

ITC 2/50 Information Technology and Control Vol. 50 / No. 2 / 2021 pp. 357-374 DOI 10.5755/j01.itc.50.2.28234	Cloud-Based Multi-Robot Path Planning in Complex and Crowded Environment Using Fuzzy Logic and Online Learnings	
	Received 2020/12/29	Accepted after revision 2021/05/18
	 http://dx.doi.org/10.5755/j01.itc.50.2.28234	

HOW TO CITE: Zagradjanin, N., Rodic, A., Pamurar, D., Pavkovic, B. (2021). Cloud-Based Multi-Robot Path Planning in Complex and Crowded Environment Using Fuzzy Logic and Online Learning. *Information Technology and Control*, 50(2), 357-374. <https://doi.org/10.5755/j01.itc.50.2.28234>

Cloud-Based Multi-Robot Path Planning in Complex and Crowded Environment Using Fuzzy Logic and Online Learning

Novak Zagradjanin

University of Belgrade, School of Electrical Engineering, Bulevar kralja Aleksandra 73, 11120 Belgrade, Serbia

Aleksandar Rodic

University of Belgrade, School of Electrical Engineering, Bulevar kralja Aleksandra 73, 11120 Belgrade, Serbia
Mihailo Pupin Institute, Volgina 15, 11060 Belgrade, Serbia

Dragan Pamucar

Department of Logistics, Military academy, University of Defence, Pavla Jurisica Sturma 33, 11000 Belgrade, Serbia

Bojan Pavkovic

Military technical institute, Ratka Resanovica 1, 11030 Belgrade, Serbia

Corresponding author: dpamucar@gmail.com

This paper considers an autonomous cloud-based multi-robot system designed to execute highly repetitive tasks in a dynamic environment such as a modern megastore. Cloud level is intended for performing the most demanding operations in order to unload the robots that are users of cloud services in this architecture. For path planning on global level D* Lite algorithm is applied, bearing in mind its high efficiency in dynamic environments. In order to introduce smart cost map for further improvement of path planning in complex and crowded environment, implementation of fuzzy inference system and learning algorithm is proposed. The results indicate the possibility of applying a similar concept in different real-world robotics applications, in order to reduce the total paths length, as well as to minimize the risk in path planning related to the human-robot interactions.

KEYWORDS: multi-robot system, path planning, cloud technology, fuzzy logic, learning algorithm.

1. Introduction

The basic purpose of an algorithm used in the robot path planning process is to determine a valid path from the start to the goal position of the robot in its configuration space. Robot configuration space implies an approach that fully defines the location of a robot, that is, all its degrees of freedom [24], in the environment in which it moves and performs tasks. In the case of multi-robot path planning, the situation is much more complicated since the growth of the joint configuration space is exponentially proportional to the number of robots [19]. Therefore, single-robot path planning techniques cannot be applied in the case of a multi-robot system without some adaptation.

Generally speaking, there are two approaches that can be applied to algorithms designed for planning the path of a multi-robot system [36]. The first approach involves a coupled technique that treats the multi-robot system as a single entity so that the paths of all robots of that system are calculated simultaneously and coordinated in the joint configuration space. Therefore, this approach can guarantee the completeness of the solution (all paths will be calculated if a solution is possible), but its disadvantages are that it is difficult to apply to a large number of robots and usually does not provide a solution in real time. The second approach involves a decoupled technique that, within a multi-robot system, first calculates the path of each individual robot independently of the others, and then applies some method of motion coordination to avoid conflict situations. This approach is fast and usually provides a real-time solution, but its disadvantages are that it cannot guarantee the completeness of the solution, as well as that movement of multiple robots along independently calculated paths can cause conflicting situations that cannot be solved.

This paper discusses an autonomous cloud-based multi-robot system designed to perform highly repetitive tasks in a dynamic environment such as a modern megastore. In robotics, the method often used to calculate the path involves the use of an algorithm for planning rough path on global level integrated with an algorithm for planning precise path on local level [11]. In complex environments such as a megastore, algorithm for planning the path on global level defines path that do not take into account crowds and other

constraints. Therefore, a human-aware algorithm for planning takes into account the crowds (as well as other limitations not covered by this paper) on local level and defines a valid local path. Such solutions are common, but if the algorithm for planning the path on global level does not fully take into account crowds, this can reduce the efficiency of the entire system and increase the risk of the task [2]. In addition, in some situations a local planner cannot solve the problem, so it is necessary to replan the path on global level during the robot motion. These situations are specially considered in this paper because they can significantly slow the robot and cause at the same time increasing the total paths length.

In order to construct a crowd-sensitive global planner, knowledge about the crowds global changes or behavior is required. This is not always feasible, because over time the crowds can behave differently. In some cases, crowd density has a relatively repeatable dynamic of change on daily, weekly or some other time basis. For instance, in some part of the store the crowd is likely to be larger every day in the morning, in the second one every weekend, in the third one for holidays or special events. Therefore, it is desirable that robots should be capable to predict crowd density change, whenever and wherever possible, in order to improve the efficiency of task and motion planning. To this end, the implementation of fuzzy inference system (FIS) is proposed, which should ensure that in the initial phase of planning the global planner takes into consideration the crowds whose density change is predictable. The goal is to penalize the path passing through the high-cluttered parts of the store proportionately to the crowd density.

In addition to using FIS, robots have the ability to learn online in order to reuse detection of unexpected long-lasting changes (that do not move/change for a certain period of time) in environment structures from previous iterations for more efficient task and motion planning in the next iteration. The changes that are referred to here are, for example, accidentally spilled larger quantities of goods that occupy certain parts of the store for some time, temporarily blocked or opened passages, etc. To this end, it is proposed to use the learning algorithm based on the online statistical estimation of changes in the environment.

D* Lite is applied as graph-based algorithm for planning the path on global level. In this paper, it is adapted so that it can be used to plan the path of a multi-robot system. A decoupled path planning technique is applied with robots motion coordination to avoid conflict situations.

As it is known, graph based algorithms require a cost map of the robot environment to operate on. Consequently, one of the approaches to improve path planning is to generate a smart cost map. The cost map is used to store various information about the environment that affects the robots motion. This area has not been sufficiently studied, especially in the case of crowded environment, and provides many opportunities for further research. In that context, the contribution of this paper is reflected in the original application of FIS to exploit predictable changes in crowd density (in spatio-temporal domain), in order to generate smart cost map to improve autonomous robot path planning in crowded environment. In addition, it is also proposed a learning algorithm that has the same purpose as the FIS, but exploits data of other characteristics. This learning algorithm is based on the M-out-of-N detector, which is most commonly used in radar technology. In this case, the original application of this technique is presented to detect the presence or absence of an obstacle in a cell of smart cost map.

In summary, the common goal of applying fuzzy logic and online learning is to generate a smart cost map that is used for defining path in the initial planning phase so that they are similar to the real ones to a feasible extent. This will be an attempt to reduce the number of paths corrections during robots motion and thus to improve system efficiency. Replanning the paths increases the execution time of the task to a lesser or larger extent, often implies robots delays, degrades path quality as well as overall performance of the system. As a result, it can be expected that the total length of the paths will be reduced (the sum of the path length of all robots in all iterations in one cycle), making them on average “smoother” in terms of a smaller number of sharp changes in direction (easier to follow by robots). The aim is also to reduce the risk in path planning, which refers to the movement of humans and robots in the same environment, that is, to ensure an amount of safe space between robots and crowds. That improves the overall performance

of the system. The same idea can be implemented in industrial and other similar environments.

2. Related Work

Graph-based techniques are used with both algorithms - algorithms for planning the path for single-robot and algorithms for planning the path for multi-robot system. M* algorithm [35] is based on the search for path in joint configuration space with dimensionality gradation depending on the need for motion coordination. In [4] A* algorithm for multi-robot path planning is proposed using decoupled approach and priorities. A lattice variant of A* algorithm for planning the path for multi-robot system is proposed in [8]. D* is an incremental algorithm widely used for path planning in a completely unknown or partially known environment, both for single robot and for multi-robot systems [6]. A multi-agent version of D* Lite, which is a variant of D* [18], is proposed in [1]. Another approach to apply D* Lite for planning the path for multi-robot system with implemented concept of parallel processing is proposed in [30]. Here, during the planning, other robots, except the origin, are considered as obstacles.

