Conflict-free Real-time AGV Routing

Rolf H. Möhring, Ekkehard Köhler, Ewgenij Gawrilow, and Björn Stenzel

Technische Universität Berlin, Institut für Mathematik, MA 6-1,
Straße des 17. Juni 136, 10623 Berlin, Germany
{moehring,ekoehler,gawrilow,stenzel}@math.tu-berlin.de

Abstract. We present an algorithm for the problem of routing Automated Guided Vehicles (AGVs) in an automated logistic system. The algorithm avoids collisions, deadlocks and livelocks already at the time of route computation (conflict-free routing). After a preprocessing step the real-time computation for each request consists of the determination of a shortest path with time-windows and a following readjustment of these time-windows. Both is done in polynomial-time. Using goal-oriented search we get computation times which are appropriate for real-time routing. Additionally, in comparison to a static routing approach, used in Container Terminal Altenwerder (CTA) at Hamburg Harbour, our algorithm had an explicit advantage.

1 Introduction

Nowadays automation in logistic systems is very popular. In such an automated logistic system Automated Guided Vehicles (AGVs) are used for transportation tasks and the control of these AGVs is the key to an efficient transportation system. Usually, the aim is to maximize the throughput.

Here, control means computation of routes (routing) on the one hand and collision avoidance on the other hand. The assignment of transportation tasks to AGVs is not part of this work.

Various aspects of the considered problem are of importance. The routes must be computed in appropriate time (for a real-time computation) with respect to the physical properties of the AGVs. The collision avoidance particularly has to deal with the dimensions of the AGVs. In our approach collisions are avoided already at the time of route computaion. This is called conflict-free routing.

One of the first paper on routing (free-ranging) AGVs without causing collisions was done by Broadbent et al. [3] in 1987. Krishnamatury, Batta and Karwan [9] discussed the problem for the special case when all requests are known right from the beginning.

We consider the online problem where requests appear sequentially and one must answer each request without having any information about later arriving requests. We extend the approaches of Huang, Palekar and Kapoor [7] and Kim and Tanchoco [8], respectively. In particular, we take physical properties of the AGVs into consideration in a more exact and flexible way and present an efficient algorithm for the problem.

2 The Model

Let $G = (V, A)$ be a directed graph which represents the lanes of the automated transportation system and let $\tau(a)$ be the transit time on arc a. Then an online routing algorithm has to deal with a sequence $\sigma = r_1, \ldots, r_n$ of requests. Each request $r_j = (s_j, t_j, \theta_j)$ represents a task with start node s_j and end node t_j while θ_j denotes the desired starting time.

2.1 Problems with static routing approaches

In static approaches for this problem one computes a route without taking time dependencies into consideration. This can be done by standard shortest path algorithms, i.e., the Dijkstra algorithm (see [4]). In such an approach congestion can be considered by additionally using load dependent arc costs (see [1]). The arising routes are not conflict-free and therefore one needs an additional conflict management at real-time. To guarantee that there are no collisions the moving AGVs must allocate the area they want to use next.

The advantage of this approach is clear: it is easy to implement a fast routing algorithm. But the disadvantages of the described collision avoidance, namely the appearance of deadlocks and livelocks (see Fig. 1), have an enormous effect on the performance of the system.

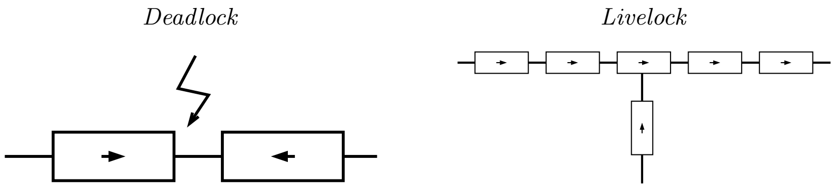

Fig. 1. Deadlocks and Livelocks

Deadlocks appear if two or more AGVs wish to allocate the same area. None is able to continue its route and the system is blocked. Livelock is the generic term for situations where an AGV is blocked repeatedly by other AGVs without having the possibility of allocating the area which is next on its route.

