This is a preprint version of the article: Hvězda, J., Rybecký, T., Kulich, M., and Přeučil, L. (2018). **Context-Aware Route Planning for Automated Warehouses**. *Proceedings of 2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, which has been published in a final form at https://doi.org/10.1109/ITSC.2018.8569712.

Context-Aware Route Planning for Automated Warehouses

Jakub Hvězda*†, Tomáš Rybecký †, Miroslav Kulich* and Libor Přeučil*
* Czech Institute of Informatics, Robotics, and Cybernetics
Czech Technical University in Prague
Prague, Czech Republic
† Department of Cybernetics
Faculty of Electrical Engineering
Czech Technical University in Prague
Prague, Czech Republic

***Abstract*—In order to ensure efficient flow of goods in an automated warehouse and to guarantee its continuous distribution to/from picking stations in an effective way, decisions about which goods will be delivered to which particular picking station by which robot and by which path and in which time have to be made based on the current state of the warehouse. This task involves solution of two suproblems: (1) task allocation in which an assignment of robots to goods they have to deliver at a particular time is found and (2) planning of collision-free trajectories for particular robots (given their actual and goal positions).**

The trajectory planning problem is addressed in this paper taking into account specifics of automated warehouses. First, assignments of all robots are not known in advance, they are instead presented to the algorithm gradually one by one. Moreover, we do not optimize a makespan, but a throughput – the sum of individual robot plan costs.

We introduce a novel approach to this problem which is based on the context-aware route planning algorithm [1]. The performed experimental results show that the proposed approach has a lower fail rate and produces results of higher quality than the original algorithm. This is redeemed by higher computational complexity which is nevertheless low enough for real-time planning.

***Index Terms*—intelligent logistics, planning, coordination**

I. Introduction

The European market for e-commerce is growing rapidly, with more than 16% just in the year 2014 [2] and so is the same for the world market. With the growing markets, the need for larger warehouses and distribution centers and their automation increases [3]. In such facilities goods for the end-users or products in the business-to-business sector are stored, commissioned and shipped. To manage the supply chains, many new warehouses have been erected, and more will follow (see Fig. 1An automated warehouse: G-COM system by Grenzebach (https://www.grenzebach.com) in a costumer applicationfigure.1 for an example of such automated warehouse). Reliable figures are hard to come by, but for instance, a company such as DHL alone operates more than 2000 warehouses. With the growing markets, the need for larger warehouses and their automation increases. Therefore the robotic and automation companies should be able to provide appropriate solutions, making scalable systems and scalable software mandatory.

Fig. 1. An automated warehouse: G-COM system by Grenzebach (https://www.grenzebach.com) in a costumer application.

One of the fundamental problems related to automated warehouses is trajectory planning and motion coordination for a fleet of robots distributing goods in a warehouse which has been theoretically studied from many perspectives since the 1980s, see [4] for a nice overview. The problem (also formulated as the warehouseman's problem) has been proven to be PSPACE-complete [5]. The complexity of the problem can be reduced for the case where robots move on a predefined graph, nevertheless, it is still NP-hard [6], which means that optimal solutions cannot generally be found in a reasonable time for non-trivial instances (e.g., for a number of robots in order of tens).

Solutions to the problem consider either coupled or decoupled approaches. Coupled (also called centralized) approaches consider a fleet of robots as a single multi-robot body with the number of degrees of freedom (DoFs) equal to the sum of DoFs of individual robots. Classical single-robot plan-

ning approaches can then be applied to plan motion of this multi-robot body in composite configuration space. Because these algorithms are typically based on classical complete and optimal algorithms, their main advantage is that they warrant to find a solution if it exists and report its non-existence if it does not exist [7], [8], [9]. This, however, leads to the main disadvantage of these algorithms which is the exponential computational time in the dimension of the composite configuration space. The appropriate use cases for this type of algorithms are thus problems that consist of a small number of robots.