In addition to the above mentioned, in recent years one of the main directions of research in robotics is application of different learning techniques to robot path planning, in order to improve its efficiency. In [29] a mission coordination architecture is proposed, which executes planning the path through check-points and assigning priorities, based on statistical estimation of changes in the environment with purpose to improve the replanning process. The robot motion planner proposed in [20, 25, 43] uses a formed database that stores segments of the local paths depending on the geometric shape of the obstacles, in order to form a global path based on them depending on the specific situation. Another learning based method to reuse information from previous motion plans, the environment, and types of the obstacles is presented in [15]. Path planning using experience is also proposed in [14, 32, 33], where so-called experience graphs are introduced, which in fact represent a path network from previous iterations. This approach is very useful in the environment which possesses a significant amount of underlying structure that is under-utilized in a classic path planning and

mobile manipulation. If there is no experience planning is done from the beginning, of course. In [5] and [9], the concept of learning based on the experience of trajectories in high-dimensional space is described, and this knowledge is used by robots for planning in the next iterations. An interesting approach of using learned maneuvers from previous situations that required avoiding collisions with obstacles in order to improve planning in subsequent iterations is considered in [34]. In [28] the concept of robot path planning using external preferences and suggestions by the user is addressed.

When using graph based algorithms, researchers often focus on smart cost map to improve path planning process. In [12] it is described how the clever design of a grid based cost map contributes to the robustness of the system in a real application. This approach used so-called combined cost map. It incorporates a cost map derived from known static obstacles, as well as a cost map based on the structure of roads in the environment and other available information about terrain that can affect the robots motion. Dynamic obstacles in this cost map are presented as a high cost region around the obstacle to ensure a safe distance. In [26] a cost map based on probability is presented, which takes into account the uncertainty of terrain characteristics for each grid cell in the map. The final cost map allows the calculation of the optimal path between the two points in the environment, as well as the distribution of likely paths between the points. To construct a crowd-sensitive path planner based on smart cost map is a special challenge. In [40] it is presented approach based on learning the cost map that best defines previously observed motion and trajectories of pedestrians. This approach uses a cost map at the beginning of the process that ignores predictions of future pedestrian locations. Then, in such a map, robot paths are planned, at the same time pedestrian trajectories are simulated (on the basis of previous observations), and based on that, new costs are defined in a cost map proportional to the hindrance probability at each cell. The paper [37] presents similar cloud-based path planning concept using D* Lite algorithm, but there the implementation of Multi-criteria decision making (MCDM) using Full consistency method (FUCOM) for improvement the efficiency of path planning is proposed. The application of MCDM using FUCOM provides an adaptive

approach to path planning, in terms of optimizing the global cost map taking into account all factors affecting the robots motion in the environment and having in mind a mission specificity that requires the management of risks arising from different sources. In [2] two algorithms (CUSUM-A* and Risk-A*) that learn a cost map in crowded environments and plan from it are described. These algorithms continuously upgrade crowd models based on current observations. The CUSUM-A* algorithm uses in the path planning process prior knowledge of changes in the crowds in the spatio-temporal domain, while Risk-A* focuses on optimizing path costs in terms of human-robot interactions.

Based on the analysis of the literature, it can be concluded that the smart cost map is a good approach to improve robots' path planning in general, even in environments they share with humans. Despite the unpredictability of human movement, it can often be stated that in public objects, especially in stores and similar environments, crowd density has a relatively repeatable dynamic of change (in spatio-temporal domain), that can be used to generate smart cost maps to improve robot path planning. The existing literature does not sufficiently consider this issue. The same can be said for the application of learning algorithms to improve robot path planning in crowded environments. Keeping that in mind, the tendency of this paper is to contribute to overcoming mentioned research gap.

3. System Overview

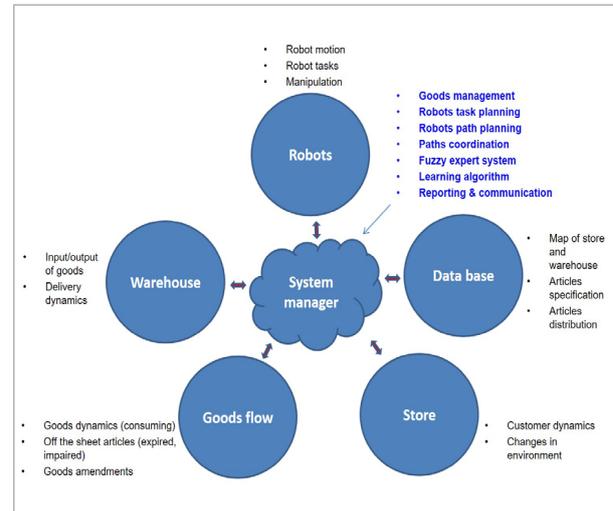
Multi-robot systems have many applications and the field of application is constantly expanding. For the purpose of performing mission successfully, the allocation of tasks among the robots in multi-robot system, path planning and motion coordination must be primarily defined. With the aim of a wider application of a group of robots, they need to be as simple as possible, energy efficient, low cost, and at the same time to perform the assigned tasks reliably. Clearly, some of these requirements are conflicting. The solution can be based on cloud technology. With the application of cloud technology, it can be achieved that the multi-robot system uses the capacities and resources of the cloud (information, memory, communication

and other) [3, 13]. In practice, this usually means that the cloud processes a large amount of data, while robots, as cloud service users, receive the necessary data and information that enables them to perform tasks [7, 16]. Let's take the modern megastore as a typical example of a complex dynamic system, where the flow of goods and money takes place and where people are present as customers and employees. In such a system, a group of robots may first be intended for smart logistics i.e. to perform tasks such as: goods delivery (delivery robots), autonomous goods handling (manipulation robots), autonomous scouting (scout robots), cleaning (cleaning service robots), info service (info robots), shopping assistance (shopping robot assistant), etc. Planning and executing the above tasks at the same time involves great resources for data processing. For these reasons, the environment of modern megastore can be considered as one of the possible scenarios where a multi-robot system based on cloud technology can be deployed. In order for this complex system to perform tasks with sufficient reliability in the cloud environment, the megastore must have an appropriate information infrastructure, that it is fully covered with a distributed network of different sensors, as well as with wireless communication between the participants of all involved processes. Any change in the environment is identified on the cloud level, processed and necessary information related to the changes is exchanged between networked clients. This approach makes possible to organize the performing of different tasks in megastore with synchronized accomplishment and to manage collaborative strategies of multi-robot system. One of the basic tasks in given model of high automated megastore is planning the path of robots in presence of static and moving obstacles. For example, the typical mission that can be put to robot in charge is to transport goods from the warehouse to the store in advance specified place, for the required time, moving along the path that is most favourable in terms of multiple criteria. This is just one of the complex tasks in the described scenario relocated from the robot level to the cloud level and as such it represents the focus of this work. For the needs of Matlab simulation of an autonomous multi-robot system designed to perform highly repetitive tasks of goods transport from the warehouse to the store and other tasks in complex dynamic environment, such as a modern megastore, a cloud-based

architecture has been designed, which consists of the following modules: Data base, Store, Goods flow, Warehouse, System manager and Robots (Figure 1).

Figure 1

Block diagram of the system architecture



The module “Data base” is intended to provide a store and warehouse map (free fields, dynamic and static obstacles, shelves with goods), as well as information on articles specification and distribution on shelves in the store and in the warehouse.

The module “Store” simulates a dynamic environment in the store in sense of appearance and movements of the buyers. The number of buyers is given as an initial parameter in the phase of initialization, while their moving is of a random nature. This module also simulates the changes in environment in the sense of blocking certain initially free cells or releasing the initially blocked transits. The initial map of the store and warehouse is shown in Figure 2.

The module “Goods flow” monitors picking of the goods from the shelves in the store and purchase of the same (with the cash register verification). This module also processes the loss of the goods due to the expiration date and/or damaging etc., updates the list and schedule of goods in the store and generates the goods amendments.

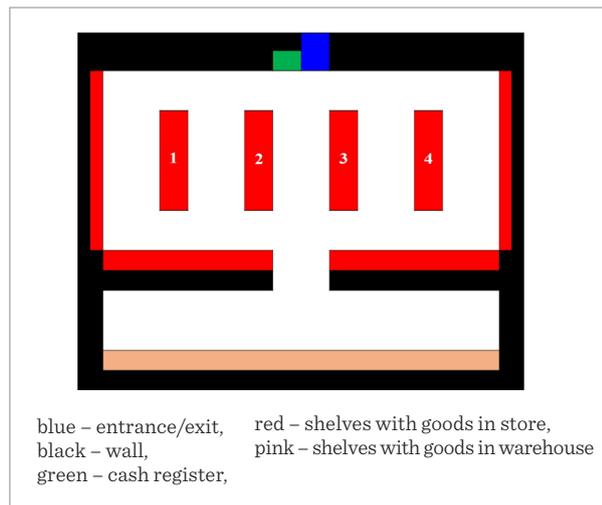
The module “Warehouse” monitors the situation in the warehouse and delivery dynamics (updates the list and schedule of warehoused goods).