2.2 Conflict-free routing

In order to avoid the problems of the simple model given in Section 2.1, we compute conflict-free routes because in a conflict-free approach there is no need for an additional collision avoidance.

There are two key ingredients which must be considered in that approach. On the one hand, one has to deal with the physical dimensions of the AGVs because they usually have to claim several arcs in the directed graph at the same time. On the other hand, the approach has to be time-dependent (dynamic).

To avoid conflicts we use polygons $P(a)$ for each arc a which describe the blocked area when an AGV (the center of an AGV) is on arc a. Thus, it is prohibited to use two arcs at the same time if the corresponding polygons intersect each other. For each arc a, this leads to a set $\mathit{confl}(a)$ of so called geographic-dependent arcs which should not be used at the same time.

The time-dependent behavior is modelled by time intervals. If an AGV travels over an arc a during the interval $[t_1, t_2]$, all geographic-dependent arcs are blocked from t_1 to t_2. This leads to time-windows for each arc of the directed graph G. These time-windows represent the times when arc a is free. After routing a request one has to readjust these time-windows according to the used arcs and their geographic-dependent arcs. If this is done consequently, one does not have to take care of the AGV dimension in route computation since it is already considered.

3 The Algorithm

The algorithm consists of a preprocessing step and (for each request) a route computation followed by a readjustment of the appropriate time-windows.

3.1 Preprocessing

In a preprocessing step all polygons $P(a)$ are compared pairwise. If two polygons $P(a)$ and $P(b)$, for arcs $a, b \in A(G)$, intersect, a is assigned to $\mathit{confl}(b)$ and b is added to $\mathit{confl}(a)$.

Additionally, in this preprocessing step the behavior of the AGVs is modelled by a list $OUT(a)$ of arcs, containing the set of arcs, which are allowed to be used after arc a respecting the physical properties of the AGV.

3.2 Route computation: shortest paths with time-windows

As pointed out in Section 2.2, the route computation can be done in an idealized model where the dimension of the AGV is not considered explicitly. One just has to compute a route for an infinitesimal point which represents the center of the AGV.

This problem is known as Shortest Path Problem with Time-Windows (SPPTW) [5,10]: Given are a graph G, a source node s, a destination node t, a start time θ, transit times τ_a, costs c_a and a set of time-windows $\mathcal{F}_a$ on each arc a. The aim is to compute a shortest path (concerning costs c) respecting the given time-windows. Here "respecting" means that AGVs wait and traverse the corresponding arc only during (open) time-windows.

The SPPTW is $\mathcal{NP}$-hard. The hardness can be shown by reduction of the of the Constrained Shortest Path Problem (CSPP, see [2]).[1]

Our algorithm for this problem is a generalized arc-based Dijkstra algorithm which deals with labels. A label $L = (a_L, d_L, I_L, pred_L)$ on an arc a_L consists of a distance value d_L, a predecessor $pred_L$ and a time interval I_L. Each label L represents a path from start node s to the tail of a_L, whereas d_L contains the overall transit time (travelling and waiting together) and the label interval $I_L = (A_L, B_L)$ represents the possible arrival times at arc a_L (at the tail of a_L). $pred_L$ is the predecessor of a_L on that path.

We define an order for these labels.

Definition 1 *A label L dominates a label L' if and only if*

$$d_L \leq d_{L'} \quad \textit{and} \quad I_{L'} \subseteq I_L.$$

The labels are stored in a priority queue H (a binary heap for example). The generalized arc-based Dijkstra algorithm works as follows.

- **Initialization.**
  Create a label $L = (a, 0, (\theta, \infty), nil)$ for all outgoing arcs a of s and add them to the priority queue H.
- **Loop.**
  Take the label L with lowest distance value d_L from H. If there is no label left in the queue, notify that there is no path from s to t. If t is the tail of a_L, write the corresponding path to output.
  - **For** each time-window on a_L.
    * **Label Expansion.**
      Try to expand the label interval along a_L through the current time-window (new label interval should be as large as possible, see Fig. 2), add the costs $c(a_L)$ to the distance value and set the predecessor. If there is no possible expansion, consider the next time-window.
    * **Dominance Test.**
      Add the new label to each outgoing arc a in $OUT(a_L)$, if it is not dominated by any other label on a.

