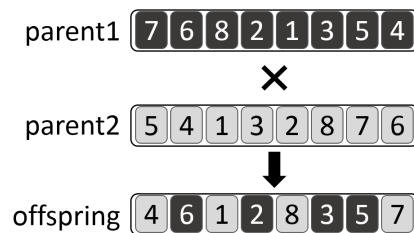

Figure 7. Order-based crossover operator.

4.2. IREANN Adaptation for GTSPC

This section describes modifications related to adaptation of the IREANN algorithm to the GTSPC, namely the representation and the tour construction procedure making use of the ENN procedure and taking into account the constraint that all frontiers are fully covered.

Representation. A candidate solution is a tour through a subset of candidate goals $\mathbf{G}' \subseteq \mathbf{G}$ such that the union of visibility regions of goals $g \in \mathbf{G}'$ covers all frontier cells. Generally, there might be multiple subsets of goals that produce a feasible solution. Thus, the representation should cover all the possibilities. Therefore, the representation used in this work is a *priority list*, i.e. the permutation, overall candidate goals in the original set $\mathbf{G}$. Note that, even though the priority list contains all candidate goals, the resulting tour can be composed of just a subset of these goals, as described in the following paragraphs.

Tour construction procedure. For the sake of efficiency, the set of frontiers $\mathcal{F}$ is split into two sets—a set $\mathcal{F}_{\mathcal{N}}$ of K frontiers nearest to the current position of the robot and a set $\mathcal{F}_{\mathcal{D}}$ of remaining distant frontiers—which are treated differently. For each frontier in the set $\mathcal{F}_{\mathcal{D}}$, a tour component C is constructed using the standard nearest neighbor algorithm such that C consists of a minimal set of goals completely covering the given frontier. The set of disjoint tour components, $\mathcal{C}$, constitutes a so-called *embryo*, which is created once before the evolutionary optimization starts. The tour components are immutable, only their connection within the whole tour is subject to further optimization.

The embryo plus individual goals belonging to frontiers from $\mathcal{F}_{\mathcal{N}}$ are passed as the input to the construction of the whole tour using the ENN procedure described in the next paragraph. Using the embryo leads to the search space reduction and possibly increased efficiency of the evolutionary algorithm.

The idea behind this two-step construction strategy is that the optimal configuration of goals (i.e., their selection and interconnection) covering the frontiers close to the current robot position is crucial for

the selection of the proper goal to move to in the next iteration of the exploration process. On the other hand, sub-optimally connected goals belonging to frontiers far from the starting robot position have a small impact on the next goal decision-making.

ENN procedure. The ENN procedure takes a priority list P and the initial set of tour components $\mathcal{C}$ as input and produces a complete tour through the goals so that all frontier cells are visible from (or covered by) at least one of the used goals. The procedure starts with the initialization of the set of uncovered frontier cells with the set of all frontier cells of uncovered frontiers (line 1). The number of available connections of each goal is set to 2 (line 2). The value of $availConn[g_i]$ indicates whether the goal g_i is already fully connected in the generated tour (i.e., $availConn[g_i] = 0$) or it is partly connected with one link and the other link is still available (i.e., $availConn[g_i] = 1$) or it is not used in the generated tour yet (i.e., $availConn[g_i] = 2$). Then, the algorithm iterates through lines 4–32 until a single tour has been constructed that covers all frontiers' cells. In each iteration i, the goal to be processed g is taken from the priority list P (line 5). If g is not yet used in the partial solution and the set of frontier cells visible from g does not contain any of the uncovered frontier cells, then it is skipped as it does not contribute to the overall coverage of the solution (lines 7–8). Otherwise, g is used to extend the current partial solution in three possible ways:

1. The goal g is not connected in the solution yet. Its nearest available neighbor goal g_{nn} is found, the two goals g and g_{nn} are connected, the set of uncovered frontier cells is updated accordingly and the available connections of g and g_{nn} are decremented (lines 9–14).
2. The goal g is already linked to one other goal. Its nearest available neighbor goal g_{nn} is found, the two goals g and g_{nn} are connected and the available connections of g and g_{nn} are updated (lines 15–19). The set of uncovered frontier cells remains unchanged.
3. The goal g is already fully connected, i.e., it is linked to two other goals in one tour component C^g. First, the nearest available neighbors $start_{nn}$ and end_{nn} of the component's boundary goals are found. Then, the shorter link of the two possible links $(start, start_{nn})$ and (end, end_{nn}) is added to the solution and the available connections of newly connected goals are updated (lines 20–32).

Note that the nearest available neighbor is such a goal that is either already connected within the solution and has at least one connection available or is not used in the solution yet and can reduce the number of uncovered frontier cells at least by one.

Figure 8 illustrates the three possible ways the partial solution can be extended. It shows a single frontier and six candidate goals sampled for this frontier and assumes the priority list P $=$ $[g_1, g_6, g_4, g_2, g_3, g_5]$ is used to construct a tour covering this frontier. A circle around each goal g_i indicates a visibility range of the goal and the portion of a frontier covered by the circle contains the set of frontier cells covered by the goal $\mathbf{R}(g)$. Figure 8b depicts the partial solution obtained after the goals g_1 and g_6 have been processed by the ENN procedure. Both were added to the solution by applying lines 9–14 of Algorithm 3. At that point, goal g_4 is going to be processed by ENN procedure. It is not yet used in the partial solution and it can still contribute to the overall coverage of the solution if added to it, so the goal will be processed using lines 9–14 as well. Assuming that $distance(g_4, g_5)$ is less than $distance(g_4, g_2)$, goal g_5 is the nearest available neighbor of g_4, resulting in partial solution in Figure 8c. The goal g_3 was not available since it does not contribute to the overall coverage of the constructed solution.

Algorithm 3: Extended nearest neighbor constructive procedure for the feasible tour generation problem.