The module “Robots” in a simulated environment represents robots that move along defined paths, observe the environment with their sensors, and perform assigned tasks. In this case, the system consists of three robots for the transport of goods (homogeneous system).

The module “System manager” is a cloud level. It is designed to perform the most complicated tasks, primarily related to data processing and activity coordination, which ensure the successful functioning of multi-robot system, for example: receipt of the requests for adding the goods to the shelves in the store, making the plan of transport the goods from the warehouse to the store (in accordance to the received requests for adding the goods to the shelves, the number of available robots and optimization criteria), making the plan of engaging the available robots with the start and goal positions of each robot, planning or replanning the path of robots, application of the fuzzy inference system and learning algorithm in order to improve the path planning efficiency, motion coordination, finding a solution to conflicting situations, map update with gathered data.

Figure 2

Map of the store and warehouse



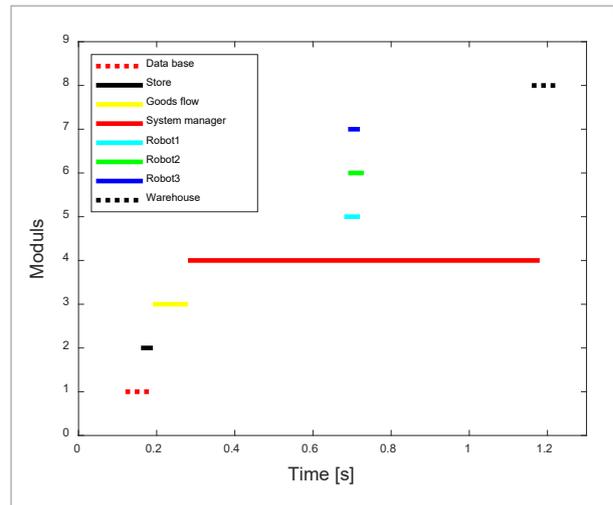
The load of each module in one iteration of a randomly selected scenario is given in Figure 3.

Based on Figure 3, it can be concluded that the system intensively uses cloud resources in the sense that “System manager” executes the most demanding

operations related to gathering and grouping of information, their processing, allocation of tasks between robots, application of the fuzzy inference system and learning algorithm, path planning and replanning, motion coordination, etc.

Figure 3

Gantt chart of operations allocation between the modules



A direct consequence of this is the unloading of robots as they are able to use cloud services.

Since the mentioned architecture is of a modular type, it can be adapted for various other applications. This may include increasing the number of robots in a system, of heterogeneous or homogeneous type.

4. Improvement of Multi-Robot Path Planning

4.1. Some Properties of D* Lite Algorithm

In order to plan the robot motion, the environment map can be presented as a 2D uniform resolution occupancy grid. In this grid, each cell is assigned a traversal cost greater than zero, which represents the difficulty of entering that cell. This grid is usually approximated as a discrete graph and some graph-based search technique can be applied for path planning. One of the techniques by which this can be performed is to define the centre of each cell as a single node in the corresponding graph. Adjacent nodes are con-

nected with the edges. Edge cost is usually adopted as some combination of the traversal cost of nodes it connects and the edge length [11]. Generally speaking, the purpose of robot path planning algorithms, including those from the A* family, is to find a path from start to goal position with minimal total cost.

The A* algorithm calculates a path assuming that the graph in which the search is performed is static, i.e. not to change. However, the real situation is usually often such that the environment in which the robot moves is only partially known or even completely unknown. In this case, path planning process is usually done assuming that parts of the environment that are unknown at that moment are free to pass, while the robot observes the environment during its motion and the map is updated on the basis of this data. In addition, changes in the environment may occur during the robot motion. In both cases, the environment map available for mission planning is dynamic and therefore the graph used for path planning also changes. A consequence of this may be that the solution generated by the A* algorithm is not optimal or even not valid. In order to avoid this, in the mentioned cases A* algorithm resets the search process (expansion of the cells) and calculates the path from the beginning (relative to the current position of the robot), without the possibility of using the search results from the previous steps. These problems are more effectively solved by so-called incremental algorithms (an extension of A* algorithm), bearing in mind that when changes in the environment map are detected, they use the results from the previous search steps to calculate a new one or to correct an existing path if necessary. D* Lite belongs to these algorithms [17].

D* Lite is a variant of the more famous D* algorithm. It is at least as efficient as the D*, and differs from it by algorithmic steps and is simpler in this respect. D* Lite works by generating an initial solution basically in a similar way as the A* algorithm. Then, if changes in the environment are detected, D* Lite performs a replanning process to check the current solution and, if necessary, to correct it or calculate the new one with the maximum possible use of search results from previous iterations. In other words, if a graph change occurs D* Lite does not reset the search (does not return that process to the beginning). This makes D* Lite significantly more efficient than the basic A* algorithm.

D* Lite unlike basic version of A* searches path from s_{goal} to s_{start} . During operation, D* Lite generates and updates the value of the following functions that characterize cell s :

- $g(s)$, the minimum cost of moving from s_{goal} to s found so far;
- $h(s)$ or heuristic value, estimates the minimum cost of moving from s to s_{start} . Using heuristic value ensures that the search tree is directed toward the most optimistic cells in terms of belonging to the optimal path from start to goal cell (this speeds up the search);
- $f(s) = g(s) + h(s)$, estimates the minimum cost of moving from s_{start} via s to s_{goal} ;
- $rhs(s) = \min_{s' \in Succ(s)} (c(s, s') + g(s'))$ or $rhs(s) = 0$ if $s = s_{goal}$, one step lookahead estimate of s and represents path cost derived from looking at the g values of its neighbors. In implementation, each cell maintains a pointer to the cell from which it derives its rhs value, so the robot should follow the pointers from its current cell to pursue an optimal path to the goal.

The pseudocode of of D* Lite is shown in Algorithm 1 [22] and its main steps will be explained below. During the path searching, D* Lite classifies cells as consistent if $g(s) = rhs(s)$, or inconsistent in other case. Cells that are inconsistent can be overconsistent (if $g(s) > rhs(s)$) or underconsistent (if $g(s) < rhs(s)$). Similar to other algorithms from the A* and D* family, D* Lite generates and updates *OPEN* list (set of inconsistent cells) to perform efficient expansion of these cells and develop a search tree. Expansion of cells (the cell becomes expanded when it is removed from *OPEN* list) is based on the *key* function as in line 8, while its value is defined as in line 1. Initializing the D* Lite algorithm involves defining the *goal* cell parameters so that it becomes inconsistent and inserted into the *OPEN* list (lines 15-17). The planner then calculates the minimum cost path by calling the *ComputePath()* function (line 19). The robot moves along the calculated path (s_{start} is also updated). If some changes in the environment occur during the motion of the robot within the range of its sensor, the algorithm updates the *rhs* parameters of all cells directly affected by the change and inserts those cells that become inconsistent in the *OPEN* list (lines 20-23). After that, it performs checking/replanning the path by recall-

ing the `ComputePath()` function. Line 18 essentially means that in a real implementation the planner ends its work when it becomes $s_{start} = s_{goal}$ (when the robot reaches the goal cell).

Algorithm 1: D* Lite algorithm (basic version)

key(s)

return $[\min(g(s), rhs(s)) + h(s_{start}, s); \min(g(s), rhs(s))];$

UpdateCell(s)

02. if s was not visited before

03. $g(s) = \infty;$

04. if $(s \neq s_{goal})$ $rhs(s) = \min_{s' \in Succ(s)} (c(s, s') + g(s'));$

05. if $(s \in OPEN)$ remove s from $OPEN;$

06. if $(g(s) \neq rhs(s))$ insert s into $OPEN$ with $key(s);$

ComputePath()

07. while $(\min_{s \in OPEN} (key(s)) < key(s_{start})$ OR $rhs(s_{start}) \neq g(s_{start})$)

08. remove cell s with the minimum key from $OPEN;$

09. if $(g(s) > rhs(s))$

10. $g(s) = rhs(s);$

11. for all $s' \in Pred(s)$ UpdateCell(s');

12. else

13. $g(s) = \infty;$

14. for all $s' \in Pred(s) \cup \{s\}$ UpdateCell(s');

Main()

15. $g(s_{start}) = rhs(s_{start}) = \infty; g(s_{goal}) = \infty;$

16. $rhs(s_{goal}) = 0; OPEN = \emptyset;$

17. insert s_{goal} into $OPEN$ with $key(s_{goal});$

18. forever

19. ComputePath();

20. wait for changes in edge costs;

21. for all directed edges (u, v) with changed edge costs

22. Update the edge cost $c(u, v);$

23. UpdateCell(u);

If the edge costs are equal to the lengths of the corresponding transitions, then D* Lite generates the path of the least cost which in this case is also the shortest path in the graph-representation of the environment.

