An optimization based algorithm for conflict-free navigation of autonomous guided vehicles

B. Csutak[1,2], T. Péni[2], G. Szederkényi[1,2]
[1]Faculty of Information Technology and Bionics, Pázmány Péter Catholic University, Práter 50/a, H-1083 Budapest, Hungary (e-mail: csutak.balazs@hallgato.ppke.hu,szederkenyi@itk.ppke.hu)
[2]Systems and Control Laboratory, Institute for Computer Science and Control (MTA SZTAKI), Hungarian Academy of Sciences, Kende 13-17, H-1111 Budapest, Hungary

Abstract: A novel optimization based route design approach for autonomous guided vehicle transport systems is presented in this paper, followed by the analysis and implementation of an algorithm capable to provide suboptimal solutions for route planning problems in real time providing conflict-, collision- and deadlock-free navigation by design.

Introduction

Optimal route planning based on transport demands is an intensively investigated topic in several engineering fields. Depending on the applied model and assumptions, the computational complexity of such task moves on a wide scale. Route planning problems are commonly modeled as optimization problems, which can indeed give us an optimal solution, but scale badly as the size of the map or the number of agents increases. This means that the real time operation of such methods is often non-realistic due to the need of re-planning.

In [1], a mixed integer optimization based distributed route-planning method is proposed for multiple mobile robots using Dijkstra's algorithm with a special cost function considered to manage vehicle interdependence. The idea of routing with high resource utilization appears in [2, 3], where the deadlock avoidance is addressed, but lacks the ability to handle routes that became unrealizable meanwhile. In [4], a similar routing strategy is introduced for airport taxiways, though it works with different assumptions and optimization goals.

In this line of research, an efficient dynamic real-time algorithm is proposed in [5, 6, 7], where time windows and resource reservation are used to ensure conflict-free navigation by design.

Following the ideas presented in [5, 6, 7], we investigate optimal route planning for multiple types of automated guided vehicles in a microscopic routing environment, where the size of the vehicles in the system is comparable to the size of the underlying network. For this reason, the route planning algorithm should be prepared to avoid collisions and handle congestion and even deadlock problems. Moreover, the algorithm should be able to address additional arising problems, such as vehicles unable to follow their previously planned routes.

As for the optimization, we are trying to find a solution for two common optimization tasks: the *on-line shortest dynamic disjoint path problem* (OSDDPP), and the *on-line quickest disjoint path problem* (OQDPP).

This research is motivated by its possible applications in automated guided vehicle (AGV) routing systems, mostly aimed for industrial application in a research-oriented experimental factory cell in Győr.

Dynamic routing

Formal problem statement

To present the problem in a formal way, following the author of [6], we model the routing environment with a graph $G = (V, E)$ with nodes $V = \{1, 2, ..., N\}$ and edges $E = \{(v_1, v_2, l) | 1 \leq v_1, v_2 \leq N\}$. The graph is directed, and has no multiple or loop edges. The weight of the edges (representing length) is denoted by l. Agents can have different traversal times, based on their maximal speeds.

Transportation tasks are continuously arriving for the agents, and are assigned to the vehicles by a higher level dispatching system.

Definition 1. *A* request *is a tuple $r = (s, t, \theta)$, where s is the source node (from where the agent starts), t is the target node, and θ is the earliest time, when execution of the requests can begin.*

Definition 2. *A* dynamic path *in a graph G is defined as a sequence*

$$P = ((v_0, \theta_0), (v_1, \theta_1), ..., (v_k, \theta_k))$$

of $v_1, ...v_k$ nodes and $\theta_1, ..., \theta_k$ timestamps. Timestamp θ_i is called a reservation *of node v_i, i.e. it constitutes the earliest time when node v_i can be entered. Similarly, interval $(\theta_{i-1}\theta_i)$ is called a* reservation *of the*

edge between v_{i-1} and v_i. The duration *of a dynamic path is defined as $\Delta p = \theta_k - \theta_0$.*

The aim of the algorithm is to compute a set of *disjoint* dynamic paths (having no overlapping time intervals between reservation times of the contained edges) in order to serve the dispatched requests, while minimizing specific cost functions.