Input: $\mathbf{G} = \{g_1, \dots, g_n\}$ – set of n goals
  $\mathcal{C}$ – set of initial tour components, i.e. the embryo
  $\mathcal{F}_\mathcal{U}$ – set of uncovered frontiers
  D   – $n \times n$ distance matrix
  P   – priority list
  r – current robot position
Output: *tour* – tour through a subset of candidate goals

---

1   $\mathbf{U} \leftarrow \mathcal{F}_\mathcal{U}$
2   `init`($availConn$)
3   $i \leftarrow 0$
4   **while** $\mathbf{U} \neq \emptyset$ **do**
5    $g \leftarrow \text{P}[i]$
6    $i \leftarrow i + 1$
7   **if** $availConn[g] = 2$ **and** $\mathbf{R}(g) \cap \mathbf{U} = \emptyset$ **then**
8   continue
9   **if** $availConn[g] = 2$ **then**
10     $g_{nn} \leftarrow$ `getNearestN`(g)
11     $\mathbf{U} \leftarrow \mathbf{U} \setminus (\mathbf{R}(g) \cap \mathbf{U})$
12   `connect`(g, g_{nn})
13   `decrement`($availConn[g]$)
14   `decrement`($availConn[g_{nn}]$)
15   **else if** $availConn[g] = 1$ **then**
16     $g_{nn} \leftarrow$ `getNearestN`(g)
17   `connect`(g, g_{nn})
18   `decrement`($availConn[g]$)
19   `decrement`($availConn[g_{nn}]$)
20   **else**
21     $C^g \leftarrow$ `getComponent`(g)
22     $\{start, end\} \leftarrow$ `getBoundary`(C^g)
23     $start_{nn} \leftarrow$ `getNearestN`($start$)
24     $end_{nn} \leftarrow$ `getNearestN`(end)
25   **if** $\text{D}[start, start_{nn}] < \text{D}[end, end_{nn}]$ **then**
26   `connect`($start, start_{nn}$)
27   `decrement`($availConn[start]$)
28   `decrement`($availConn[start_{nn}]$)
29   **else**
30   `connect`(end, end_{nn})
31   `decrement`($availConn[end]$)
32   `decrement`($availConn[end_{nn}]$)
33   `oneopt`(*tour*)
34   `twoopt`(*tour*)
35   **return** *tour*

The next goal in the priority list is g_2, which is already partly connected in the solution. However, it has one connection available still. Thus, its nearest available neighbor is found, i.e., goal g_4, and the two are connected resulting in the solution in Figure 8d. These actions correspond to lines 15–19.

The next goal in the priority list is g_3. However, this goal is not in the solution yet, and it would not contribute to the overall coverage of the constructed solution if it were added to. Thus, it is skipped.

The remaining goal, g_5, is already fully connected in the solution and it will be processed according to lines 20–32. As it cannot be connected itself to any other goal, the nearest neighbors of the boundary goals g_1 and g_6 of the component $C^{g_5} = [g_1, g_2, g_4, g_5, g_6]$ will be found. Then, the shorter link leading from the boundary goals will be added to the solution (not shown in Figure 8).

Figure 8e shows another way the frontier f can be covered if the priority list $\text{P} = [g_1, g_6, g_4, g_2, g_3, g_5]$ was used. This time, the goal g_4 would be omitted.

Once a single tour covering all frontier cells is constructed, the tour is further refined by one-opt and two-opt heuristics (lines 33–34). These two refinement heuristics operate on a fixed set of goals and just their order can be modified.
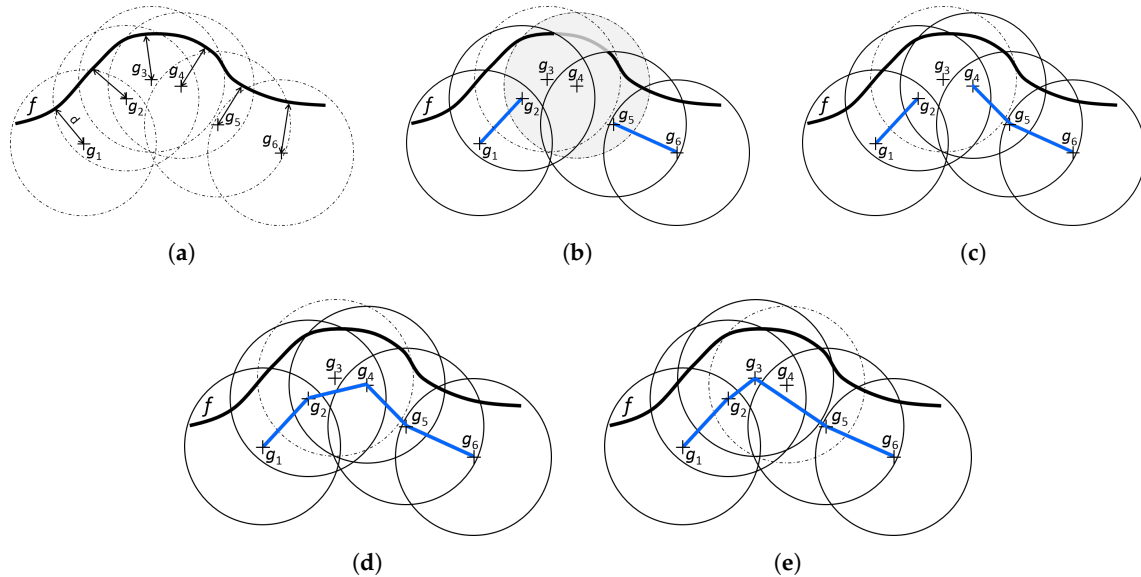

Figure 8. An example with a single frontier, six candidate goals equally distant from the frontier and the priority list $P = [g_1, g_6, g_4, g_2, g_3, g_5]$. (**a**) the scenario; (**b**,**c**) partial coverages of frontier f after applying first two and three steps of the ENN procedure to goals g_1, g_6 and g_4; (**d**) the complete coverage of frontier f induced by processing of the goals in the order g_1, g_6, g_4 and g_2; (**e**) an alternative coverage of frontier induced by processing of the goals in the order g_1, g_2, g_6 and g_5.

5. Experimental Evaluation

5.1. Simulations

Performance of the proposed evolutionary algorithm as a goal selection strategy in the exploration framework was statistically evaluated and compared with three state-of-the-art approaches. The first one is Yamauchi's greedy approach (greedy) [10], probably the most popular and used strategy nowadays—see, e.g., [66–68]. The second one is our previous strategy [27], which minimizes the cost over several steps. As the number of steps corresponds to the number of candidates, we call the strategy Full Horizon Planning (FHP). To the best of our knowledge, FHP is currently one of the best strategies which do not use background knowledge. For example, although the recently published approach [28] benefits from a priori known topological map of the environment, it produces similar results to FHP in similar environments in comparison to greedy. The third method is an information-based strategy [22] that determines the cost of a candidate as a weighted sum of the information gained from visiting the goal and the Euclidean distance to the candidate. A hysteresis gain is added to prefer goals in robot's vicinity. This method is further referred to as UMARI.