A measure of the efficiency of algorithms that perform path planning based on graph search is the number of expanded nodes [21, 22]. By expanding a fewer cells, the algorithm comes to the solution (calculates the path) faster. Analyzing from the aspect of D* Lite algorithm, any detected change in the environment during the motion of the robot leads to calling the

function for checking the validity of the current path and to correct it if necessary. Within any particular execution of this function no one cell is expanded more than twice [23]. Multiplying the calling of this function generally causes a cumulative increase in the total number of expanded cells in planning/re-planning process.

4.2. Fuzzy Inference System

The megastore environment usually involves static structures, such as walls and shelves, whose location is apriori known and they are not considered as particular challenge for generating the path of robots. On other side, people (workers and/or buyers) that move relatively unpredictably can slow or stop the robots, increasing their travel time and total paths length. Modern technologies provide a relatively simple monitoring of statistics on the average popularity in some location depending on the time. One of the examples of statistical data processing in this regard is Google's Popular times graph. This graph shows how busy some location typically is during different times of the day and it is based on the average number of visitors over the last several weeks.

From the aspect of robots path planning in crowded and complex environment, such as megastore, changes in crowd density or the average number of buyers per unit of the time are important information, because they can affect navigation costs, as it was mentioned. If dynamic of change of crowd density in store or its parts is relatively repeatable over time (for example at a daily or weekly level), this can be used to improve the efficiency of task and motion planning for robots that operate in this environment. Having that in mind, in this paper the application of fuzzy inference system is proposed in order to regulate the change in the traversal cost of the cells in the map to follow the patterns of change in the crowd density. In our approach, traversal cost of the cells depends on the time (i.e. iteration number of the robots engagement) and the distance of the cells from the predefined part of the store. In this way, D* Lite will generate initial path through the areas that are expected to be high-cluttered only when open space or other more favorable options are not available.

Suppose that the map of store (Figure 2) is presented as a discrete grid of 16x16 cells, intended for planning the path on global level. In order to present the proposed concept for improvement of path planning in this me-

gastore scenario, our approach makes assumption that in one part of store - around the third row of shelves (Figure 2), the dynamic of change of crowd density is relatively repeatable on a daily level. From the aspect of the search algorithm, a larger number of buyers in some part of the store mean higher probability of occupancy of appropriate cells in map. Therefore, the repetitive statistics is supposed to be as follows:

- probability of occupancy of the cells nearest to the third row of shelves depending on the iteration number (part of the day) is as shown in Figure 4,

- moving away from the third row of shelves for one cell in all directions the probability of occupancy decreases by 33%.

Also, it is supposed that the time between two consecutive iterations of engagement of a multi-robot system for transport of goods is approximately constant - about 10 minutes. Working time of megastore is from 6h to 23h, so the total number of iterations of the engagement of the multi-robot system is about 100.

Figure 4

Probability of occupancy of the cells nearest to the third row of shelves depending on the time

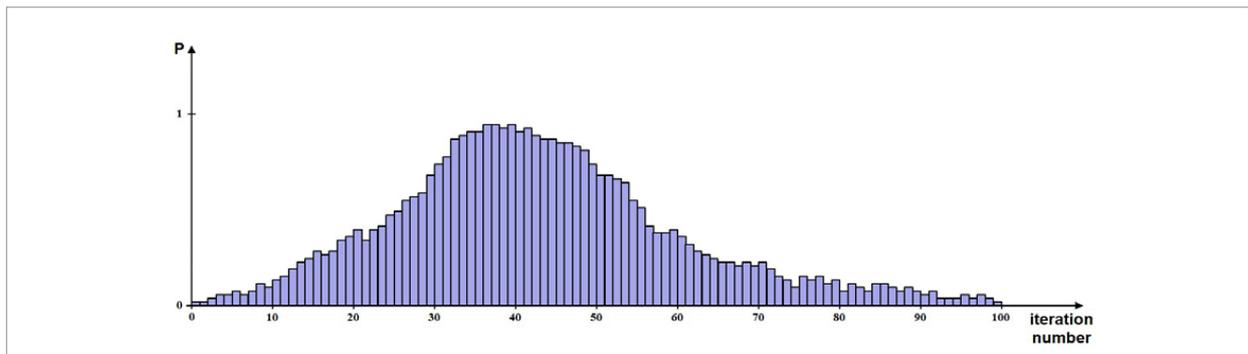
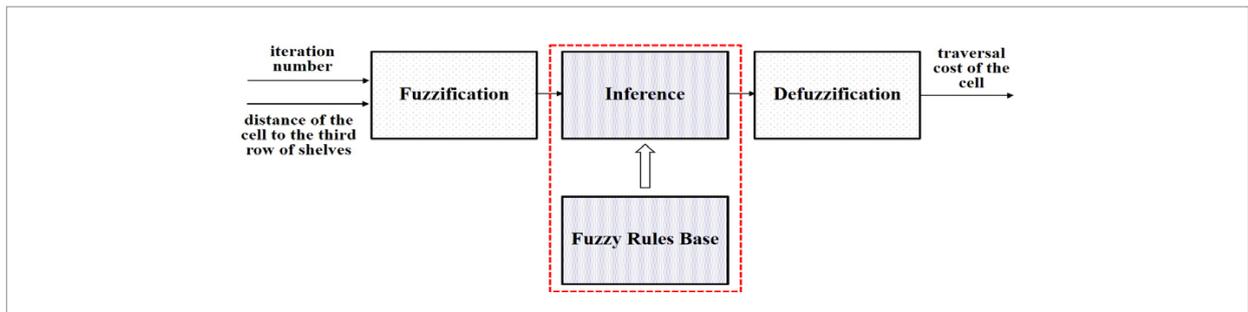


Figure 5

Block diagram of fuzzy inference system for calculating the traversal cost of the cells



Block diagram of the proposed fuzzy inference system is shown in Figure 5.

The fuzzy membership functions for the input linguistic variables, as well as the output linguistic variable are given in Figure 6.

The fuzzy rules for determining the traversal cost of cells in the map are constructed as in Table 1.

Fuzzy logic techniques are efficient in solving complex, ill-defined problems that are characterized by uncertainty of environment and fuzziness of information [36, 37]. Taking into account that disturbances and noises are common sources of uncertainties, it can be concluded that from the aspect of fuzzy implementation this system is highly resistant to noise and disturbance [38, 39, 40].