Definition 3. *The Online Shortest Dynamic Disjoint Path Problem is defined as follows: Being given a sequence of requests $(s_i, t_i, \theta_i), i = 1, \ldots, k$ find a sequence of disjoint paths $P_1, ..., P_n$, for which $\sum \Delta p_i$ is minimal.*

Definition 4. *The Online Quickest Disjoint Path Problem is defined as follows: Being given a sequence of requests $(s_i, t_i, \theta_i), i = 1..k$ find a sequence of disjoint paths $P_1, ..., P_n$ with minimal maximum completion time over all paths (so that $\max_{i=1...n} \theta_i$ is minimal)*

These problems do not have a common solution, in practice however, the same suboptimal algorithm turns out to be suitable for both cases. It must be noted as well, that finding the optimal solution is not possible due to the continuously arriving requests.

The routing algorithm

Following the ideas presented in [5, 6, 7], we propose an improved greedy algorithm, which, instead of minimizing the overall cost function of the system, it focuses on minimizing the route completion time for the individual agents as the requests arrive, without disturbing the routes already computed.

To achieve this behavior, the algorithm introduces time windows for the graph edges, so agents can reserve all the edges in their path at the very beginning. In this way, the unnecessary waiting or deadlocks can be completely avoided, as the planning algorithm considers these reservations, and looks for the quickest route.

Formally, this goal can be described as follows:

Definition 5. *The Quickest Path Problem with Time Windows: It is given a graph $G = (V, E)$, a set of time windows for the edges, a request $r = (s, t, \theta)$ and an agent in s. Compute a dynamic path with minimal completion time, which uses the edges of the graph in the free time windows only.*

For this task, an algorithm is given in [6], which resembles Dijkstra's simple route planning algorithm, but instead of a single cost value being stored for a graph node, it is based on multiple labels $(a, b, ...)$ being assigned to edges, representing the time intervals, in which the agent can arrive to the respective edge. The algorithm is initialized with labels (t, ∞) on the edges having s as tail, and they are expanded to neighbouring labels iteratively (like in Dijkstra's algorithm), taking in consideration the free time windows.

Handling agent delays

As the system is aimed to be suitable in a real environment, practical considerations must be made. Due to the nature of such environments, there are several factors that can influence the routes planned, and cause already planned routes to become impossible to complete.

Minor time differences, arising from uncertainties in vehicle control can be easily solved, by reserving an interval slightly longer than necessary for the traversed edges. When this safety interval is not enough, and an agent can not free the resource until it would be obliged to, re-planning must take place.

In case of a severe latency, we decided to stop all vehicles in the system, and then all agents are required to re-plan their routes sequentially one after another taking into consideration the eventual changes in the graph (eg. a broken vehicle permanently blocking an edge). As a consequence, all edges where the agents are actually located (and then stopped) should be reserved until they can leave that edge. Though the algorithm can handle the computations in real time, a difficulty is that the agent, who is planning first cannot foresee when will the edges be released where the other agents are still waiting for the possibility to re-plan their routes. On the other hand, the agent, who is planning last may face the problem that all neighbouring edges are already reserved, thus it cannot leave its position until a certain time. Generally, it is not straightforward how to determine the length of the time window for each initially occupied edge. This interval must be chosen carefully, since, if we consider a short time window, the agent planning last might not be able to leave the edge until the end of this time window, and thus causing another delay.

In our solution, a simple heuristics was applied: for all agents stopped at time T_0, a reservation for the interval $(T_0, T_0 + \Delta T)$ was made, and the route planning system assumed, that the agents can leave their position in that interval. This behavior was further aided by initializ-

ing the route planning algorithms with starting labels containing this interval, so all agents try to leave their place as soon as possible.

If an agent is unable to leave its location until the and of this time window, that delay is handled by another severe latency, causing a repeated recalculation of all routes in the system (eventually with a higher ΔT value).