The methods differ in the evaluation of goal candidates as well as in the generation of these. While greedy selects the nearest cell among all frontiers cells, FHP clusters frontier cells using k-means, which is also used in our implementation of UMARI. The proposed approach (EA) employs a two-stage sampling process as described in Section 3.

The comparison follows Level-0 of the methodology presented in [44], which suggests studying theoretical behavior of methods without the influence of sensor noise, localization imprecision,

and inaccuracies of motion control. Therefore, we employ our robotic simulator, in which the complete exploration framework with the above-mentioned strategies integrated was implemented in C++ except the evolutionary algorithm itself, which was done in Java.

Three maps from Motion Planning Maps Dataset [69] scaled to 20 m × 20 m representing various types of environments were chosen for comparison. The first one (*empty*) is a map without obstacles giving a robot high flexibility in motion. The *potholes* map represents an unstructured space with about 20 small obstacles. Finally, *jari-huge* is a map of an administrative building with many rooms and corridors between them.

A sensor with 360° field of view with various visibility ranges $\rho \in \{1.5, 2.0, 3.0, 5.0, 10.0\}$ (in meters) was used, while an occupancy grid with cell size 0.05×0.05 m was chosen to represent the working environment. Robot size was 0.1 m, the inflation radius was, $d = 0.25$ m, the distance for the uniform sampling was set to 0.25 m, and every fourth cell was considered as a goal candidate (constant k in Algorithm 1).

The evolutionary algorithm was run with the following configuration: population size = 200, maximal number of fitness evaluations = 3000, crossover rate = 80%, mutation rate = 25%, number of nearest frontiers $K = 5$, and tournament size = 3. For the UMARI strategy, the settings recommended by the authors is used: $\lambda = 3$, $h_{gain} = 2$, $h_{rad} =$ sensor range (see [22] and https://github.com/hasauino/rrt_exploration).

Fifty trials were run for each combination ⟨map, range, method⟩, which gives 2250 trials in total. The obtained results for greedy, FHP and EA are statistically summarized in Table 1, where *avg* stands for the time needed to explore the whole environment (exploration time T_{exp}) averaged over all 50 runs, *min* and *max* are minimal and maximal T_{exp} over these runs and *stdev* stands for the standard deviation of T_{exp}. R_{gr} expresses a ratio of the average T_{exp} obtained with the proposed method to the average T_{exp} obtained with the greedy one. Similarly, R_{FHP} is a ratio of average T_{exp} values of EA and FHP. A value of less than 100% indicates the proposed approach is better than the respective compared one, and vice versa. Differences between the average T_{exp} values of the compared algorithms were evaluated using a two-sample *t*-test with the significance level $\alpha = 0.01$. The null hypothesis being the two data vectors are from populations with equal means. Results of the statistical tests are presented in the last column of the table. A sign '+' means the average value obtained with the proposed algorithm is significantly better than the one of the compared algorithm. A sign '-' indicates the opposite case. A situation when the two compared means are statistically indifferent is indicated by the '=' sign.

It can be seen that FHP and EA significantly outperform the greedy approach for the *empty* map as they benefit from longer planning horizon allowing them to explore the space systematically. Moreover, EA provides better results than FHP, especially for $\rho = 3.0$, where the difference is more than 22%. This is because EA directs a robot to a distance ρ from obstacles contrary to FHP, where k-means generates goals to be visited closer to obstacles—see Figure 9a.

Table 1. Comparison of the exploration strategies.

