

Routing Driverless Transport Vehicles in Car Assembly with Answer Set Programming

Martin Gebser, Philipp Obermeier, Torsten Schaub

University of Potsdam, Germany

Michel Ratsch-Heitmann, Mario Runge

Mercedes-Benz Ludwigsfelde GmbH, Germany

submitted [n/a]; revised [n/a]; accepted [n/a]

Abstract

Automated storage and retrieval systems are principal components of modern production and warehouse facilities. In particular, automated guided vehicles nowadays substitute human-operated pallet trucks in transporting production materials between storage locations and assembly stations. While low-level control systems take care of navigating such driverless vehicles along programmed routes and avoid collisions even under unforeseen circumstances, in the common case of multiple vehicles sharing the same operation area, the problem remains how to set up routes such that a collection of transport tasks is accomplished most effectively. We address this prevalent problem in the context of car assembly at Mercedes-Benz Ludwigsfelde GmbH, a large-scale producer of commercial vehicles, where routes for automated guided vehicles used in the production process have traditionally been hand-coded by human engineers. Such ad-hoc methods may suffice as long as a running production process remains in place, while any change in the factory layout or production targets necessitates tedious manual reconfiguration, not to mention the missing portability between different production plants. Unlike this, we propose a declarative approach based on Answer Set Programming to optimize the routes taken by automated guided vehicles for accomplishing transport tasks. The advantages include a transparent and executable problem formalization, provable optimality of routes relative to objective criteria, as well as elaboration tolerance towards particular factory layouts and production targets. Moreover, we demonstrate that our approach is efficient enough to deal with the transport tasks evolving in realistic production processes at the car factory of Mercedes-Benz Ludwigsfelde GmbH.

Under consideration for publication in Theory and Practice of Logic Programming (TPLP)

KEYWORDS: automated guided vehicle routing, car assembly operations, answer set programming

1 Introduction

Automated guided vehicles play a key role in modern industries, let it be in warehouses, mines, or as in our case production facilities. Most of the time, however, these vehicles are programmed by human engineers to execute specific tasks. This makes it impossible to quickly reassign tasks in case of breakdowns or to easily react to changing production requirements, not to mention the missing portability between different production plants and factory layouts. The lack of elaboration tolerance does not only lead to high expenditures, but the resulting rigid control also rules out any flexible fleet management.



Fig. 1. Real-world factory layout with transport corridors and directions indicated by arrows

In particular, no conclusions can be drawn about the effectiveness or even optimality of pre-programmed vehicle routes.

In view of these common circumstances, we address the challenge of devising a new control system for automated guided vehicles supplying the assembly lines at the car factory of Mercedes-Benz Ludwigsfelde GmbH, which is expected to be flexible enough to adapt to malfunctions and emerging requirements, and whose quality of operation can be measured and optimized relative to given objectives. To make our specific task more precise, consider the retouched layout of an assembly hall at the production plant of Mercedes-Benz Ludwigsfelde GmbH in Figure 1. The overall goal is to guarantee that all car components are at their designated place next to the assembly line when they are due for installation. The corresponding transport tasks are accomplished by a heterogeneous fleet of automated guided vehicles that fetch the necessary components from several storage areas. For instance, a vehicle may first halt at some storage location to load production material, then move on to an assembly station in need of the material, and from there cart off leftover material to a recycling facility. The sketched task thus involves stopovers at three distinct locations, between which the vehicle must pick a route without getting blocked by others, where dedicated parking spaces are included in the layout to let vehicles make room. Notably, the regular production process runs periodically, so that the given transport tasks have to be repeatedly executed in fixed intervals.

To make the execution of transport tasks more effective, we propose to perform task assignment and vehicle routing by means of Answer Set Programming (ASP; Lifschitz 1999), given its declarative and elaboration tolerant approach to combinatorial multi-objective problem solving. This results in a transparent and easily adjustable problem description along with optimal solutions relative to objective criteria. Regarding performance, it also turns out that our approach is efficient enough to meet the industrial-scale requirements of production processes at the car factory of Mercedes-Benz Ludwigsfelde GmbH.