Test cases

The simulation environment

To extensively test the implemented framework system for verification, and check the usability of the implemented algorithm in the scenarios needed, an exact model of the experimental factory cell from Győr was realised in the simulation system. This process involved loading and transforming the data acquired from measurements of the place, so that a three dimensional representation of walls and objects would appear. Next, a directed graph based on the loaded floorplan was created, followed by the generation of nodes and edges in the air, suitable for quadcopters only. Finally, after generating a planner graph and defining some parking places / workstations (nodes, from which and to which transportation requests are arriving), two ground AGVs and two quadcopters were loaded and launched. The floorplan of the graph, together with the planned routes, can be observed in Figure 1.

The experiment was carried out using MATLAB 2018b, on a Dell Vostro 5471, having processor model Intel Core i7-8550U (4 cores, 8 threads, up to 4GHz) and 8 GB RAM. The planner graph resulting from the scene had 158 vertices and 506 edges.

Loading AGVs and generating requests

The AGVs were loaded to the following positions: two ground AGVs to nodes 9 and 18, and two quadcopters to nodes 1 and 16. In this order, the agents had the targets 18, 16, 9, 20. The route planning took place in order 16, 18, 9, 1, where the numbers refer to the departure nodes of the corresponding agents. The routes calculated and followed by the agents can be seen in Figure 1. One can observe that the agent starting from node nr. 18 has chosen a route $\{18, 13, 17, 14, 16\}$ instead of the spatially shorter one $\{18, 13, 12, 14, 16\}$ in order to avoid the interference with the already scheduled route starting from node nr. 16. As for the performance of the algorithm, all route planning operations took place in a time less than 0.1 seconds.