| Map	ρ	Greedy				FHP-Based				EA-Based				R_{gr} %	R_{FHP} %	+/−
		avg	min	max	stdev	avg	min	max	stdev	avg	min	max	stdev			
empty	1.5	2419.70	2171.00	2701.00	111.43	2241.90	2041.00	2436.00	89.45	2118.60	1911.00	2226.00	68.67	87.56	94.50	+/+
	2.0	1924.90	1741.00	2186.00	82.77	1679.90	1441.00	1816.00	86.97	1524.70	1381.00	1621.00	45.78	79.21	90.76	+/+
	3.0	1330.70	1101.00	1556.00	91.78	1050.40	966.00	1196.00	64.12	918.50	881.00	966.00	18.77	69.02	87.44	+/+
	5.0	739.90	621.00	856.00	65.79	585.20	571.00	621.00	9.44	542.20	516.00	566.00	10.91	73.28	92.65	+/+
	10.0	428.50	341.00	471.00	39.70	305.30	286.00	326.00	7.83	294.90	271.00	361.00	13.07	68.82	96.59	+/+
potholes	1.5	2604.80	2361.00	2886.00	125.88	2391.80	2171.00	2506.00	78.33	2172.50	2021.00	2311.00	69.24	83.40	90.83	+/+
	2.0	2021.40	1846.00	2321.00	105.73	1816.40	1581.00	1931.00	73.48	1724.60	1606.00	1856.00	47.20	85.32	94.95	+/+
	3.0	1293.60	1271.00	1461.00	36.65	1321.00	1141.00	1451.00	75.07	1259.20	1176.00	1346.00	42.91	97.34	95.32	+/+
	5.0	1032.90	1011.00	1141.00	27.86	957.40	876.00	1056.00	34.70	911.90	856.00	951.00	23.20	88.29	95.25	+/+
	10.0	951.30	741.00	1026.00	94.23	684.40	631.00	771.00	32.27	696.90	641.00	796.00	29.18	73.26	101.83	+/=
jari-huge	1.5	2703.70	2511.00	2916.00	69.50	2331.40	1831.00	2391.00	77.58	2392.80	2331.00	2461.00	26.66	88.50	102.63	+/−
	2.0	2022.90	1841.00	2281.00	135.65	1776.40	1711.00	1841.00	27.51	1855.70	1826.00	1896.00	21.34	91.73	104.46	+/−
	3.0	1370.00	1256.00	1521.00	93.48	1194.60	1136.00	1236.00	36.20	1190.20	1146.00	1251.00	30.61	86.88	99.63	+/=
	5.0	1254.70	1166.00	1331.00	70.15	1055.10	986.00	1101.00	28.51	1079.60	1041.00	1101.00	15.05	86.04	102.32	+/−
	10.0	1168.50	1136.00	1201.00	14.85	952.90	896.00	1001.00	22.56	973.30	926.00	1086.00	39.80	83.29	102.14	+/−
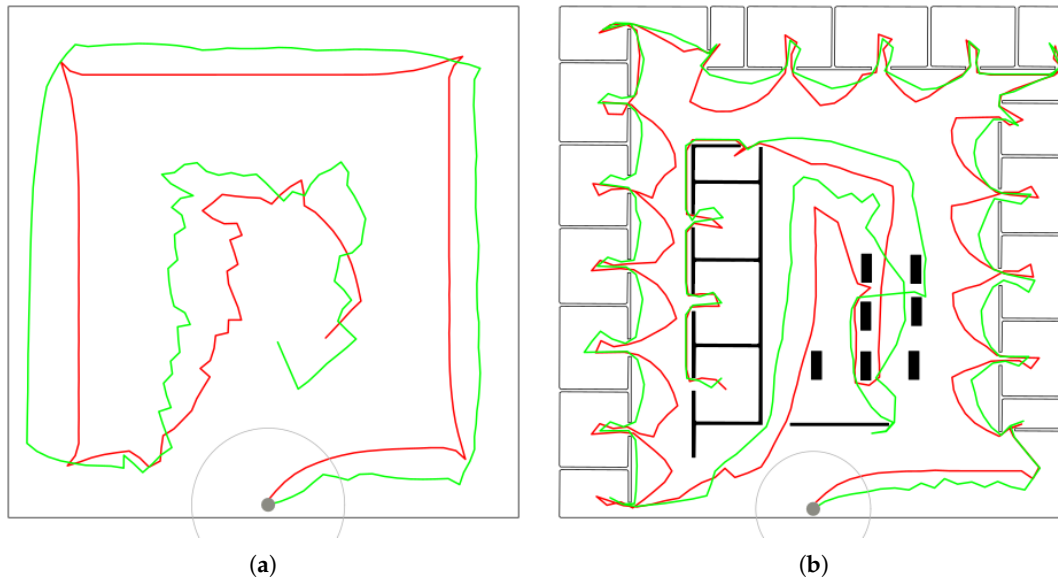
Figure 9. The best results found by EA (red) and FHP (green) on (**a**) *empty* with $\rho = 3.0$ m and (**b**) *jari-huge* with $\rho = 2.0$ m.

The situation is similar for *potholes*, where greedy is outperformed by the sophisticated approaches, although the difference is not as big as for *empty*. In addition, EA gives better results than FHP in all cases except one.

Finally, EA performs better by approx. 8–15% than greedy for *jari-huge*. On the other hand, it is slightly (up to 5%) outperformed by FHP due to the same reason that it is better than FHP for *empty*. Here, directing the robot far from obstacles when going between neighboring rooms is contra-productive (see in Figure 9b). Conversely, EA is more effective in the empty area in the middle.

The results for UMARI are presented in Table 2. The meaning of the symbols is the same as in Table 1 except R_{gr}, which expresses a ratio of the average T_{exp} obtained with UMARI to the average T_{exp} obtained with greedy. Similarly, R_{EA} is a ratio of average T_{exp} values of UMARI and EA. It can be seen that the performance of UMARI is the worst in all cases even in comparison to greedy.

Table 2. Experimental results for UMARI.

Map	ρ	Greedy				R_{gr} %	R_{EA} %
		avg	min	max	stdev		
empty	1.5	2766.21	2506.00	3066.00	132.46	114.32	130.57
	2.0	2196.61	1971.00	2431.00	112.40	114.12	144.07
	3.0	1596.71	1406.00	1816.00	101.32	119.99	173.84
	5.0	1290.15	1136.00	1446.00	77.96	174.37	237.95
potholes	1.5	2711.93	1966.00	3026.00	268.21	104.11	124.83
	2.0	2198.11	1451.00	2441.00	194.48	108.74	127.46
	3.0	1495.38	436.00	2026.00	402.12	115.60	118.76
	5.0	1429.27	471.00	1686.00	189.93	138.37	156.73
jari-huge	1.5	2845.89	2651.00	3061.00	91.46	105.26	118.94
	2.0	2447.43	2191.00	2651.00	102.75	120.99	131.89
	3.0	1738.65	1571.00	1866.00	65.00	126.91	146.08
	5.0	1596.52	1511.00	1661.00	41.87	127.24	147.88

An indirect comparison can also be done with recent (deep) learning-based strategies, as the authors of these strategies present a comparison to the greedy approach. Chen et al. [26] show that the greedy approach outperforms imitation learning, reinforcement techniques as well as Curiosity-based Exploration [70] when localization error is below 3% (which is the case we are focused on). On the other hand, their strategy performs better than `greedy` when localization error increases. Zhu et al. [25] present the evaluation of their strategy based on Reinforcement Learning supervised Bayesian Optimization on ten office plans (similar to the *jari-huge* map but smaller). Their method is worse than greedy in four cases (by 41% in one case), while better in the other cases (by up to 32%). Table 1 shows that `EA`, on the other hand, performs better than `greedy`on *jari-huge* in all cases by 8–17%.