In general, the algorithm cannot be executed in polynomial-time, unless $\mathcal{P} \neq \mathcal{NP}$. But in our setting we have special costs on the arcs: the transit times of the paths including the corresponding waiting times. In that case we get a polynomial-time algorithm, because costs (distance values) correlate with the lower bounds of the label intervals.

Theorem 2 *In the case of transit times (including waiting times) as cost function, the described generalized arc-based Dijkstra algorithm solves the SPPTW in polynomial-time.*

[1] The instance of the SPPTW is constructed by placing time-windows $[0, R]$ at each arc while R denotes the resource constraint in the CSPP instance.

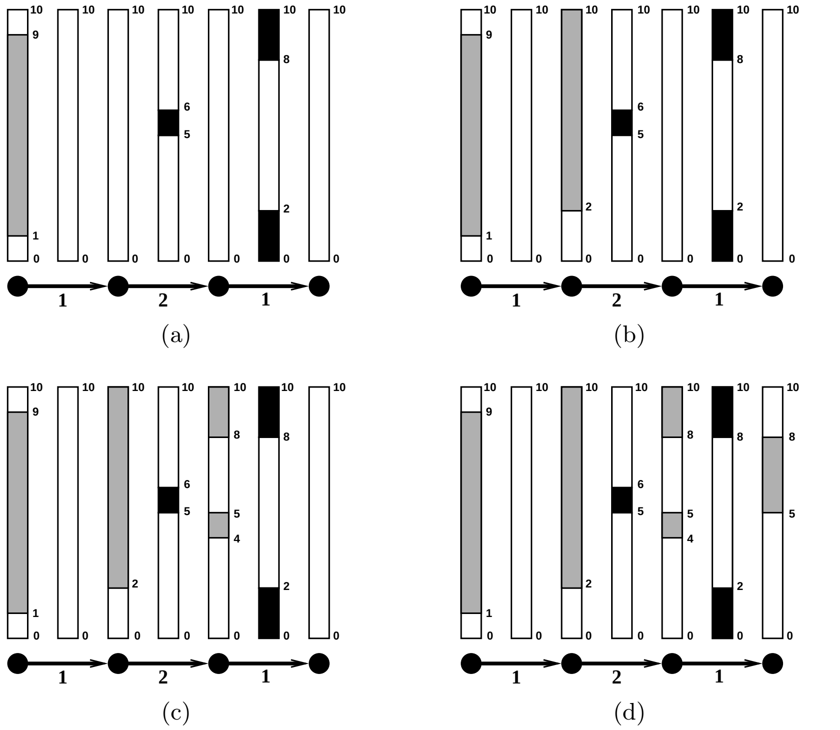

Fig. 2. Label Expansion. The label intervals are represented by grey bars (nodes). The blockings are colored black (arcs). The white intervals between these blockings are the time-windows. The figures (a) to (d) show the successive expansion of the label intervals.

Proof. The algorithm computes all required paths since the expansion of the label intervals is maximal and no required path (label) will be dominated.

Consider the correlation between costs and transit times: they differ just by an constant additive, namely the starting time. Thus, for two labels that are expanded via the same time-window the distance value controls the dominance relation. That means, that in this case a label dominates another if and only if it has a lower distance value.

Therefore, the number of labels on an arc a is bounded by the number of time-windows on all ingoing arcs. Thus, the number of iterations in each loop is either bounded by the number of arcs or bounded by the number of time-windows. Hence, the algorithm terminates in polynomial-time with a shortest path (or the notification that there is no path). □

We can directly conclude for the SPPTW:

Corollar 3 *The SPPTW with transit time (including waiting times) as cost function can be solved in polynomial-time.*

For additional acceleration of the algorithm we use goal-oriented search [6].