Figure 6
Fuzzy membership functions

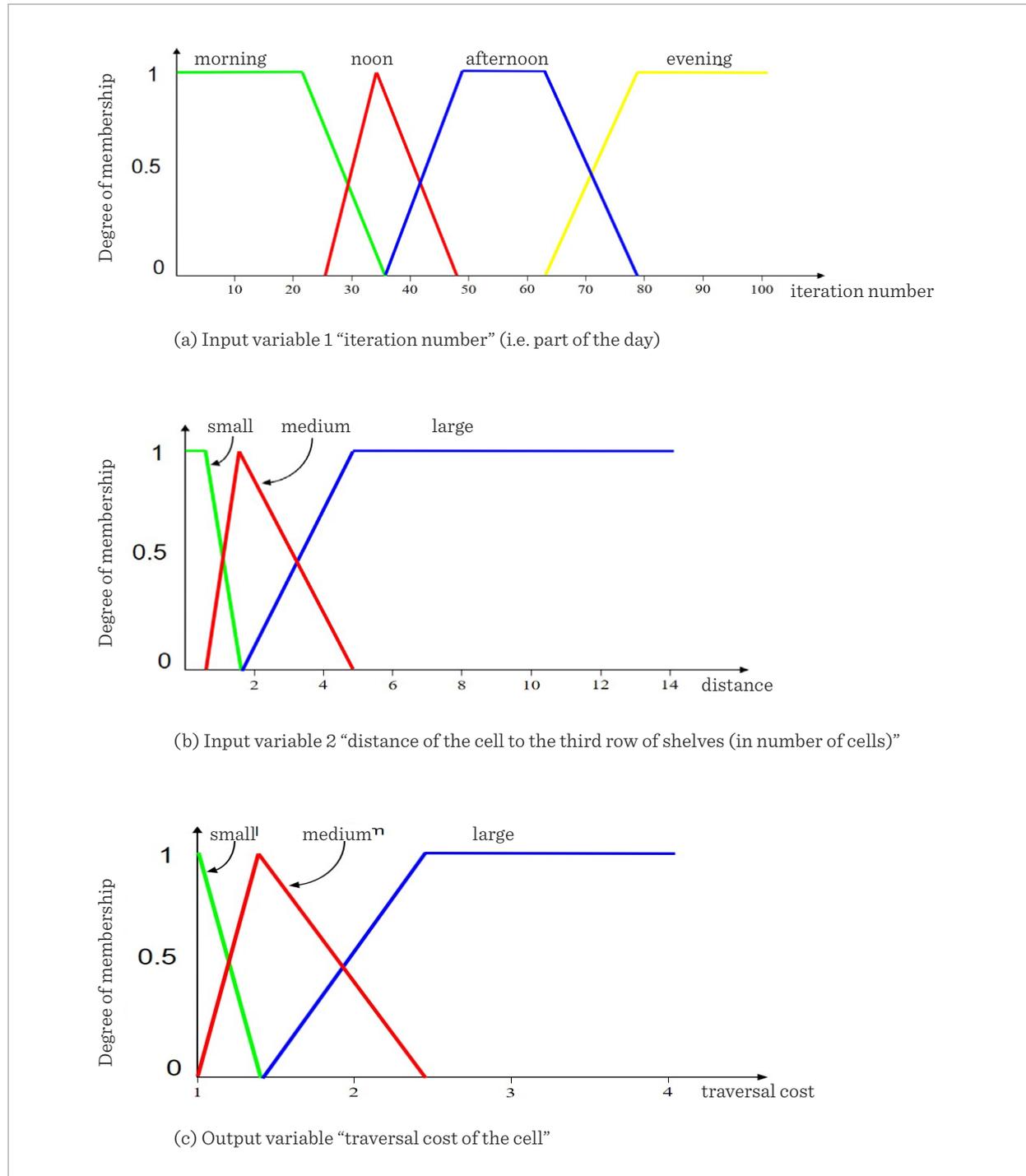


Table 1

Fuzzy rules table for calculating the traversal cost of the cells in the map

Rule number	Fuzzy input 1	Fuzzy input 2	Fuzzy output
1	morning	small distance	medium cost
2	morning	not small distance	small cost
3	noon	small distance	large cost
4	noon	medium distance	medium cost
5	noon	large distance	small cost
6	afternoon	small distance	medium cost
7	afternoon	not small distance	small cost
8	evening	-	small cost
Connection type "AND"			

4.3. Online Learning

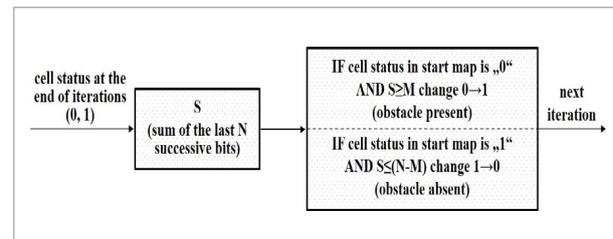
In addition to fuzzy inference system, it is proposed to integrate in considered architecture, under the introduced assumptions, an online learning algorithm based on binary moving-window or M -out-of- N detector [10]. With this learning-based approach, we will try to obtain reusing data collected with robots' sensors from previous iterations whenever possible, in order to additionally improve the performance of multi-robot system with every execution of repetitive tasks

As the robots travel and observe the environment, they update the map. The updating of the map in essence depends on the number of robots, their paths, as well as the range of sensors. In order to apply proposed learning algorithm, status of each cell of map observed by iterations of robots engagement is presented as binary time series of "0" (free cell) and "1" (blocked cell). The moving-window slides on these bits and the change of the start cell status (status according to the basic map) for the initial path generation process in the first subsequent iteration is declared when test statistics satisfies the following: there exist at least M of N bits of the opposite value in regard to the actual bit value, where N is the window length (Figure 7). Applied to specific case, this means that if in the N successive iterations an obstacle is detected M times in a cell that should be free according to the basic map, then during generation process of the initial path in the first subsequent iteration this

cell is declared as blocked. In general, the same logic is applied to cells that should be blocked according to the basic map. Of course, there will be iterations that some cells are outside of the sensors range of all robots, so at the end of these iterations statuses as in the start (basic) map are assigned to them (with the aim of statistical estimation in the learning algorithm).

Figure 7

Block diagram of learning system



This detector provides the forgetting of past iterations, which is regulated by window length N . Parameter M is adjusted according to the specific environment, depending on the period of engagement of robots and expected changes in the environment.

Suggested learning system has a simple structure and therefore it is very easy to be implemented. After each iteration, "System Manager" performs the analysis of detected changes in the environment, as well as the analysis of the effects of applying the learning algorithm in previous iterations. The results of these anal-

ysis can be used in further upgrading of the proposed concept to correct the application of the learning algorithm (in terms of correction the values of parameters M and N , etc).

The resistance of this segment of the system to the influence of disturbances and noises mostly depends on the probability of detection P_D and false alarm P_F of the applied M -out-of- N detector, which will be defined below [10].

If the probability of detection of single cell status changing in map by robot's sensors, i.e. bit status changing in learning algorithm in each of N single bits is p_d , then the probability of declaring a initial bit status changing in at list M out of N bits, is:

$$P_D = \sum_{k=M}^N P(k) = \sum_{k=M}^N \binom{N}{k} p_d^k (1-p_d)^{(N-k)}.$$

Similarly, if the probability of false alarm per single cell status changing in map by robot's sensors is p_{fa} , then the probability of declaring a false alarm in at least M out of N bits, resulting in the declaration of a false initial bit status changing, is:

$$P_F = \sum_{k=M}^N \binom{N}{k} p_{fa}^k (1-p_{fa})^{(N-k)}.$$

In real systems, P_D and P_F depend on the characteristics of the robot's sensors, but also on the characteristics of the multi-robot system and the environment in which the robots move, and it is usually valid that $p_d \gg p_{fa}$.

5. Simulation, Results and Discussion

The simulation was organized completely in accordance with Figure 1 and the description of the system architecture given in Section 3.

All modules are included, but some processes can be underlined as specific from the aspect of the topic of this paper, as follows:

- checking the shelves and determining the percentage of their fullness with goods after the specified intervals,
- generating a notification if the shelf fullness is lower than the preset level,
- planning of engagement the available robots for goods transport from the warehouse to the store

with the specification of the start and the goal positions of each robot,

- planning/replanning of robots paths,
- implementation of fuzzy inference system and learning algorithm with a purpose of improvement the path planning efficiency,
- motion coordination,
- solving a conflict situations and updating of environment map based on information collected with robots sensors.

These are the most complicated and time consuming tasks, so they are performed on cloud level in simulation (Figure 1).

In order to test the characteristics of the proposed approach for improvement the efficiency of path planning on global level, we applied it to a large number of various data sets in simulated megastore scenario. The map of store in accordance with Section 4.2. is represented as a discrete 16x16 grid. To apply graph-based search technique, it is adopted that the robot can move from an unblocked cell to one of the eight neighboring cells, if it is also unblocked (eight-connected graph). The costs of edges are obtained by multiplying of the traversal cost of the cell in which the robot makes the move and the length of the edge (one for orthogonal movements or $\sqrt{2}$ for diagonal movements), while the costs of edges into obstacles are infinite.

The information about environment changing is not apriori available to robots and system manager. As the robots move from start to goal locations, they receive new information concerning the traversability of surrounding cells within sensor range (one cell in all directions).

Fuzzy system and learning algorithm were tested separately to simplify the analysis of the effect of their application. This approach is applied because these two systems work independently and are designed to improve path planning by exploiting different information from the environment and its dynamics. With this in mind, in a real situation they would not interfere with each other, but the effects of their application would collectively contribute to the improvement of path planning. The cloud approach has always been applied, both in the case of the fuzzy system application and in the case of the learning algorithm application.

In order to test fuzzy inference system presented in Section 4.2., the probability of occupancy of the cells

nearest to the third row of shelves depending on the iteration number was exactly as on the graph in the Figure 4, but with the possibility of deviation in each iteration of $\pm 20\%$ (to test at the same time the robustness of the system and the proposed method from the aspect of fuzzy implementation). Also, moving away from the third row of shelves for one cell in all directions the probability of occupancy decreases by 33%. The traversal costs of the cells are defined in accordance with the Figure 6.