On the other hand, algorithms that use the decoupled approach typically provide solutions much faster than coupled planners for the price of the solutions being sub-optimal. In addition to this, decoupled algorithms generally do not guarantee the completeness and may suffer from deadlocks. These approaches can be typically split into two categories - path coordination and prioritized planning. Path coordination tunes velocities of individual robots along precomputed trajectories to avoid collisions of these robots [10], [11]. Prioritized planning assigns each agent a priority according to which an ordering is made for the agents. The planning algorithm then plans trajectories for individual robots according to this ordering, while considering already planned robots as moving obstacles that have to be avoided. [12], [13], [14].

[1] presents a similar idea, but codes information about trajectories of robots with higher priorities into the planning graph rather than into the planning algorithm itself. It does so by constructing a resource graph, where each resource can be for example a node of the original graph or intersection graph edges. Every such resource holds information about time intervals in which it is not occupied by already planned robots. An adaptation of the A* algorithm is used on this graph to find the shortest path through these intervals (called free time windows) to obtain a path that avoids all already planned robots.

[15] deals with a real-life problem of routing vehicles in Container Terminal Altenwerder in Hamburg harbor. They used a similar approach to keeping a set of free time windows for path arcs in the graph. Their algorithm contains a preprocessing of the graph for the use of specific vehicles followed by computation of paths for individual vehicles on this preprocessed graph.

Another similar approach is presented in [16], where each robot looks for a viable path in a 2D spatial grid map and checks for collisions with moving obstacles using a temporal occupancy table. [17] adopted a similar approach to [1] with enhanced taxiway modeling approach to improve performance on airport graph structures. [18] compares Context-Aware Route Planning (CARP) [1] with a fixed-path scheduling algorithm using k shortest paths [19] and a fixed-path scheduling algorithm using k disjoint paths [20]. The experiments show that the CARP algorithm is superior in all measured qualities.

[21] compares several heuristic approaches of assigning priority to robots and concludes that the heuristics which plans longest paths first perform best when a makespan is to be minimized. A greedy best-first heuristics provides best results regarding joint plan cost. However, its downside is that it calls the planning algorithm for all yet unplanned robots in every round and it is thus very time-consuming.

In this paper, we consider the planning problem from the perspective of usability in automated warehouses (AWs). The key difference in AWs is that the assignments are not known all at once at the time of planning, but they instead come in time sequentially as requests for goods to be delivered are being received. This emphasizes a quick time to compute a plan for newly requested robots while minimizing the sum of time it takes for all robots in the system to complete their plans. We present a novel approach that can calculate paths for newly added robots to the system while maximizing its throughput.

The rest of the paper is organized as follows. The multi-robot path-finding problem for automated warehouses is presented as well as the used terms are defined in Section IIThe problemsection.2. The proposed planning algorithm is described in Section IIIAlgorithmsection.3, while performed experiments, their evaluation, and discussion are presented in Section IVExperimentssection.4. Finally, Section VConclusionsection.5 is dedicated to concluding remarks and future work.

II. The problem

Assume a finite weighted connected graph $G(V, E)$ and a set $\mathcal{A} = (a_1, a_2, ..., a_{k-1})$ of robots each of which has already planned a trajectory in G without collisions. The aim is to find a non-colliding trajectory for a robot a_k from its start position to the given target position while optimizing a global cost function, possibly without replanning all robots.

We employ the CARP algorithm [1] for the planning of individual robots, so we use its assumptions on the graph structure as follows. The infrastructure the robots plan on is modeled as a resource graph $G_R = (R, E_R)$ where resources R can be the original nodes, edges, intersections of edges, etc. The path the robot can follow is restricted to edges E_R which limit transitions of robots such that a robot can move from a resource r_1 to a resource r_2 only if r_2 is a neighbour of r_1 in the graph G_R, i.e. the edge $(r_1, r_2) \in E_R$. Each resource also has two main attributes associated with it. These are capacity $c(r)$ which corresponds to the maximum number of robots that can occupy the resource r at the same time and duration $d(r) > 0$ that represents the minimal time it takes the robot to traverse the given resource r. Plans for every robot then contain not only a sequence of resources on its path but also time intervals during which the robot visits them. The idea is that if the infrastructure is modeled this way, collision avoidance is shifted into deciding which resources at what time the robots can visit, rather than finding paths without collision in space and time, which is PSPACE-complete [5].