The paper is structured as follows. In the next section, we start by formalizing automated guided vehicle routing for transport tasks evolving in production processes at the car factory of Mercedes-Benz Ludwigsfelde GmbH. Once this is accomplished, we provide in Section 3 a corresponding ASP encoding, incorporating the hard and soft constraints on solutions. In Section 4, we empirically evaluate our approach on use cases designed to test the practical applicability of control systems for automated guided vehicles. Finally, we conclude the paper with a brief discussion of related work and the achieved results.

2 Problem Formalization

In what follows, we formally describe the automated guided vehicle routing scenarios faced in production processes at the car factory of Mercedes-Benz Ludwigsfelde GmbH, the conditions on their solutions, as well as objective criteria concerning solution quality devised together with production engineers at Mercedes-Benz Ludwigsfelde GmbH.

An automated guided vehicle routing scenario is specified in terms of a directed graph (V, E) , where the nodes V stand for *locations* of interest and the edges $E \subseteq V \times V$ provide *connections* between them, along with a set T of transport *tasks*. Among the nodes in V , we distinguish particular *halt* and *park* nodes, given by $h(V) \subseteq V$ and $p(V) \subseteq V$ such that $h(V) \cap p(V) = \emptyset$ holds. Moreover, each task $t \in T$ has an associated non-empty sequence $\langle s_1, \dots, s_m \rangle$ of *subtasks*, whose elements are halt nodes, i.e., $\{s_1, \dots, s_m\} \subseteq h(V)$, and we write $s(t) = m$ and $t[i] = s_i$, for $1 \leq i \leq m$, to refer to the number of or individual subtasks, respectively, associated with t . In addition, every edge, halt or park node, and task is characterized by some positive integer, denoted by $d(x)$ for $x \in E \cup h(V) \cup p(V) \cup T$, where $d(e)$ expresses the *move duration* for a connection $e \in E$, $d(v)$ provides the *halt* or *park duration* for a location $v \in h(V) \cup p(V)$, and $d(t)$ constitutes the *deadline* for completing a task $t \in T$. Finally, a set C of *vehicles* usable for transport tasks is given together with, for each $c \in C$, an initial location $l(c) \in V$ such that $l(c) \neq l(c')$ when $c' \in C \setminus \{c\}$. The vehicles can accomplish transport tasks by taking connections and halting at the locations of respective subtasks, whose halt durations reflect time spent to operate carried materials (e.g., loading or unloading), while further stops are admitted at park nodes only, included for allowing a vehicle to wait in case its subsequent route is temporarily blocked by others.

Example 1

The scenario depicted (in black) in Figure 2 consists of the nodes $V = \{1, \dots, 7\}$ and edges $E = \{(1, 2), (1, 7), (2, 3), (3, 4), (4, 5), (4, 7), (5, 6), (6, 1), (7, 1), (7, 4)\}$. The halt nodes $h(V) = \{2, 4, 5, 6\}$ are indicated by diamonds in Figure 2, and the park node in $p(V) = \{7\}$ is marked by a square. The two tasks in $T = \{t_1, t_2\}$ include three subtasks each, where the respective sequences of halt nodes, $\langle t_1[1] = 5, t_1[2] = 4, t_1[3] = 2 \rangle$ and $\langle t_2[1] = 6, t_2[2] = 4, t_2[3] = 2 \rangle$, are listed within red or blue labels, respectively, near the subtasks. Similarly, the initial locations $l(c_1) = 1$ and $l(c_2) = 2$ of the vehicles in $C = \{c_1, c_2\}$ are indicated by corresponding labels. While Figure 2 does not display the durations, we assume a uniform move duration $d(e) = 4$ per connection $e \in E$, a halt duration $d(v) = 3$ at each of the four halt nodes $v \in \{2, 4, 5, 6\}$, and a park duration $d(7) = 2$ at the park node 7. Moreover, the deadline for both tasks, t_1 and t_2 , is given by $d(t_1) = d(t_2) = 60$ and also omitted in Figure 2 for better clarity. ■