5.2. Time Complexity

A set of experiments was performed to evaluate time complexity of the proposed EA algorithm on a workstation with the Intel® Core™ i7-3770 CPU (Intel Corporation, Santa Clara, CA, USA) at 3.4 GHz running Sabayon Linux with the kernel 3.19.0. Fifteen trials of full exploration were run in the most complex setup: the *jari-huge* map and the visibility range $\rho = 1.5$ m, while the other parameters were set to the same values as in the previous case. Figure 10a shows how the number of candidates (NoC) and computational time (T) of EA change during a typical trial. It can be seen that approximately 500 runs of EA were executed during the trial and that NoC grows as the robot explores new areas at the beginning, while it starts to decrease in the middle of the process. The curve of computational time follows the one of NoC but not so exactly as one would expect. This is caused by the fact that the complexity of the GTSPC problem does not only depend on NoC, but also on the number of clusters and distribution of candidates over them. The number of clusters varied up to 22 in this particular case.
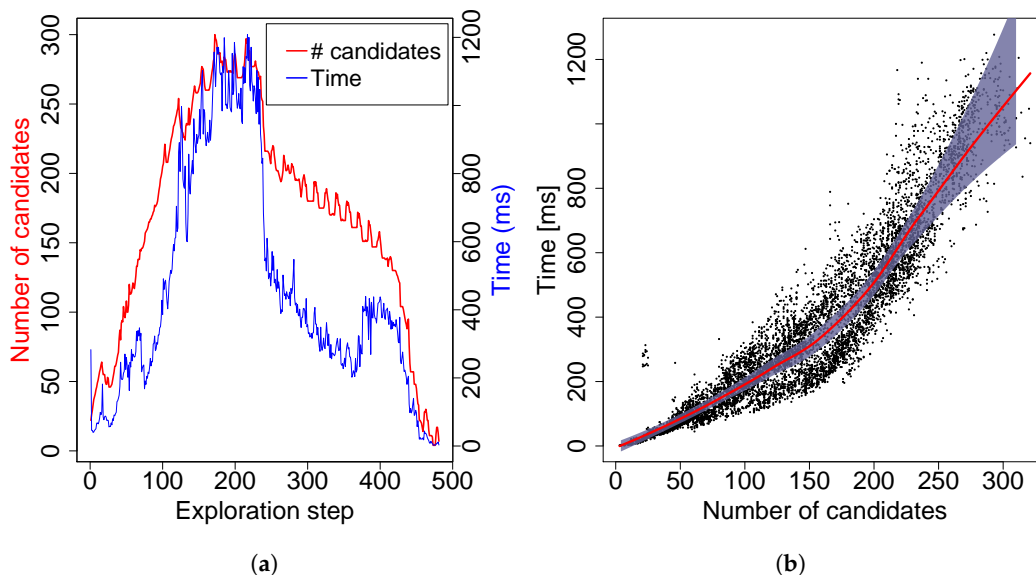

(**a**)   (**b**)

Figure 10. (**a**) evolution of computational time of EA and the number of goal candidates during a typical run on *jari-huge* with $\rho = 1.5$ m; (**b**) dependency of computational time of EA on a number of candidates. The dots represent measurements from 15 runs on *jari-huge* with $\rho = 1.5$ m. The red curve is a smoothed computational time averaged over the number of candidates, while the grey area represents the confidence interval computed for mean on a 95% confidence level.

Data from all 15 runs containing 7156 executions of EA are shown in Figure 10b in the form of dependency of T on NoC together with the averaged times for particular NoCs and the confidence interval

computed for the means on a 95% confidence level. In fact, the averages and the confidence intervals are too noisy, so local polynomial fitting is applied to make the curves smooth. The maximal number of candidates was 321, while computational time does not exceed 1.3 s. This qualifies the method to be deployed in real time.

5.3. Real Deployment

The whole exploration framework was deployed on a real robot in the SyRoTek system [71] developed at Czech Technical University to demonstrate the applicability of the proposed solution with real hardware. SyRoTek is a platform for e-learning and distant experimentation in robotics and related areas consisting of thirteen robots equipped with standard robot sensors (laser range-finders, sonars, odometry, etc.). The SyRoTek robot is called S1R and its body consists of the main chassis and an optional sensor module [72]. The robot is based on a differential drive with the maximal velocity designed to 0.35 m/s and operating time 8 h. The on-board computer (OBC) is the Gumstix Overo Fire module with the ARM Cortex-A8 OMAP3530 processor unit (ARM, Cambridge, UK) operating at 600 MHz and running the Linux kernel. The connection with the control computer is provided by the integrated WiFi module of the Overo board. The robots operate in the Arena of size 3.5×3.8 m and are fully programmable and remotely controlled. A HOKUYO URG-04LX laser range finder (Hokuyo Automatic Co., Ltd., Osaka, Japan) with a sensing view $240°$ and a range limited to 0.5 m was used as a main sensor for the experiment. Resolution of the occupancy grids was set to 2 cm. A Smooth Nearness Diagram (SND) algorithm [45] was used to control the robot motion and to avoid obstacles. Robot position was taken from the localization system provided by SyRoTek, which is capable of continuous and errorless operation with the precision of ± 1 cm in robot position and $\pm 3°$ in robot orientation. Figure 11 shows several phases of the exploration with the EA as the goal selection strategy.

(**a**)	(**b**)	(**c**)	(**d**)

Figure 11. Experiment with a real robot in the SyRoTek system. (**a**) the robot during an experiment; (**b**,**c**) phases of the exploration process; (**d**) the final map and the performed path.

6. Conclusions and Future Work

This article presents a new approach to the goal selection task solved within the robotic exploration of an unknown environment. While state-of-the-art approaches generate a relatively small set of goal candidates from which a next goal is chosen based on some evaluation function, our approach integrates goal candidates generation with a selection of the next goal, which leads to a solution of the constrained Generalized Traveling Salesman Problem. The proposed approach thus allows higher flexibility in the planning of the next robot actions.

The overall exploration framework including the proposed goal selection strategy is evaluated in a simulation environment on three different maps and compared with three state-of-the-art techniques. The results show that the proposed method can compute results for robotic problems of a standard size

in less than 1300 ms, which is sufficient for real-time usage. It statistically significantly outperforms the other three strategies in empty environments and areas with low density of obstacles: exploration times are by more than 30% lower than for the widely used `greedy` approach and up to 12.5% lower than for `FHP` in some cases. On the other hand, the proposed method is worse than `FHP` in narrow corridors by up to 4.5%, but still better than `greedy` by more than 10% on average. `UMARI` is even worse. In general, the method exhibits the best overall performance. Thus, our approach is a good choice when the type of the environment to be explored is not known in advance.

In future work, we want to focus on the generation of goal candidates. A density of generated candidates can be controlled according to several criteria: distance to the robot, distance to obstacles, a shape of the environment in the vicinity of a candidate, topology of the environment, experience from previous runs of GTSPC, etc. Generation of candidates especially at places where it is interesting can influence both the quality of a GTSP solution as well as computational complexity of GTSPC. We also want to extend the method for multiple robots.