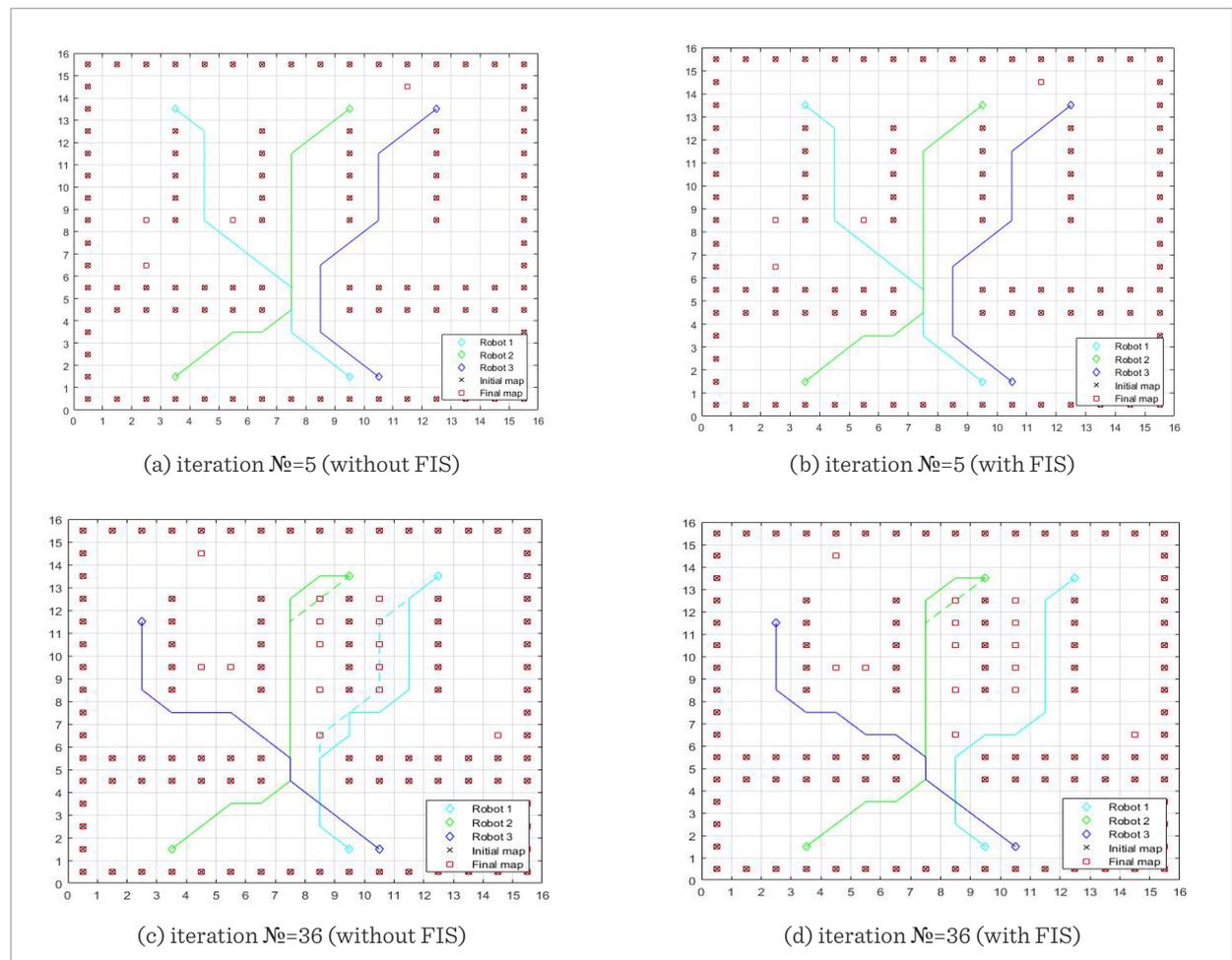
The results of robots path planning in the phase of goods transport from the warehouse to the store, in iterations that present the effects of using fuzzy inference system, are shown in Figure 8.

Dashed line represents initially computed path, while continuous line is final path. These two lines do not

match in general case, having in mind that during the robot motion along the initial path the changes in an environment are detected which can require the correction of the motion plan.

The situation shown in Figure 8 a) and Figure 8 b) implies the time when the crowd in the observed part of the store is not large, so the fuzzy inference system defines relatively low traversal cost of the appropriate cells in the map. In this case, D* Lite algorithm generates the same initial solution regardless of whether or not fuzzy logic is applied. There is no correction of global path, also. On the other side, in Figure 8 c) and Figure 8 d), path planning in situation of the largest crowd density is presented. It can be seen that Robot 1 was forced to make two corrections of the path (on global level) in the situation without using the fuzzy

Figure 8
Path planning results without using FIS and with using FIS



inference system. The assumption is that a local planner cannot solve this problem in a safe way so a path replanning on global level is needed. In the situation with the application of the fuzzy inference system there was no correction of the path. Also, it can be concluded that path in situation without using the fuzzy inference system is quite risky from the aspect of the human-robot interactions.

We have run several cycles with 100 consecutive iterations in every cycle and calculated mean total paths length per cycle. We got that the mean total paths length per cycle in the situation with the implementation of the fuzzy system is shorter for about 2% compared to the option without the application of a fuzzy system.

Also, the simulation experiments for testing learning approach were conducted in the same map and discrete grid. Two scenarios are presented in Figure 9.

Figure 9

Scenarios I and II overview (new obstacles are shown in grey)

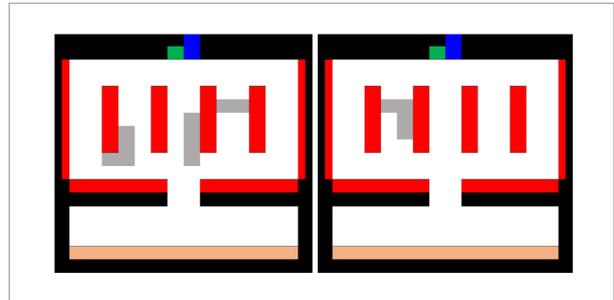
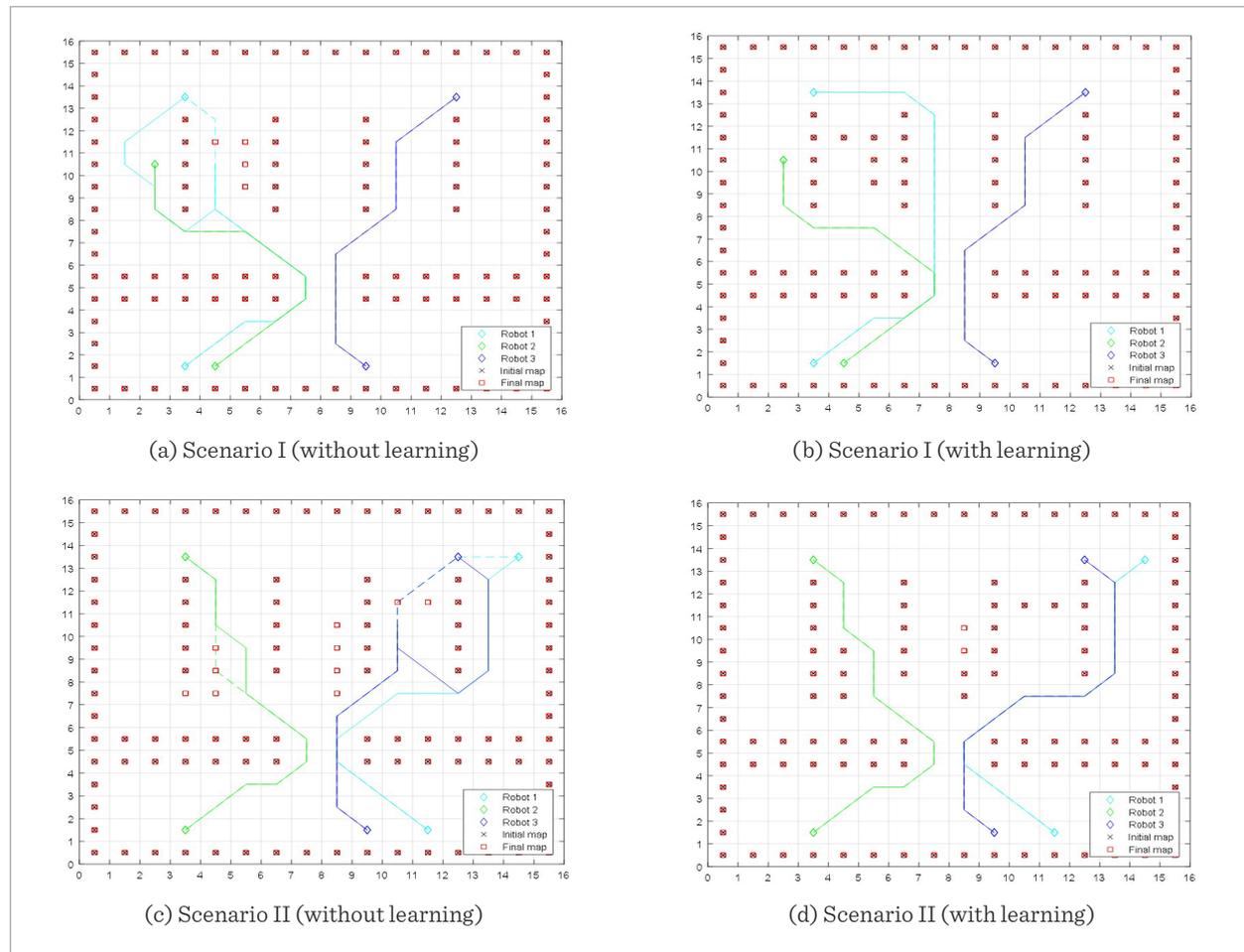