III. Algorithm

As mentioned, the proposed algorithm employs CARP [1]. It was chosen because of its ability to plan sequentially one

robot at a time while keeping the information about movement of other robots in a compact and easy to update manner. More specifically, CARP uses the infrastructure described above to find paths in a resource graph, where each node of the infrastructure keeps information about its capacity and occupancy in given time windows. This allows it to use a modified A* algorithm that finds a free time window in the start resource corresponding to a start node and attempts to find a path through these time windows to a free time window in the resource that corresponds to the goal node.

The proposed algorithm aims to generate a trajectory for an robot a_k assuming that trajectories for $k-1$ robots are already planned which can possibly lead to modification of those planned trajectories. The main idea is to iteratively build a set of robots whose trajectories mostly influence an optimal trajectory of a_k, see Algorithm 1Algorithmalgocf.1. The algorithm maintains two structures:

- $\mathcal{N}$ – a set of robots in the neighborhood of a_k which is initially set to contain a_k (line 3AlgorithmAlgoLine.1.3), and
- $\mathcal{A}$ – a sequence of robots not in $\mathcal{N}$. This sequence initially stores the order in which trajectories of the robots $a_1 \dots a_{k-1}$ were generated (line 1AlgorithmAlgoLine.1.1).

Moreover, cost of the best solution that has been found so far is set to a high number.

New robots are iteratively added to the neighborhood in the loop starting at line 5AlgorithmAlgoLine.1.5. The robot $a_b \in \mathcal{A}$ which minimizes the distance to the neighborhood is found at each iteration first, where the distance of a robot to a set is defined as the distance of the robot to the closest robot in the set. The distance between two robots is then determined as the average Euclidean distance of robots' positions at discrete time steps:

$$d(a_i, a_j) = \frac{\sum_{\tau=\tau_S}^{\tau_G} \left| t_i(\tau), t_j(\tau) \right|}{\tau_G - \tau_S},$$

where $t_i(\tau)$ is a position of a_i at time τ if it follows the trajectory t_i, $t_j(\tau)$ is a position of a_j at time τ if it follows the trajectory t_j, and $\langle \tau_S, \tau_G \rangle$ is a time interval when a_i or a_j moves. Note that trajectories of robots in $\mathcal{A}$ are initially taken from the input set $\mathcal{T}$ while an initial trajectory of a_k is determined as the shortest path between its start and goal positions on G making use of the A* algorithm (line 2AlgorithmAlgoLine.1.2). These initial trajectories are updated as soon as new plans are found at the next steps of the algorithm.

The found closest robot a_b is then removed from the sequence $\mathcal{A}$ (line 7AlgorithmAlgoLine.1.7), added to a set of neighbors $\mathcal{N}$ (line 7AlgorithmAlgoLine.1.7) and new trajectories for $\mathcal{A}$ are computed by the CARP algorithm from the scratch (line 9AlgorithmAlgoLine.1.9) as demonstrated in Fig. 2Adding an robot into a set of neighborsfigure.2.

All possible permutations of robots in $\mathcal{N}$ are considered next (line 10AlgorithmAlgoLine.1.10). A set of trajectories $\mathcal{P}$ is determined by the CARP algorithm taking into account trajectories $\mathcal{C}$ for each such permutation (line 11AlgorithmAlgoLine.1.11). This is realized by running CARP for $\mathcal{N}$ on a resource graph with free windows generated by CARP when computing $\mathcal{C}$ at line 9AlgorithmAlgoLine.1.9. The set $\mathcal{P}$ is then added to $\mathcal{C}$, and the cost of this solution is computed (line 12AlgorithmAlgoLine.1.12) and compared with the best solution found till now (line 13AlgorithmAlgoLine.1.13). If the new solution is better than the currently best, it is stored together with its cost (lines 14AlgorithmAlgoLine.1.14 and 15AlgorithmAlgoLine.1.15). The best-found solution is finally reported at line 16AlgorithmAlgoLine.1.16.