The evolutionary algorithm itself can be optimized as well. In the current implementation, when evaluating a particular priority list, it is first translated to the corresponding tour, which is then locally improved by 1-opt and 2-opt heuristics. However, the optimized tour is only used to assess the quality of the priority list. The final order of goals within the tour is not stored for later use. Experience from the field of memetic algorithms suggests that it might be beneficial for the efficiency of the evolutionary process to keep the locally optimized solution instead of the original one. Thus, we will investigate possibilities to keep the optimized tour along with the priority list in the individual and reuse it within the crossover operator.

Author Contributions: M.K. developed the idea, designed the exploration framework, and performed the experiments. J.K. developed the IREANN algorithm, while L.P. supervised the work.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Qi, J.; Song, D.; Shang, H.; Wang, N.; Hua, C.; Wu, C.; Qi, X.; Han, J. Search and Rescue Rotary-Wing UAV and Its Application to the Lushan Ms 7.0 Earthquake. *J. Field Robot.* **2016**, *33*, 290–321. [CrossRef]
2. Nedjati, A.; Izbirak, G.; Vizvari, B.; Arkat, J. Complete Coverage Path Planning for a Multi-UAV Response System in Post-Earthquake Assessment. *Robotics* **2016**, *5*, 26. [CrossRef]
3. Bagosi, T.; Hindriks, K.V.; Neerincx, M.A. Ontological reasoning for human-robot teaming in search and rescue missions. In Proceedings of the 2016 11th ACM/IEEE International Conference on Human-Robot Interaction (HRI), Christchurch, New Zealand, 7–10 March 2016; pp. 595–596.
4. Trevelyan, J.; Hamel, W.R.; Kang, S.C. Robotics in Hazardous Applications. In *Springer Handbook of Robotics*; Siciliano, B., Khatib, O. Eds.; Springer International Publishing: Cham, Switzerland, 2016; pp. 1521–1548.
5. Montes, H.; Mena, L.; Fernández, R.; Sarria, J.; Armada, M. Inspection platform for applications in humanitarian demining. In *Assistive Robotics Proceedings of the 18th International Conference on CLAWAR 2015*; World Scientific: HangZhou, China, 2015; pp. 446–453.
6. Murphy, R.R.; Tadokoro, S.; Kleiner, A., Disaster Robotics. In *Springer Handbook of Robotics*; Siciliano, B., Khatib, O. Eds.; Springer International Publishing: Cham, Switzerland, 2016; pp. 1577–1604.

7.　　Leonard, J.J.; Bahr, A. Autonomous Underwater Vehicle Navigation. In *Springer Handbook of Ocean Engineering*; Dhanak, M.R., Xiros, N.I. Eds.; Springer International Publishing: Cham, Switzerland, 2016; pp. 341–358.

8.　　Choi, S.; Kim, E.; Oh, S. Real-time navigation in crowded dynamic environments using Gaussian process motion control. 2014 IEEE International Conference on Robotics and Automation (ICRA), Hong Kong, China, 31 May– 7 June 2014; pp. 3221–3226.

9.　　Kim, E.; Choi, S.; Oh, S. Structured Kernel Subspace Learning for Autonomous Robot Navigation. *Sensors* **2018**, *18*, 582. [CrossRef] [PubMed]

10.　Yamauchi, B. A frontier-based approach for autonomous exploration. In Proceedings of the 1997 IEEE International Symposium on Computational Intelligence in Robotics and Automation: Towards New Computational Principles for Robotics, Monterey, CA, USA, 10–11 July 1997; pp. 146–151.

11.　Koenig, S.; Tovey, C.; Halliburton, W. Greedy mapping of terrain. *Proc. IEEE Int. Conf. Robotics Autom.* **2001**, *4*, 3594–3599.

12.　Koenig, S. Improved analysis of greedy mapping. In Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, Las Vegas, NV, USA, 27–31 October 2003; pp. 3251–3257.

13.　Mei, Y.; hsiang Lu, Y.; Lee, C.S.G.; Hu, Y.C. Energy-efficient mobile robot exploration. In Proceedings 2006 IEEE International Conference on Robotics and Automation, Orlando, FL, USA, 15–19 May 2006.

14.　Gonzalez-Banos, H.H.; Latombe, J.C. Navigation Strategies for Exploring Indoor Environments. *Int. J. Robot. Res.* **2002**, *21*, 829–848. [CrossRef]

15.　Stachniss, C.; Grisetti, G.; Burgard, W. Information Gain-based Exploration Using Rao-Blackwellized Particle Filters. In Proceedings of the Robotics: Science and Systems, Cambridge, MA, USA, 8–11 June 2005.

16.　Ström, D.P.; Bogoslavskyi, I.; Stachniss, C. Robust exploration and homing for autonomous robots. *Robot. Auton. Syst.* **2017**, *90*, 125–135. [CrossRef]

17.　Amigoni, F.; Caglioti, V. An information-based exploration strategy for environment mapping with mobile robots. *Robot. Auton. Syst.* **2010**, *58*, 684–699. [CrossRef]

18.　Makarenko, A.A.; Williams, S.B.; Bourgault, F.; Durrant-Whyte, H.F. An experiment in integrated exploration. In Proceedings 2002 IEEE/RSJ International Conference on Intelligent Robots and Systems, Lausanne, Switzerland, 30 September–4 October 2002; pp. 534–539.

19.　Basilico, N.; Amigoni, F. Exploration Strategies based on Multi-Criteria Decision Making for an Autonomous Mobile Robot. In Proceedings of the 4th European Conference on Mobile Robots, ECMR'09, Mlini/Dubrovnik, Croatia, 23–25 September 2009; pp. 259–264.

20.　Basilico, N.; Amigoni, F. Exploration strategies based on multi-criteria decision making for searching environments in rescue operations. *Auton. Robot.* **2011**, *31*, 401–417. [CrossRef]

21.　Girdhar, Y.; Dudek, G. Modeling Curiosity in a Mobile Robot for Long-term Autonomous Exploration and Monitoring. *Auton. Robot.* **2016**, *40*, 1267–1278. [CrossRef]

22.　Umari, H.; Mukhopadhyay, S. Autonomous robotic exploration based on multiple rapidly-exploring randomized trees. 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Vancouver, BC, Canada, 24–28 September 2017; pp. 1396–1402.