A solution to a guided vehicle routing scenario comprises a *task assignment* $\alpha : T \rightarrow C$ along with a strict partial *order* \prec on T such that either $t \prec t'$ or $t' \prec t$ holds when $\alpha(t) = \alpha(t')$ for tasks $t \neq t'$. In addition, for each vehicle $c \in C$, it contains a *route* $\pi(c) = \langle r_1, \dots, r_k \rangle$, where $\{r_1, \dots, r_k\} \subseteq E \cup h(V) \cup p(V)$, subject to the following conditions:

1. If $r_i = (v, v')$, for $(v, v') \in E$, or $r_i = v$, for $v \in h(V) \cup p(V)$, holds for some $1 \leq i \leq k$, then we require that $r_{i-1} = (v'', v)$ or $r_{i-1} = v$, where we let $r_0 = l(c)$.
2. For any subtask $t[j]$ of a task $t \in T$ with $\alpha(t) = c$, we define its completion index by $g(t[j]) = \min\{i \mid 1 \leq i \leq k, r_i = t[j], \max(\{g(t[j-1]) \mid 1 < j\} \cup \{g(t'[s(t')]) \mid t' \prec t\}) < i\}$ and require that $g(t[j]) \leq \sum_{1 \leq i \leq g(t[j])} d(r_i) \leq d(t)$. In words, the subtasks of all

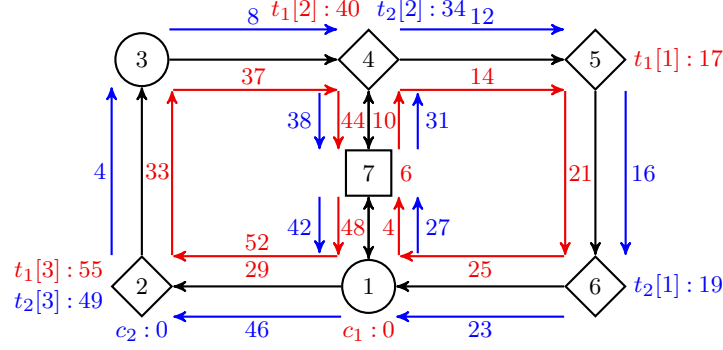


Fig. 2. Optimal routes to accomplish two transport tasks with three subtasks each by two vehicles

tasks assigned to a vehicle c have to be completed in the order given by \prec (as well as sequences of subtasks) within their respective deadlines, where a subtask is completed once the route $\pi(c)$ includes a halt at its location such that all preceding subtasks have been completed before.

3. If $r_i = v$ for a halt node $v \in h(V)$ and $1 \leq i \leq k$, then we require that $g(t[j]) = i$ for some subtask $t[j]$ of a task $t \in T$ with $\alpha(t) = c$. That is, halts may be included in the route of a vehicle c exclusively for completing subtasks of tasks assigned to c .
4. With each $x \in E \cup V$, we associate a set of occupation times defined as $u(c, x) = \{d + \sum_{1 \leq j < i} d(r_j) \mid 1 \leq i \leq k, r_i = x, 1 \leq d \leq d(x)\} \cup \{\sum_{1 \leq j \leq i} d(r_j) \mid 1 \leq i \leq k, r_i \in E, r_i = (v, x)\}$. For any vehicle $c' \in C \setminus \{c\}$, the following requirements check that the routes of c and c' do not both lead to a joint location at the same time, and that c and c' do not meet in between nodes by taking connections in opposite directions: for each location $v \in V$, we require that $u(c, v) \cap u(c', v) = \emptyset$, while $u(c, (v, v')) \cap u(c', (v', v)) = \emptyset$ must hold for each bidirectional connection pair $\{(v, v'), (v', v)\} \subseteq E$.