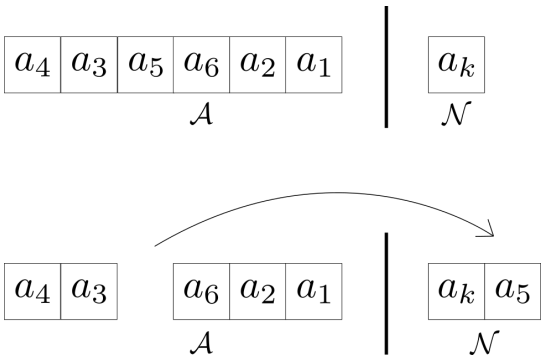

Fig. 2. Adding an robot into a set of neighbors.

Algorithm 1: Plan update

Input: $G = (V, E)$ – a graph
$\mathcal{T} = \{t_i\}_{i=1}^{k-1}$ – already planned trajectories
V_S – start position
V_G – goal position
M – size of a neighborhood

Output: $\mathcal{T}^{new} = \{t_i\}_{i=1}^{k}$ – updated set of trajectories

1 $\mathcal{A} \leftarrow \langle a_1, a_{k-1} \rangle$
2 $t_k \leftarrow$ shortest path from V_S to V_G in G
3 $\mathcal{N} \leftarrow \{a_k\}$
4 $best \leftarrow \infty$
5 $M-1$ **times do**
6 $a_b = \text{argmin}_{a_i \in \mathcal{A}} d(a_i, \mathcal{N})$
7 $\mathcal{A} \leftarrow \mathcal{A} \setminus \{a_b\}$
8 $\mathcal{N} \leftarrow \mathcal{N} \cup \{a_b\}$
9 $\mathcal{C} \leftarrow CARP(G, \mathcal{A})$
10 **foreach** $\pi \in \Pi(\mathcal{N})$ **do**
11 $\mathcal{P} \leftarrow CARP(G, \pi, C)$
12 $c \leftarrow cost(\mathcal{C} \cup \mathcal{P})$
13 **if** $c < best$ **then**
14 $best \leftarrow c$
15 $\mathcal{T}^{new} \leftarrow \mathcal{C} \cup \mathcal{P}$
16 **return** $\mathcal{T}^{new}$

Calculation of computational complexity of the proposed

algorithm is based on the fact that CARP for n robots comprises n calls of A*. We call CARP $M-1$ times at line 9AlgorithmAlgoLine.1.9 gradually for $k-2, k-3, \dots k-M-1$ robots which leads to $\frac{M}{2}(2k-M-3)$ calls of A*. Similarly, CARP at line 11AlgorithmAlgoLine.1.11 is called $\sum_{N=2}^{M} N!$ times which leads to $\sum_{N=2}^{M} NN!$ calls of A*.

The total number of A* calls can be significantly reduced in two ways. Firstly, when calling CARP at line 9AlgorithmAlgoLine.1.9 after removal of a_b not all trajectories have to be recomputed. We can instead preserve trajectories of robots which were in $\mathcal{A}$ before a_b as they are not influenced by a_b. Only trajectories of robots behind a_b in $\mathcal{A}$ have to be recomputed which leads to a reduction of A* calls by 50% in average.

The second reduction is similar. If the permutations are generated in a lexicographic order at line 10AlgorithmAlgoLine.1.10 then two consecutive permutations have typically a big joint head as depicted in Fig. 3First permutations of five elements in the lexicographic order. The yellow plans can preservedfigure.3. Plans of robots in that head can be preserved while recomputation has to be done only for the rest. Assuming neighborhood size $|\mathcal{N}| = 4$, instead of calling A* $4 \times 4! = 96$ times, only 64 calls is performed which is 67%. The reduction is even greater for $|\mathcal{N}| = 5$: 325 calls instead of 600 which is 54%, while only 45% (1956 instead of 4320) A* calls are needed for $|\mathcal{N}| = 5$.

1	2	3	4	5
1	2	3	5	4
1	2	4	3	5
1	2	4	5	3
1	2	5	3	4

Fig. 3. First permutations of five elements in the lexicographic order. The yellow plans can preserved.