3.3 Readjustment of the time-windows

For each arc a of a computed route we consider all geographic dependent arcs in *confl*(a) during the time interval $[t_1, t_2]$ representing the transit over a. Then we verify for each time-window on an arc of *confl(a)*, whether the time-window and $[t_1, t_2]$ overlap. Depending on that, the time-window is shortened, erased or left unchanged.

4 Computational Results

Two questions certainly arise concerning the presented conflict-free approach. Is the approach really better than the static one? Is the algorithm suitable for real-time computation? Concerning these questions we present some results for test instances (scenarios) on a graph with about 30.000 arcs.

4.1 Comparison with a static approach

In order to measure the performance of the computed routes we consider the sum of all transit times (all requests). We compare these overall transit times with a static approach used in Container Terminal Altenwerder (CTA) at Hamburg Harbour.

The comparison shows that the conflict-free approach is superior to the static one. The strongest dominance of our algorithm is achieved for scenarios with heavy traffic (many AGVs) which clearly indicates the potential of the conflict-free approach.

4.2 Computation times

Besides obtaining a better solution in comparison with other approaches, a real-time computation requires fast answers. The computation times[2] in Table 1 show that our algorithm is able to provide this. The table is divided into computation times for determining a shortest path with time-windows (search) and computation times for readjusting the time-windows (blocking).

On average, the computation (in both cases) take not more than some hundredth of a second. The entire real-time computation (search and blocking together) in all tested cases takes less than half a second which is suitable for this real-time computation.

[2] Hardware: AMD-Athlon 2100+ (1,7 Mhz) with 512 MB RAM.

Table 1. Computational times (in sec.)

Scenarios	Search		Blocking	
	maximal	∅	maximal	∅
1A (15 AGVs)	0.21	0.018	0.19	0.026
2A (15 AGVs)	0.14	0.020	0.17	0.038
3A (20 AGVs)	0.22	0.021	0.20	0.030
4A (44 AGVs)	0.21	0.019	0.19	0.022
1B (32 AGVs)	0.18	0.022	0.18	0.024
2B (38 AGVs)	0.28	0.021	0.17	0.025
1C (22 AGVs)	0.09	0.021	0.13	0.042
2C (44 AGVs)	0.17	0.023	0.09	0.029
3C (48 AGVs)	0.17	0.022	0.12	0.040

References

1. Aspnes, J. et al. (1997) On-line routing of virtual circuits with applications to load balancing and machine scheduling. Journal of the ACM 44, 486–504
2. Beasley, J. E., Christofides, N. (1989) An algorithm for the resource constrained shortest path problem. Networks 19, 379–394
3. Broadbent, A. J. et al. (1987) Free-ranging agv and scheduling system. In Automated Guided Vehicle Systems, 301–309
4. Cormen, T. H., Leiserson, Ch. E., Rivest, R. L. (1990) Introduction to Algorithms. MIT Press, Cambridge, Massachusetts
5. Desrosiers et al. (1986) Methods for routing with time windows. European Journal of Operations Research 23, 236–245
6. Hart, P., Nilsson, N., Raphael, B. (1968) A formal basis for the heuristic determination of minimum cost paths. In IEEE Transactions on Systems, Science and Cybernetics SCC-4, 100–107
7. Huang, J., Palekar, U. S., Kapoor, S. (1993) A labeling algorithm for the navigation of automated guided vehicles. Journal of engineering for industry 115, 315–321
8. Kim, Ch. W., Tanchoco, J. M. A. (1991) Conflict-free shortest-time bidirectional agv routeing. International Journal of Production Research 29(12), 2377–2391
9. Krishnamurthy, N., Batta, R., Karwan, M. (1993) Developing conflict-free routes for automated guided vehicles. Operations Research 41, 1077–1090
10. Sancho, N. G. F. (1994) Shortest path problems with time windows on nodes and arcs. Journal of mathematical analysis and applications 186, 643–648