Example 2

The red and blue arrows and labels in Figure 2 indicate the routes of a solution to the automated guided vehicle routing scenario from Example 1, where the task assignment is given by $\alpha(t_1) = c_1$ and $\alpha(t_2) = c_2$. As no distinct tasks are assigned to the same vehicle, we have that the order relation \prec is empty, while c_1 and c_2 have to complete the subtasks of t_1 or t_2 , respectively, in order. The routes taken by c_1 and c_2 can be traced by considering the times of completing route elements, included in labels along nodes and edges, in increasing order, starting from $c_1 : 0$ and $c_2 : 0$ at the initial locations $l(c_1) = 1$ and $l(c_2) = 2$. For vehicle c_1 , this yields the route $\pi(c_1) = \langle (1, 7), 7, (7, 4), (4, 5), 5, (5, 6), (6, 1), (1, 2), (2, 3), (3, 4), 4, (4, 7), (7, 1), (1, 2), 2 \rangle$, where the four stops at halt or park nodes in $h(V) \cup p(V) = \{2, 4, 5, 6, 7\}$ are of particular interest. In fact, the inclusion of the park node 7 at the second position of $\pi(c_1)$ delays the subsequent move $(7, 4)$ by the park duration $d(7) = 2$ in order to avoid arriving at node 4 at the same time as the other vehicle c_2 . Unlike that, the later stops at the halt nodes 5, 4, and 2 are made to complete the sequence $\langle t_1[1] = 5, t_1[2] = 4, t_1[3] = 2 \rangle$ of subtasks. While $\pi(c_1)$ makes c_1 revisit the halt nodes $t_1[2] = 4$ and $t_1[3] = 2$, a halt would not be admitted on the first arrival because the preceding subtask $t_1[1] = 5$ or $t_1[2] = 4$, respectively, has not yet been

completed. Hence, the halt at 5 comes before returning to and halting at 4, and then in turn at 2 at the very end of the route $\pi(c_1)$. As one can check, the sum of durations over connections as well as halt and park nodes in $\pi(c_1)$ is 55, which means that the task t_1 gets completed within its deadline $d(t_1) = 60$. Regarding the other vehicle c_2 , tracing the blue labels in Figure 2 yields the route $\pi(c_2) = \langle (2, 3), (3, 4), (4, 5), (5, 6), 6, (6, 1), (1, 7), (7, 4), 4, (4, 7), (7, 1), (1, 2), 2 \rangle$ for completing the sequence $\langle t_2[1] = 6, t_2[2] = 4, t_2[3] = 2 \rangle$ of subtasks. Along this route, c_2 visits the halt nodes $t_2[2] = 4$ and $t_2[3] = l(c_2) = 2$ twice, where halts are in both cases made on the second arrival for completing the subtasks of t_2 in order. Moreover, no stop at the park node 7 is needed to avoid meeting the other vehicle c_1 , and the durations of connections and halt nodes in $\pi(c_2)$ sum up to 49, so that also the task t_2 gets completed within its deadline $d(t_2) = 60$. ■

While an automated guided vehicle routing scenario may have plenty feasible solutions, we apply the following objective criteria, below ordered by significance, to distinguish preferred collections of routes $\pi(c_i) = \langle r_{1_i}, \dots, r_{k_i} \rangle$ for the vehicles $c_i \in C$:

1. The *makespan* ms of a solution is the maximum sum of durations over the route elements of some vehicle, i.e.,

$$ms = \max\{\sum_{1 \leq j_i \leq k_i} d(r_{j_i}) \mid c_i \in C\}.$$

2. The *route length* rl of a solution is the sum of durations over elements in the routes of all vehicles, i.e.,

$$rl = \sum_{c_i \in C, 1 \leq j_i \leq k_i} d(r_{j_i}).$$

3. The *crossing number* cn is the number of pairs $(\{c_i, c_{i'}\}, v)$ such that $c_i \in C$, $c_{i'} \in C \setminus \{c_i\}$, and we have that $r_{j_i} = (v', v)$ and $r_{j_{i'}} = (v'', v)$ for some connections $\{(v', v), (v'', v)\} \subseteq E$ with $v' \neq v''$, $1 \leq j_i \leq k_i$, and $1 \leq j_{i'} \leq k_{i'}$, i.e.,

$$cn = |\{(\{c_i, c_{i'}\}, v) \mid c_i \in C, c_{i'} \in C \setminus \{c_i\}, 1 \leq j_i \leq k_i, 1 \leq j_{i'} \leq k_{i'}, \{r_{j_i}, r_{j_{i'}}\} \subseteq E, r_{j_i} = (v', v), r_{j_{i'}} = (v'', v), v' \neq v''\}|.$$

In words, the crossing number estimates how often the routes of distinct vehicles come together at a joint location (at different times). Although solutions are such that vehicles do not share any location at the same time, it is desirable to make routes as disjoint as possible to keep knock-on effects in case of a disruption short.

4. The *overlap number* on is the number of triples $(\{c_i, c_{i'}\}, r_{j_i}, r_{j_{i'}})$ such that $c_i \in C$, $c_{i'} \in C \setminus \{c_i\}$, and we have that $r_{j_i} = (v, v')$ and $r_{j_{i'}} = (v, v')$ (or $r_{j_{i'}} = (v', v)$) for some connection $(v, v') \in E$ (or connections $\{(v, v'), (v', v)\} \subseteq E$), $1 \leq j_i \leq k_i$, and $1 \leq j_{i'} \leq k_{i'}$, i.e.,

$$on = |\{(\{c_i, c_{i'}\}, r_{j_i}, r_{j_{i'}}) \mid c_i \in C, c_{i'} \in C \setminus \{c_i\}, 1 \leq j_i \leq k_i, 1 \leq j_{i'} \leq k_{i'}, \{r_{j_i}, r_{j_{i'}}\} \subseteq E, r_{j_i} = (v, v'), r_{j_{i'}} \in \{(v, v'), (v', v)\}\}|.$$

The motivation for aiming at a small overlap number is the same as with the crossing number, while connections taken by distinct vehicles are considered instead of crossing locations. Note that, for a bidirectional connection pair $\{(v, v'), (v', v)\} \subseteq E$, two vehicles c_i and $c_{i'}$ that move between v and v' in either direction can contribute one, two, or four triples $(\{c_i, c_{i'}\}, r_{j_i}, r_{j_{i'}})$, depending on whether neither, one, or both include (v, v') as well as (v', v) in their routes. Unlike that, only the triple $(\{c_i, c_{i'}\}, (v, v'), (v, v'))$ is obtained when $(v, v') \in E$ but $(v', v) \notin E$.

Fact	Meaning
node (v)	$v \in V$
halt ($v, d(v)$)	$v \in h(V)$ with halt duration $d(v)$
park ($v, d(v)$)	$v \in p(V)$ with park duration $d(v)$
stay ($v, d(v)$)	$v \in h(V) \cup p(V)$ with halt or park duration $d(v) > 1$
edge ($v, v', d(v, v')$)	$(v, v') \in E$ with move duration $d((v, v'))$
less (v', v'', v)	$\{(v', v), (v'', v)\} \subseteq E$ for lexicographically consecutive locations $v' < v''$
time (n)	$0 \leq n \leq \max\{d(t) \mid t \in T\}$
task (t)	$t \in T$
task ($t, d(t)$)	$t \in T$ with deadline $d(t)$
tasks (t, t')	$t \in T$ and $t' \in T \setminus \{t\}$
subtask ($t, s(i)$)	$t \in T$ with $1 \leq i \leq s(t)$
subtask ($t, s(i), v$)	$t \in T$, $1 \leq i \leq s(t)$, and $t[i] = v$ with $v \in h(V)$
vehicle (c)	$c \in C$
vehicle (c, v)	$c \in C$ with initial location $l(c) = v$

Table 1. Fact format for specifying automated guided vehicle routing scenarios in ASP

For each of the four objective criteria specified above, a smaller value is preferred to greater ones. The order of significance is chosen such that the completion of all transport tasks in as little time as possible has the highest priority, minimizing the overall utilization of vehicles comes second, keeping routes disjoint is third, and the avoidance of overlapping connections takes the lowest priority.