IV. Experiments

The experiments were performed on a computer equipped with Intel Xeon E5-2690. The maps the experiments were performed on were created to show how the proposed algorithm performs depending on the density of the given graph, specifically the number of edges. The set of 21 maps was generated by creating a minimum spanning tree of a 20×20 grid map and then iteratively adding a given number of original edges to it until the original grid was recreated. Furthermore, 500 different random assignments were generated for a fleet of 100 robots by randomly sampling start and goal nodes for each robot. Each of these assignments was tested on all 21 maps.

The goal of the experiments was to test how the proposed algorithm scales with the number of edges in the graph as robots are sequentially added to the system. For comparison we ran the original CARP algorithm with no change to priorities of the robots, i.e., priorities were set randomly. Because this approach proved to have a high failure rate, we introduced two variants that after each planning attempt randomly shuffled the robot order 10 and 100 times (`CARP10`, `CARP100` respectively) and tried planning again from scratch. The best plan regarding the sum of the number actions of individual robots was considered as the result. We also added CARP with the longest first heuristic to determine robot priorities as introduced in [21] to the comparison as `LF`. The proposed algorithm was run in several variants that differ in the parameter $\mathcal{M}$ specifying the size of the neighborhood. For the parameters 4,5,6 (`Proposed_4`, `Proposed_5`, `Proposed_6` respectively) all the permutations of the neighborhood were considered. For the parameter 10 (`Proposed_10`) a 150 different neighborhood permutations were chosen randomly to decrease computational complexity.

Orders for all 100 robots were presented sequentially to all algorithms on all maps and assignments in the first experiment. Failure rate (Fig. 4Algorithms fail ratefigure.4) and the number of actions of all robots were observed (Fig. 6Sum of actions performed by all robotsfigure.6). In the second experiment, the algorithms were sequentially given one robot at a time at each iteration for all 500 assignments on a map shown in Fig. 5Example of an experimental mapfigure.5 with the goal of showing how much time it takes to generate a plan for each algorithm after adding robot into the system.

The results for the failure rate of the algorithms can be seen in Figure 4Algorithms fail ratefigure.4. It can be seen that the proposed algorithm has a fail rate in between the fail rates of `CARP` and `CARP10`.

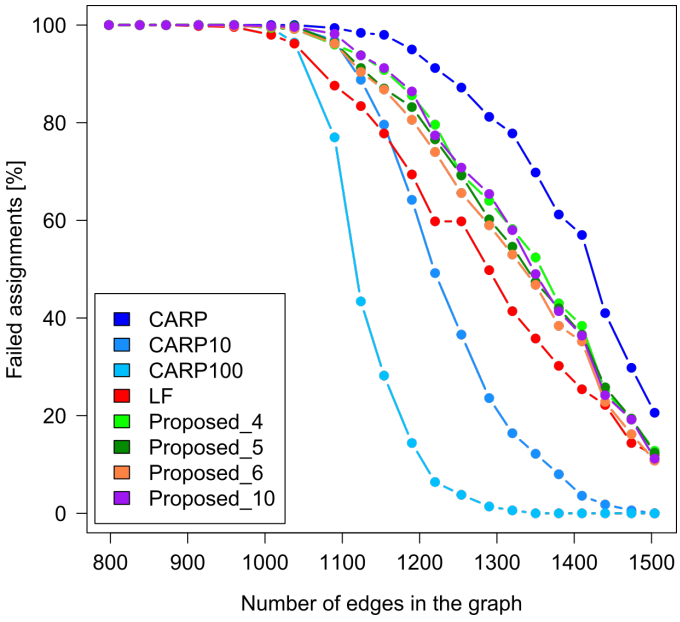

Fig. 4. Algorithms fail rate

The Figure 6Sum of actions performed by all robotsfigure.6

TABLE I
TIMES OF ADDITION FOR 50^{th} AND 100^{th} ROBOT

Algorithm	Time for k^{th} robot [ms]	
	50^{th} robot	100^{th} robot
Proposed_4	3.23	7.6
Proposed_5	10.86	19.48
Proposed_6	50.58	75.81
Proposed_10	43.84	54.10
CARP	0.63	1.60
CARP10	5.29	12.15
CARP100	52.08	126.57
LF	0.85	2.59