Figure 10

Path planning results without using learning algorithm and with using learning algorithm



In scenario I there is one obstacle that occupied several cells with occupancy-time $T_1=30T$, where T is time between two consecutive iterations of robots engagement. In scenario II three obstacles appeared with occupancy-times T_{21} , T_{22} and T_{23} ($T_{21} \neq T_{22} \neq T_{23}$) with duration between $10T$ and $35T$. The parameters of binary moving-window detector are set to $M=3$ and $N=5$.

For every scenario we have run several cycles with 100 consecutive iterations in every cycle and calculated mean expanded cells and mean total paths length per cycle.

The results of robots path planning in the phase of transport of goods from the warehouse to the store, for scenarios I and II in iterations that present the effects of using learning algorithm, are shown in Figure 10.

The statistics for described scenarios is shown in Table 2.

Table 2

Mean total paths length reduction with using learning algorithm

Scenario	Mean total paths length per cycle of 100 iterations		Reduction (%)
	<i>without learning algorithm</i>	<i>with learning algorithm</i>	
I	4342	4289	1,22%
II	4558	4391	3,66%

Based on Table 2, it can be concluded that mean total paths lengths are significantly reduced with using learning algorithm (Figure 10 b and Figure 10 d), compared to situations when learning is not applied (Figure 10 a and Figure 10 c). This is because the application of learning algorithm, as well as the fuzzy inference system, provides an adaptive approach to path calculating by D^* Lite, with taking into account the experience in early planning phase. In this way the creation of smart cost map is achieved, so that the initial paths on global level are searched with predicted changes in the environment.

The robustness of the system and the proposed method from the aspect of learning algorithm implementation depends in general on probability of detection and false alarm of applied M -out-of- N detector and this problem is considered in Section 4.3.

6. Conclusions

In this paper an autonomous cloud-based multi-robot system designed to execute highly repetitive tasks in a dynamic environment such as a modern megastore is considered. Cloud level is intended for performing the most demanding operations related to data processing, allocation of tasks between robots, application of the fuzzy inference system and learning algorithm, path planning and replanning, motion coordination, etc. A direct consequence of this is the unloading of robots because they are users of cloud services in this architecture.

D^* Lite is applied as algorithm for path planning on global level, taking into account its high efficiency in environments that are partially known or completely unknown. In order to further improve path planning process in complex and crowded environment, implementation of smart cost map based on fuzzy inference system and learning algorithm is proposed.

Proposed concept substantially reduces the total paths length. This makes preconditions for energy saving, as well as for increasing the average speed of the robots, so that they reach the destination for a shorter time and perform tasks more efficiently. Also, the risk in path planning related to the human-robot interactions can be reduced.

The important advantage of the system is simplicity of implementation. Its main disadvantage is sensitivity of the communication loss.

A perspective for future works would be to implement some advanced machine learning techniques in combination with multi-criteria decision making methods, in order to reinforce smart cost map intended for path planning. The idea is based on the fact that a dynamic environment may have, besides crowds and obstacles, other specifics that affect the robots motion [41, 42] and that are desirable to consider in the initial planning process.

Acknowledgement

The authors of this paper would like to express their gratitude to the Ministry of Education, Science and Technological Development of Republic of Serbia, for their support of Contract No 451-03-68/2020-14/200325.

References

1. Al-Mutib, K., AlSulaiman, M., Emaduddin, M., Ramdane, H., Mattar, E. D* Lite Based Real-Time Multi-Agent Path Planning in Dynamic Environments. In 2011 Third International Conference on Computational Intelligence, Modelling & Simulation, Langkawi, Malaysia, 2011. <https://doi.org/10.1109/CIMSim.2011.38>
2. Aroor, A., Epstein, S. L., Korpan, R. Online Learning for Crowd-Sensitive Path Planning. Proceedings of the 17th International Conference on Autonomous Agents and Multi-Agent Systems, 2018, 1702-1710.
3. Benavidez, P., Muppidi, M., Rad, P., Prevost, J. J., Jamshidi, M., Brown, L. Cloud-Based Realtime Robotic Visual SLAM. Proceedings of the 2015 Annual IEEE Systems Conference, Vancouver, BC, Canada, 2015, 773-777. <https://doi.org/10.1109/SYSCON.2015.7116844>
4. Bennewitz, M., Burgard, W., Thrun, S. Optimizing Schedules for Prioritized Path Planning of Multi-Robot Systems. Proceedings of the 2001 IEEE International Conference on Robotics and Automation, Seoul, South Korea, 2001, 271-276. <https://doi.org/10.1109/ROBOT.2001.932565>
5. Berenson, D., Abbeel, P., Goldberg, K. A Robot Path Planning Framework that Learns from Experience. 2012 IEEE International Conference on Robotics and Automation, Saint Paul, MN, USA, 2012, 3671-3678. <https://doi.org/10.1109/ICRA.2012.6224742>
6. Brumitt, B., Stentz, A. GRAMMPS: A Generalized Mission Planner for Multiple Mobile Robots in Unstructured Environments. Proceedings of the 1998 IEEE International Conference on Robotics and Automation, Leuven, Belgium, 1998, 1564-1571. <https://doi.org/10.1109/ROBOT.1998.677360>
7. Choudhary, S., Carlone, L., Nieto, C., Rogers, J., Liu, Z., Christensen, H. I., Dellaert, F. Multi Robot Object-Based SLAM. Proceedings of the International Symposium on Experimental Robotics, Tokyo, Japan, 2016, 729-741. https://doi.org/10.1007/978-3-319-50115-4_63
8. Cirillo, M., Uras, T., Koenig, S. A Lattice-Based Approach to Multi-Robot Motion Planning for Non-Holonomic Vehicles. 2014 IEEE/RSJ International Conference on Intelligent Robots and Systems, Chicago, IL, USA, 2014, 232-239. <https://doi.org/10.1109/IROS.2014.6942566>
9. Coleman, D., Sucas, I. A., Moll, M., Okada, K., Correll, N. Experience-Based Planning with Sparse Roadmap Spanners. 2015 IEEE International Conference on Robotics and Automation, Seattle, WA, USA, 2015, 900-905. <https://doi.org/10.1109/ICRA.2015.7139284>
10. Dillard, G. M. A moving window detector for binary integration. IEEE Transactions on Information Theory, 1967, 13(1), 2-6. <https://doi.org/10.1109/TIT.1967.1053967>
11. Ferguson, D., Stentz, A. Field D*: An Interpolation-Based Path Planner and Replanner. Robotics Research - Results of the 12th International Symposium of Robotics Research, San Francisco, CA, USA, 2005, 239-253. https://doi.org/10.1007/978-3-540-48113-3_22
12. Ferguson, D., Likhachev, M. Efficiently Using Cost Maps For Planning Complex Maneuvers. Proceedings of International Conference on Robotics and Automation - Workshop on Planning with Cost Maps, 2008.
13. Hunziker, D., Gajamohan, M., Waibel, M., D'Andrea, R. Rapyuta: The RoboEarth Cloud Engine. 2013 IEEE International Conference on Robotics and Automation, Karlsruhe, Germany, 2013, 438-444. <https://doi.org/10.1109/ICRA.2013.6630612>
14. Hwang, V., Phillips, M., Srinivasa, S., Likhachev, M. Lazy Validation of Experience Graphs. 2015 IEEE International Conference on Robotics and Automation, Seattle, WA, USA, 2015, 912-919. <https://doi.org/10.1109/ICRA.2015.7139286>
15. Jetchev, N., Toussaint, M. Trajectory Prediction: Learning to Map Situations to Robot Trajectories. Proceedings of the 26th Annual International Conference on Machine Learning, Montreal, Quebec, Canada, 2009, 449-456. <https://doi.org/10.1145/1553374.1553433>
16. Kamburugamuve, S., He, H., Fox, G. C., Zhao, W. Cloud based Real-Time Multi-Robot Collision Avoidance for Swarm Robotics. International Journal of Grid and Distributed Computing, 2016, 9(6), 339-358. <https://doi.org/10.14257/ijgdc.2016.9.6.30>
17. Koenig, S., Likhachev, M. D* Lite. Proceedings of the Eighteenth National Conference on Artificial Intelligence and Fourteenth Conference on Innovative Applications of Artificial Intelligence, Edmonton, Alberta, Canada, 2002, 476-483
18. Koenig, S., Likhachev, M. Fast Replanning for Navigation in Unknown Terrain. IEEE Transactions on Robotics, 2005, 21(3), 354-363. <https://doi.org/10.1109/TRO.2004.838026>
19. LaValle, S. M. Planning Algorithms. Cambridge University Press, 2006. <https://doi.org/10.1017/CBO9780511546877>