Example 3

The solution to the automated guided vehicle routing scenario from Example 1 indicated in Figure 2 and described further in Example 2 happens to be optimal relative to the applied objective criteria. Given that the sums 55 and 49 are obtained for the durations of elements of the route $\pi(c_1)$ or $\pi(c_2)$, respectively, the makespan ms matches the maximum 55, and the route length rl amounts to $55 + 49 = 104$. Regarding the crossing number cn , $\pi(c_1)$ and $\pi(c_2)$ visit three locations, 1, 4, and 7, via different connections, as witnessed by occurrences of (6,1) and (7,1), (3,4) and (7,4), as well as (1,7) and (4,7) in $\pi(c_1)$ or $\pi(c_2)$, respectively. Moreover, the overlap number $on = 14$ is obtained in view of the inclusion of the connections (1,2), (2,3), (3,4), (4,5), (5,6), and (6,1) in both $\pi(c_1)$ and $\pi(c_2)$ along with moves between the nodes 1 and 4 as well as 4 and 7 in either direction, each of the latter contributing four triples counted together by on . While the gap between the makespan $ms = 55$ and the deadline $d(t_1) = d(t_2) = 60$ for t_1 and t_2 may seem small, let us mention that 561 solutions are feasible and that only one of them is optimal. Such discrepancy clearly indicates that human engineers will hardly be able to perform an exhaustive optimization of routes in realistic scenarios of greater size only by hand. ■

3 Problem Encoding

Following the common modeling methodology of ASP, we represent (optimal) solutions to automated guided vehicle routing scenarios by facts specifying an instance along with a uniform problem encoding. To begin with, Table 1 surveys the format of facts describing the locations, connections, tasks, and vehicles belonging to an automated guided vehicle routing scenario. The respective fact representation of the scenario from Example 1 is given in Listing 1, using the shorthands ‘..’ and ‘;’ of *clingo* (Gebser et al.

Listing 1. Instance of automated guided vehicle routing with two tasks and two vehicles

```

node(v(1..7)).
halt(v(2),3). stay(v(2),3).
halt(v(4),3). stay(v(4),3).
halt(v(5),3). stay(v(5),3).
halt(v(6),3). stay(v(6),3).
park(v(7),2). stay(v(7),2).

edge(v(6;7),v(1),4). less(v(6),v(7),v(1)).
edge(v(1),v(2),4).
edge(v(2),v(3),4).
edge(v(3;7),v(4),4). less(v(3),v(7),v(4)).
edge(v(4),v(5),4).
edge(v(5),v(6),4).
edge(v(1;4),v(7),4). less(v(1),v(4),v(7)).

time(0..60).
task(t(1)). task(t(1),60). tasks(t(1),t(2)).
task(t(2)). task(t(2),60). tasks(t(2),t(1)).
subtask(t(1),s(1)). subtask(t(1),s(1),v(5)).
subtask(t(1),s(2)). subtask(t(1),s(2),v(4)).
subtask(t(1),s(3)). subtask(t(1),s(3),v(2)).
subtask(t(2),s(1)). subtask(t(2),s(1),v(6)).
subtask(t(2),s(2)). subtask(t(2),s(2),v(4)).
subtask(t(2),s(3)). subtask(t(2),s(3),v(2)).

vehicle(c(1)). vehicle(c(1),v(1)).
vehicle(c(2)). vehicle(c(2),v(2)).