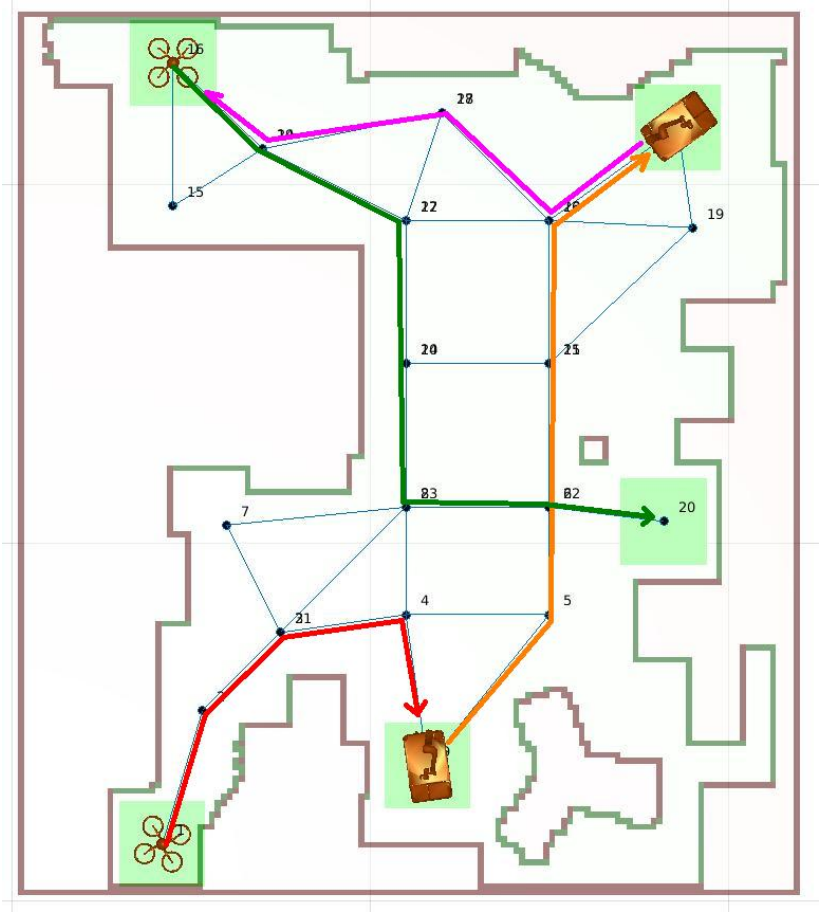

Figure 1: Routes planned using the algorithm

Further test cases

To assess the performance in a more complex scenario, an extended version of the experimental factory cell (see Fig. 2) was created (including as part the original one as well). The underlying graph has above 200 nodes, resulting in a planner graph with over 600 nodes 1000 edges. There were 10 AGVs (4 ground vehicles and 6 quadcopters) loaded, each of them starting randomly from one of the 11 workstations / parking places. Targets for the agents were generated randomly. During the simulation, as soon as one of the agents reached its target, a new one was randomly assigned to it.

The algorithm had a decent performance, computation times for a single route remaining below 1 second in any circumstances, which could be further reduced by some fine-tuning of the implementation.

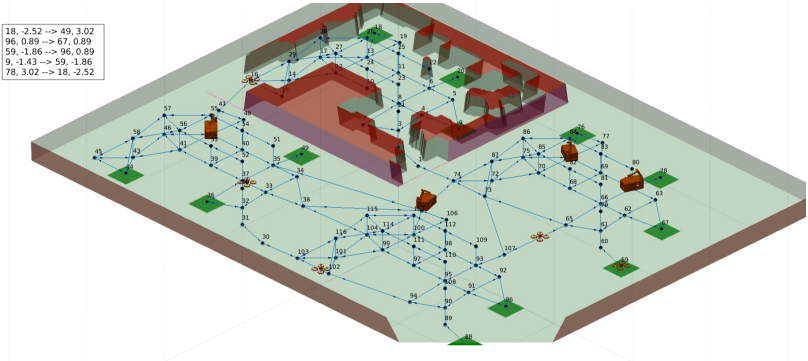

Figure 2: Extended experimental factory cell

Movement of the AGVs for all the test cases can be seen in the demonstration videos available online at `https://drive.google.com/open?id=1-6iPlFZdlAf10fzSWVcLuRQq5ybuvCyd`.

Summary

In this work, we briefly presented the operating principle of an algorithm, capable of providing online disjoint autonomous vehicle route planning for multiple type of AGVs moving in a microscopic routing environment. The algorithm was implemented and thoroughly tested, together with our method for handling serious delays of the agents.

Acknowledgements

B. Csutak gratefully acknowledge the support of the New National Excellence Program scholarship (ÚNKP-18-1-I-PPKE-46). This work was supported in part by the grant EFOP-3.6.2-16-2017-00013.

References

[1] T. Nishi, M. Ando, and M. Konishi. Distributed route planning for multiple mobile robots using an augmented lagrangian decomposition and coordination technique. *IEEE Transactions on Robotics*, 21(6):1191–1200, Dec 2005.

[2] Thomas Lienert and Johannes Fottner. No more deadlocks - applying the time window routing method to shuttle systems. In *ECMS*, 2017.

[3] Kaspar Schüpbach and Rico Zenklusen. An adaptive routing approach for personal rapid transit. *Mathematical Methods of Operations Research*, 77(3):371–380, Jun 2013.

[4] Stefan Ravizza, Jason A. D. Atkin, and Edmund K. Burke. A more realistic approach for airport ground movement optimisation with stand holding. *Journal of Scheduling*, 17(5):507–520, Oct 2014.

[5] Rolf H. Möhring, Ekkehard Köhler, Ewgenij Gawrilow, and Björn Stenzel. Conflict-free real-time AGV routing. In Hein Fleuren, Dick den Hertog, and Peter Kort, editors, *Operations Research Proceedings 2004*, pages 18–24, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg.

[6] Björn Stenzel. *Online disjoint vehicle routing with application to AGV routing*. PhD thesis, Technical University of Berlin, 2008.

[7] Ewgenij Gawrilow, Ekkehard Köhler, Rolf H. Möhring, and Björn Stenzel. *Dynamic routing of automated guided vehicles in real-time*, pages 165–177. Springer Berlin Heidelberg, Berlin, Heidelberg, 2008.