20. Lien, J.-M., Lu, Y. Planning Motion in Environments with Similar Obstacles. In 2009 Conference Robotics: Science and Systems V, 2009. <https://doi.org/10.15607/RSS.2009.V.012>
21. Likhachev, M., Ferguson, D., Gordon, G., Stentz, A., Thrun, S. Anytime Search in Dynamic Graphs. *Journal Artificial Intelligence*, 2008, 172(14), 1613-1643. <https://doi.org/10.1016/j.artint.2007.11.009>
22. Likhachev, M., Ferguson, D., Gordon, G., Stentz, A., Thrun, S. Anytime Dynamic A*: An Anytime, Replanning Algorithm. Proceedings of the 15th International Conference on Automated Planning and Scheduling, Monterey, CA, USA, 2005, 262-271.
23. Likhachev, M., Ferguson, D., Gordon, G., Stentz, A., Thrun, S. Anytime Dynamic A*: The proofs. Technical Report, CMU-RI-TR-05-12, Robotics Institute, Carnegie Mellon University, 2005.
24. Lozano-Perez, T. Spatial Planning: A Configuration Space Approach. *IEEE Transactions on Computing*, 1983, C-32(2), 108-120. <https://doi.org/10.1109/TC.1983.1676196>
25. Martin, S. R., Wright, S. E., Sheppard, J. W. Offline and Online Evolutionary Bi-directional RRT Algorithms for Efficient Replanning in Environments with Moving Obstacles. 2007 IEEE International Conference on Automation Science and Engineering, Scottsdale, AZ, USA, 2007. <https://doi.org/10.1109/COASE.2007.4341761>
26. Murphy, E. Planning and Exploring Under Uncertainty. PhD thesis, University of Oxford, 2010.
27. Messinis, S., & Vosniakos, G. C. (2020). An Agent-Based Flexible Manufacturing System Controller with Petri-Net Enabled Algebraic Deadlock Avoidance. *Reports in Mechanical Engineering*, 1(1), 77-92. <https://doi.org/10.31181/rme200101077m>
28. Nardi, L., Stachniss, C. Experience-Based Path Planning for Mobile Robots Exploiting User Preferences. 2016 IEEE/RSJ International Conference on Intelligent Robots and Systems, 2016, 1170-1176. <https://doi.org/10.1109/IROS.2016.7759197>
29. Oliver, C. S., Saptharishi, M., Dolan, J. M., Trebilcock, A., Khosla, P. K. Multi-Robot Path Planning by Predicting Structure in a Dynamic Environment. Proceedings of the Conference on Mechatronic Systems, 2000, 555-560. [https://doi.org/10.1016/S1474-6670\(17\)39203-0](https://doi.org/10.1016/S1474-6670(17)39203-0)
30. Peng, J.-H., Li, I.-H., Chien, Y.-H., Hsu, C.-C., Wang, W.-Y. Multi-Robot Path Planning Based on Improved D* Lite Algorithm. 2015 IEEE 12th International Conference on Net-working, Sensing and Control, Taipei, Taiwan, 2015, 353-353. <https://doi.org/10.1109/ICNSC.2015.7116061>
31. Precup, R.-E., Preitl, S., Petriu, E., Bojan-Dragos, C.-A., Szedlak-Stinean, A.-I., Roman, R.-C., Hedrea, E.-L. Model-Based Fuzzy Control Results for Networked Control Systems. *Reports in Mechanical Engineering*, 2020, 1(1), 10-25. <https://doi.org/10.31181/rme200101010p>
32. Phillips, M., Likhachev, M. Speeding up Heuristic Computation in Planning with Experience Graphs. 2015 IEEE International Conference on Robotics and Automation, Seattle, WA, USA, 2015, 893-899. <https://doi.org/10.1109/ICRA.2015.7139283>
33. Phillips, M., Cohen, B., Chitta, S., Likhachev, M. E-graphs: Bootstrapping Planning with Experience Graphs. In 2012 Conference Robotics: Science and Systems VIII, 2012. <https://doi.org/10.15607/RSS.2012.VIII.043>
34. Saha, O., Dasgupta, P. Fast Path Planning Using Experience Learning from Obstacle Patterns. In 2016 Conference AAAI Spring Symposium Series, Palo Alto, California, 2016, 60-67.
35. Wagner, G., Choset, H. M*. A Complete Multirobot Path Planning Algorithm with Performance Bounds. 2011 IEEE/RSJ International Conference on Intelligent Robots and Systems, San Francisco, CA, USA, 2011, 3260-3267. <https://doi.org/10.1109/IROS.2011.6095022>
36. Wiktor, A., Scobee, D., Messengery, S., Clark, C. Decentralized and Complete Multi-Robot Motion Planning in Confined Spaces. 2014 IEEE/RSJ International Conference on Intelligent Robots and Systems, Chicago, USA, 2014, 1168-1175. <https://doi.org/10.1109/IROS.2014.6942705>
37. Vilela, M., Oluyemi, G., Petrovski, A. A Fuzzy Inference System Applied to Value of Information Assessment for Oil and Gas Industry. *Decision Making: Applications in Management and Engineering*, 2019, 2(2), 1-18. <https://doi.org/10.31181/dmame1902001v>
38. Milosevic, T., Pamucar, D., Chatterjee, P. Model for Selecting a Route for the Transport of Hazardous Materials Using a Fuzzy Logic System. *Military Technical Courier*, 2021, 69(2), 355-390. <https://doi.org/10.5937/vojtehg69-29629>
39. Dimic, S., Ljubojevic, S. Decision Making Model in Forest Road Network Management. *Military Technical Courier*, 2019, 67(1), 93-115. <https://doi.org/10.5937/vojtehg67-18446>

40. Vilela, M., Oluyemi, G., & Petrovski, A. A Holistic Approach to Assessment of Value of Information (VOI) with Fuzzy Data and Decision Criteria. *Decision Making: Applications in Management and Engineering*, 2020, 3(2), 97-118. <https://doi.org/10.31181/dma-me2003097v>
41. Zagradjanin, N., Pamucar, D., Jovanovic, K. Cloud-Based Multi-Robot Path Planning in Complex and Crowded Environment with Multi-criteria Decision Making Using Full Consistency Method. *Symmetry*, 2019, 11(10), 1-15. <https://doi.org/10.3390/sym11101241>
42. Ziebart, B. D., Ratliff, N., Gallagher, G., Mertz, C., Peterson, K., Bagnell, J. A., Hebert, M., Dey, A. K., Srinivasa, S. Planning-Based Prediction for Pedestrians. *Proceedings of the 2009 IEEE/RSJ International Conference on Intelligent robots and systems*, St. Louis, MO, USA, 2009, 3931-3936. <https://doi.org/10.1109/IROS.2009.5354147>
43. Zucker, M., Kuffner, J., Branicky, M. Multipartite RRTs for Rapid Replanning in Dynamic Environments. *Proceedings of the 2007 IEEE International Conference on Robotics and Automation*, Rome, Italy, 2007, 1603-1609. <https://doi.org/10.1109/ROBOT.2007.363553>



This article is an Open Access article distributed under the terms and conditions of the Creative Commons Attribution 4.0 (CC BY 4.0) License (<http://creativecommons.org/licenses/by/4.0/>).