shows the average overall quality of the plan from the first experiment measured as the total number of actions of all robots. The graph shows that all versions of the proposed algorithm perform similarly to each other with `Proposed_10` having the best results. Compared to the basic CARP algorithm the proposed algorithm has up to 65 less number of total actions performed across all robots in `Proposed_6` variant (on the 10th map) and up to 35 fewer actions for `Proposed_6` and `Proposed_10` on the original grid map. Moreover, all variants of the proposed algorithm outperform `CARP10` and are at least comparable to `CARP100` which is much more time-consuming. It can also be noticed that `LF` generates worst results. It is not much surprising as it was designed to optimize a makespan.

The results of the second experiment are presented in Fig. 7Time to plan k^{th} robotfigure.7 which shows the required time to find a trajectory for k-th robot considering trajectories of robots $1 \dots k-1$ are already planned. It is evident that `CARP100` is the slowest of all tested algorithms with `Proposed_6` as the second slowest. It is worth noticing that for `Proposed_6` the time of testing all permutations of the neighborhood took longer than replanning of the rest of the robots. The table ITimes of addition for 50^{th} and 100^{th} robottable.1 shows the actual times to plan 50th and 100th robot in this experiment.

From the results as a whole we can see that `Proposed_4` and `Proposed_5` show the best ratio between the quality of the found plan and the time required to compute it. Additionally, if the time requirements for planning are not as tight, it is possible to run a more demanding version of the algorithm to increase the quality of the solution. To increase the overall success rate of the algorithm, it is possible to combine the proposed algorithm with a version that has a higher success rate, such as `CARP100` or even higher in case the proposed algorithm does not find a solution.

V. CONCLUSION

In this paper, we present a novel algorithm for planning in automated warehouses that consider specific requirements