23.　Gao, W.; Booker, M.; Adiwahono, A.H.; Yuan, M.; Wang, J.; Yun, Y.W. An improved Frontier-Based Approach for Autonomous Exploration. In Proceedings of the 15th International Conference on Control, Automation, Robotics and Vision (ICARCV), Singapore, Singapore, 18–21 November 2018; pp. 292–297.

24.　Tovar, B.; Muñoz-Gómez, L.; Murrieta-Cid, R.; Alencastre-Miranda, M.; Monroy, R.; Hutchinson, S. Planning exploration strategies for simultaneous localization and mapping. *Robot. Auton. Syst.* **2006**, *54*, 314–331. [CrossRef]

25.　Zhu, D.; Li, T.; Ho, D.; Wang, C.; Meng, M.Q. Deep Reinforcement Learning Supervised Autonomous Exploration in Office Environments. In Proceedings of the 2018 IEEE International Conference on Robotics and Automation (ICRA), Brisbane, Australia, 21–25 May 2018; pp 7548–7555.

26.　Chen, T.; Gupta, S.; Gupta, A. Learning Exploration Policies for Navigation. In Proceedings of the International Conference on Learning Representations, New Orleans, LO, USA, 6–9 May 2019.

27. Kulich, M.; Faigl, J.; Preucil, L. On distance utility in the exploration task. In Proceedings of the Robotics and Automation (ICRA), Shanghai, China, 9–13 May 2011; pp. 4455–4460.

28. Oßwald, S.; Bennewitz, M.; Burgard, W.; Stachniss, C. Speeding-Up Robot Exploration by Exploiting Background Information. *IEEE Robot. Automat. Lett.* **2016**, *1*, 716–723.

29. Faigl, J.; Kulich, M. On determination of goal candidates in frontier-based multi-robot exploration. In Proceedings of the 2013 European Conference on Mobile Robots (ECMR), Barcelona, Spain, 25–27 September 2013; pp. 210–215.

30. Faigl, J.; Vaněk, P.; Kulich, M. Self-organizing map for determination of goal candidates in mobile robot exploration. In Proceedings of the 22th European Symposium on Artificial Neural Networks, ESANN 2014, Bruges, Belgium, 23–25 April 2014.

31. Andre, T.; Bettstetter, C. Assessing the Value of Coordination in Mobile Robot Exploration using a Discrete-Time Markov Process. Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Tokyo, Japan, 3–7 November 2013.

32. Burgard, W.; Moors, M.; Stachniss, C.; Schneider, F.E. Coordinated multi-robot exploration. *IEEE Trans. Robot.* **2005**, *21*, 376–386. [CrossRef]

33. Ravankar, A.; Ravankar, A.A.; Kobayashi, Y.; Emaru, T. On a bio-inspired hybrid pheromone signalling for efficient map exploration of multiple mobile service robots. *Artif. Life Robot.* **2016**, *21*, 221–231. [CrossRef]

34. Faigl, J.; Kulich, M.; Preucil, L. Goal assignment using distance cost in multi-robot exploration. In Proceedings of the 2012 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Vilamoura, Algarve Portugal, 7–12 October 2012; pp. 3741–3746.

35. Yamauchi, B.; Schultz, A.; Adams, W. Mobile robot exploration and map-building with continuous localization. In Proceedings of the 1998 IEEE International Conference on Robotics and Automation, Leuven, Belgium, 20–22 May 1998; pp. 3715–3720.

36. Lidoris, G. *State Estimation, Planning, and Behavior Selection Under Uncertainty for Autonomous Robotic Exploration in Dynamic Environments*; Kassel University Press GmbH: Kassel, Germany, 2011.

37. Papachristos, C.; Khattak, S.; Alexis, K. Uncertainty-aware receding horizon exploration and mapping using aerial robots. In Proceedings of the 2017 IEEE International Conference on Robotics and Automation (ICRA), Marina Bay Sands, NE, USA, 29 May–3 June 2017; pp. 4568–4575.

38. Selin, M.; Tiger, M.; Duberg, D.; Heintz, F.; Jensfelt, P. Efficient Autonomous Exploration Planning of Large Scale 3D-Environments. *IEEE Robot. Autom. Lett.* **2019**, *4*, 1699–1706. [CrossRef]

39. Delmerico, J.; Mueggler, E.; Nitsch, J.; Scaramuzza, D. Active autonomous aerial exploration for ground robot path planning. *IEEE Robot. Autom. Lett.* **2017**, *2*, 664–671. [CrossRef]

40. Bircher, A.; Kamel, M.; Alexis, K.; Oleynikova, H.; Siegwart, R. Receding horizon path planning for 3D exploration and surface inspection. *Auton. Robot.* **2018**, *42*, 291–306. [CrossRef]

41. Amigoni, F.; Li, A.Q.; Holz, D. Evaluating the Impact of Perception and Decision Timing on Autonomous Robotic Exploration. In Proceedings of the 2013 European Conference on Mobile Robots, Barcelona, Spain, 25–27 September 2013; pp. 68–73.

42. Newman, P.M.; Bosse, M.; Leonard, J.J. Autonomous Feature-based Exploration. In Proceedings of the 2003 IEEE International Conference on Robotics and Automation,Taipei, Taiwan, 14–19 September 2003.

43. Elfes, A. Using occupancy grids for mobile robot perception and navigation. *Computer* **1989**, *22*, 46–57. [CrossRef]

44. Faigl, J.; Kulich, M. On Benchmarking of Frontier-Based Multi-Robot Exploration Strategies. In Proceedings of the European Conference on Mobile Robots, Lincoln, UK, 2–4 September 2015.

45. Durham, J.W.; Bullo, F. Smooth Nearness-Diagram Navigation. In Proceedings of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS), Nice, France, 22–26 September 2008; pp. 690–695.

46. Mitchell, J.S.B. Approximating Watchman Routes. In Proceedings of the Twenty-Fourth Annual ACM-SIAM Symposium on Discrete Algorithms, New Orleans, LO, USA, 6–8 January 2013; pp. 844–855.

47. Chin, W.; Ntafos, S. Optimum watchman routes. *Inf. Process. Lett.* **1988**, *28*, 39–44. [CrossRef]

48. Reddy, P.R.; Amarnadh, V.; Bhaskar, M. Evaluation of stopping criterion in contour tracing algorithms. *Int. J. Comp. Sci. Inf. Technol.* **2012**, *3*, 3888–3894.