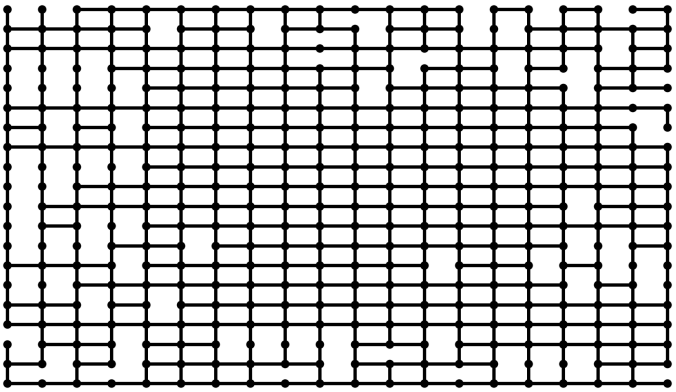

Fig. 5. Example of an experimental map

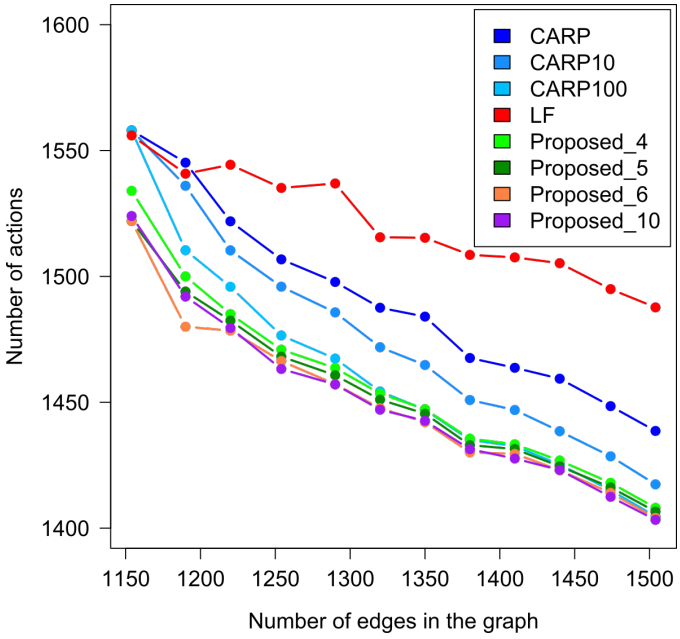

Fig. 6. Sum of actions performed by all robots

in these environments for sequential addition of robots into the system while optimizing the total number of actions all robots must perform. The proposed algorithm employs the standard CARP algorithm for the planning of individual robots, which is one of the best practical algorithms nowadays. The experimental results show that the proposed approach finds better solutions than the original CARP algorithm after several random shuffles of the robots' priorities while requiring significantly less computational time for adding individual robots into the system. Moreover, it is much faster than `CARP100` which produces similar results.

The only drawback of the algorithm is the success rate of finding the solution, and thus the future work should focus on its improvement. One approach could be the use of deep learning to better determine either robot priorities or the set of robots mostly influencing the planned robots which need to be replanned. Another route of improvement would be to

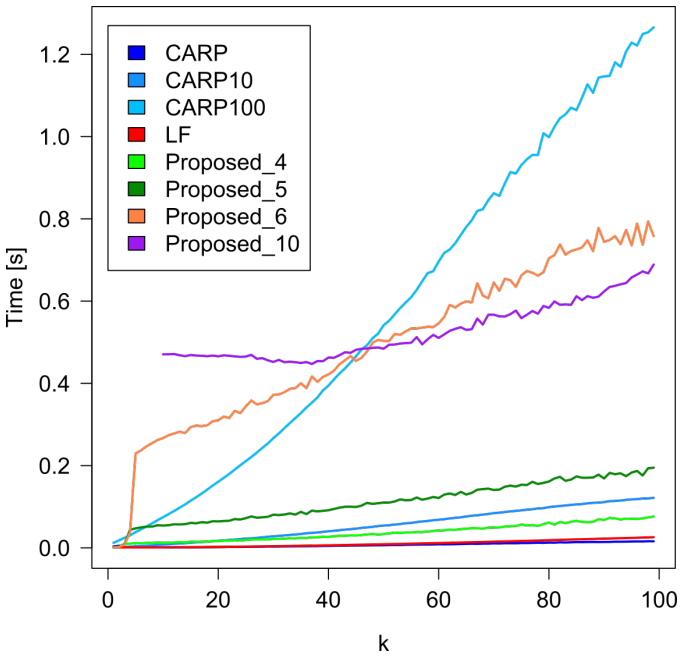

Fig. 7. Time to plan k^{th} robot

adapt the system for the use of heterogeneous robots and more importantly humans, who need to have the highest priority to have their trajectories planned to minimize the time humans spend in the warehouse to eliminate the risk of injury.

ACKNOWLEDGEMENT

This work has been supported by the European Union's Horizon 2020 research and innovation programme under grant agreement No 688117, by the Technology Agency of the Czech Republic under the project no. TE01020197 "Centre for Applied Cybernetics", and by the European Regional Development Fund under the project Robotics for Industry 4.0 (reg. no. CZ.02.1.01/0.0/0.0/15 003/0000470). The work of Jakub Hvězda was also supported by the Grant Agency of the Czech Technical University in Prague, grant No. SGS18/206/OHK3/3T/37.

REFERENCES

[1] A. W. ter Mors, C. Witteveen, J. Zutt, and F. A. Kuipers, "Context-aware route planning," in *Multiagent System Technologies*, J. Dix and C. Witteveen, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 138–149.

[5] J. Hopcroft, J. Schwartz, and M. Sharir, "On the Complexity of Motion Planning for Multiple Independent Objects; PSPACE- Hardness of the "Warehouseman's Problem"," *The International Journal of Robotics Research*, vol. 3, no. 4, pp. 76–88, Dec. 1984.