49. Cormen, T.H.; Leiserson, C.E.; Rivest, R.L.; Stein, C. *Introduction to Algorithms, Third Edition*; 3rd ed.; The MIT Press: Cambridge, MA, USA, 2009.

50. Johnson, D.B. Efficient Algorithms for Shortest Paths in Sparse Networks. *J. ACM* **1977**, *24*, 1–13. [CrossRef]

51. Dimitrijević, V.; Šarić, Z. An Efficient Transformation of the Generalized Traveling Salesman Problem into the Traveling Salesman Problem on Digraphs. *Inf. Sci.* **1997**, *102*, 105–110. [CrossRef]

52. Ben-arieh, D.; Gutin, G.; Penn, M.; Yeo, A.; Zverovitch, A. Transformations of generalized ATSP into ATSP. *Oper. Res. Lett.* **2003**, *31*, 357–365. [CrossRef]

53. Laporte, G.; Nobert, Y. Generalized traveling salesman problem through n-sets of nodes-An integer programming approach. *INFOR* **1983**, *21*, 61–75.

54. Fischetti, M.; Salazar Gonzalez, J.J.; Toth, P. A branch-and-cut algorithm for the symmetric generalized traveling salesman problem. *Oper. Res.* **1997**, *45*, 378–394. [CrossRef]

55. Kara, I.; Guden, H.; Koc, O.N. New Formulations for the Generalized Traveling Salesman Problem. In Proceedings of the 6th International Conference on Applied Mathematics, Simulation, Modelling, Athens, Greece, 7–9 March 2012; World Scientific and Engineering Academy and Society (WSEAS): Stevens Point, WI, USA, 2012; pp. 60–65.

56. Ardalan, Z.; Karimi, S.; Poursabzi, O.; Naderi, B. A novel imperialist competitive algorithm for generalized traveling salesman problems. *Appl. Soft Comput.* **2015**, *26*, 546–555. [CrossRef]

57. El Krari, M.; Ahiod, B.; El Benani, B. Using Cluster Barycenters for the Generalized Traveling Salesman Problem. In Proceedings of the 16th International Conference on Intelligent Systems Design and Applications (ISDA 2016), Porto, Portugal, 16–18 December 2016; Madureira, A.M., Abraham, A., Gamboa, D., Novais, P., Eds.; Springer International Publishing: Cham, Switzerland, 2017; pp. 135–143.

58. Karapetyan, D.; Gutin, G. Efficient local search algorithms for known and new neighborhoods for the generalized traveling salesman problem. *Eur. J. Oper. Res.* **2012**, *219*, 234–251. [CrossRef]

59. Snyder, L.V.; Daskin, M.S. A random-key genetic algorithm for the generalized traveling salesman problem. *Eur. J. Oper. Res.* **2006**, *174*, 38–53. [CrossRef]

60. Gutin, G.; Karapetyan, D. A Memetic Algorithm for the Generalized Traveling Salesman Problem. *Nat. Comp.* **2010**, *9*, 47–60. [CrossRef]

61. Mou, L.M. A novel ant colony system with double pheromones for the generalized TSP. In Proceedings of the Seventh International Conference on Natural Computation, ICNC 2011, Shanghai, China, 26–28 July 2011.

62. Pintea, C.; Pop, P.C.; Chira, C. The Generalized Traveling Salesman Problem solved with Ant Algorithms. *CoRR* **2013**, *abs/1310.2350*, 1–9.

63. Reihaneh, M.; Karapetyan, D. An Efficient Hybrid Ant Colony System for the Generalized Traveling Salesman Problem. *Algorithmic Oper. Res.* **2012**, *7*, 22–29.

64. Kubalík, J.; Snížek, M. A novel evolutionary algorithm with indirect representation and extended nearest neighbor constructive procedure for solving routing problems. In Proceedings of the 14th International Conference on Intelligent Systems Design and Applications, ISDA 2014, Okinawa, Japan, 28–30 November 2014; pp. 156–161.

65. Syswerda, G. Schedule Optimization Using Genetic Algorithms. In *Handbook of Genetic Algorithms*; Davies, L., Ed.; Van Nostrand Reinholt: New York, NY, USA, 1991; pp. 332–349.

66. Campos, F.M.; Marques, M.; Carreira, F.; Calado, J.M.F. A complete frontier-based exploration method for Pose-SLAM. In Proceedings of the 2017 IEEE International Conference on Autonomous Robot Systems and Competitions (ICARSC), Coimbra, Portugal, 26–28 April 2017; pp. 79–84.

67. Quin, P.; Paul, G.; Liu, D. Experimental Evaluation of Nearest Neighbor Exploration Approach in Field Environments. *IEEE Trans. Autom. Sci. Eng.* **2017**, *14*, 869–880. [CrossRef]

68. Zheng, G.; Zhang, L.; Zhang, H.Y.; Ding, B. Design of an Indoor Exploration and Multi-Objective Navigation System. In Proceedings of the 37th Chinese Control Conference (CCC), Wuhan, China, 25–27 July 2018; pp. 5451–5458.

69. Motion Planning Maps Dataset. Available online: http://agents.fel.cvut.cz/~faigl/planning/maps.xml. (accessed on 29 August 2016).

70. Burda, Y.; Edwards, H.; Pathak, D.; Storkey, A.; Darrell, T.; Efros, A.A. Large-Scale Study of Curiosity-Driven Learning. In Proceedings of the Seventh International Conference on Learning Representations (ICLR), New Orleans, LO, USA, 6–9 May 2019.

71. Kulich, M.; Chudoba, J.; Košnar, K.; Krajník, T.; Faigl, J.; Přeučil, L. SyRoTek—Distance Teaching of Mobile Robotics. *IEEE Trans. Educ.* **2013**, *56*, 18–23. [CrossRef]

72. Chudoba, J.; Faigl, J.; Kulich, M.; Krajník, T.; Košnar, K.; Přeučil, L. A Technical Solution of a Robotic e-Learning System in the SyRoTek Project. In Proceedings of the 3rd International Conference on Computer Supported Education, Noordwijkerhout, Netherlands, 6–8 May 2011; Verbraeck, A.; Helfert, M.; Cordeiro, J.; Shishkov, B., Eds. SciTePress: Setúbal, Portugal, 2011, pp. 412–417.