[2] B. Nagelvoort, R. van Welie, P. van den Brink, A. Weening, and J. Abraham, "European b2c e-commerce report 2014," 2014. [Online]. Available: http://www.ecommerce-europe.eu/facts-figures

[3] W. Matthews and S. Dawson, "The shed of the future e-commerce: its impact on warehouses," 2014. [Online]. Available: http://www2.deloitte.com/uk/en/pages/real-estate/articles/shed-of-the-future.html

[4] L. E. Parker, "Path Planning and Motion Coordination in Multiple Mobile Robot Teams," *Encyclopedia of Complexity and System Science*, 2009.

[6] O. Goldreich, "Finding the Shortest Move-Sequence in the Graph-Generalized 15-Puzzle Is NP-Hard," in *Studies in Complexity and Cryptography. Miscellanea on the Interplay between Randomness and Computation - In Collaboration with Lidor Avigad, Mihir Bellare, Zvika Brakerski, Shafi Goldwasser, Shai Halevi, Tali Kaufman, Leonid Levin, Noam Nisan, Dana Ron,*, ser. Lecture Notes in Computer Science, O. Goldreich, Ed. Springer, 2011, vol. 6650, pp. 1–5.

[7] J.-C. Latombe, "Robot Motion Planning," Jul. 1991.

[8] S. M. Lavalle, "Rapidly-Exploring Random Trees: A New Tool for Path Planning," *Technical Report (Computer Science Deptartment, Iowa State University)*, vol. 11, 1998.

[9] M. R. Ryan, "Exploiting Subgraph Structure in Multi-Robot Path Planning," *Journal of Artificial Intelligence Research*, pp. 497–542, 2008.

[10] S. LaValle and S. Hutchinson, "Optimal motion planning for multiple robots having independent goals," *IEEE Transactions on Robotics and Automation*, vol. 14, no. 6, pp. 912–925, 1998.

[11] T. Simeon, S. Leroy, and J.-P. Lauumond, "Path coordination for multiple mobile robots: a resolution-complete algorithm," *IEEE Transactions on Robotics and Automation*, vol. 18, no. 1, pp. 42–49, 2002.

[12] J. van den Berg and M. Overmars, "Prioritized motion planning for multiple robots," in *2005 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2005, pp. 430–435.

[13] M. Bennewitz, W. Burgard, and S. Thrun, "Optimizing schedules for prioritized path planning of multi-robot systems," in *Robotics and Automation, 2001. Proceedings 2001 ICRA. IEEE International Conference on*, vol. 1, 2001, pp. 271 – 276 vol.1.

[14] M. Cap, P. Novak, A. Kleiner, M. Selecky, and M. Pechoucek, "Prioritized Planning Algorithms for Trajectory Coordination of Multiple Mobile Robots," *IEEE Transactions on Automation Science and Engineering*, vol. Special Is, 2015.

[15] E. Gawrilow, E. Köhler, R. H. Möhring, and B. Stenzel, *Dynamic Routing of Automated Guided Vehicles in Real-time*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 165–177.

[16] W. Wang and W.-B. Goh, "Multi-robot path planning with the spatio-temporal A* algorithm and its variants," in *Advanced Agent Technology*, F. Dechesne, H. Hattori, A. ter Mors, J. M. Such, D. Weyns, and F. Dignum, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 313–329.

[17] T. Zhang, M. Ding, B. Wang, and Q. Chen, "Conflict-free time-based trajectory planning for aircraft taxi automation with refined taxiway modeling," vol. 50, 07 2015.

[18] A. ter Mors, C. Witteveen, C. Ipema, F. de Nijs, and T. Tsiourakis, "Empirical evaluation of multi-agent routing approaches," in *2012 IEEE/WIC/ACM International Conferences on Web Intelligence and Intelligent Agent Technology*, vol. 2, Dec 2012, pp. 305–309.

[19] J. Y. Yen, "Finding the k shortest loopless paths in a network," *Management Science*, vol. 17, no. 11, pp. 712–716, 1971.

[20] J. Suurballe, "Disjoint paths in a network," vol. 4, pp. 125 – 145, 01 1974.

[21] A. ter Mors, "Evaluating heuristics for prioritizing context-aware route planning agents," in *2011 International Conference on Networking, Sensing and Control*, April 2011, pp. 127–132.