```

2015) to abbreviate facts for a range or collection of arguments, respectively. Note that facts of the form $\text{stay}(v, d(v))$ combine halt and park nodes $v \in h(V) \cup p(V)$ whose associated duration $d(v)$ is greater than one, as such non-atomic durations are subject to dedicated conditions in our problem encoding below. For locations $v \in V$ that have several incoming connections, facts of the form $\text{less}(v', v'', v)$ provide lexicographically consecutive locations $v' < v''$ such that $\{(v', v), (v'', v)\} \subseteq E$, based on the standard term order of the ASP-Core-2 language (Calimeri et al. 2012). Our encoding below makes use of the lexicographical order of predecessor locations for a compact formulation of conditions to detect crossings. Moreover, facts $\text{time}(n)$ give time points of interest, ranging from zero to the latest deadline of any transport task, and $\text{tasks}(t, t')$ holds for distinct tasks $t \in T$ and $t' \in T \setminus \{t\}$, which have to be ordered by \prec when $\alpha(t) = \alpha(t')$. Facts of the remaining predicates list further properties of automated guided vehicle routing scenarios in a straightforward way, and their respective meanings are summarized in Table 1.

Our uniform problem encoding, given in Listings 2 and 3, can be understood as a merger of four logical parts, addressing task assignment and ordering, task completion, vehicle routing, as well as objective criteria (the latter part shown separately in Listing 3). Table 2 follows this logical structure in providing the meanings of predicates defined in each of the four parts, where a condition written in the right column applies if and only if a corresponding atom of the form given in the left column belongs to a stable model.

Atom	Meaning
assign (c, t)	$\alpha(t) = c$ for $t \in T$ and $c \in C$
share (t, t')	$\alpha(t) = \alpha(t')$ for $\{t, t'\} \subseteq T$ such that $t < t'$ lexicographically
order (t, t')	$t < t'$ for $t \in T$ and $t' \in T \setminus \{t\}$ such that $\alpha(t) = \alpha(t')$
check (t)	$t \in T$ is topological in $(T, \{(t', t'') \mid \text{order}(t', t'') \text{ holds}\})$
check (t, t')	$t \not< t'$ or check (t) holds for $t \in T$ and $t' \in T \setminus \{t\}$
pass ($c, t, s(i), v, d(v), n$)	$\alpha(t) = c$, $1 \leq i \leq s(t)$, $t[i] = v$, and $n \in u(c, v) \cup \{0 \mid l(c) = v\}$ for $t \in T$ and $c \in C$ with $\pi(c) = \langle r_1, \dots, r_k \rangle$ such that $m \leq \sum_{1 \leq j \leq m} d(r_j) \leq n < d(t)$, where $m = \max(\{g(t[i-1]) \mid 1 < i\} \cup \{g(t'[s(t')]) \mid t' < t\})$
done ($t, s(i), n$)	$t \in T$, $1 \leq i \leq s(t)$, and $1 \leq n \leq d(t)$ such that $g(t[i]) \leq$ $\sum_{1 \leq j \leq g(t[i])} d(r_j) \leq n$ for $\alpha(t) = c$ with $\pi(c) = \langle r_1, \dots, r_k \rangle$
wait (t, n)	$t' < t$ and $0 \leq n < \min\{d(t), d(t')\}$ for $t \in T$ and $t' \in T \setminus \{t\}$ such that $n < \max\{g(t'[s(t')]), \sum_{1 \leq j \leq g(t'[s(t')])} d(r_j)\}$, where $\alpha(t) = \alpha(t') = c$ with $\pi(c) = \langle r_1, \dots, r_k \rangle$
at (c, v, n)	$n \in u(c, v) \cup \{0 \mid l(c) = v\}$ for $c \in C$ and $v \in V$ with $n \leq \max\{d(t) \mid t \in T\}$
move (c, v, v', n)	$\pi(c) = \langle r_1, \dots, r_i, \dots, r_k \rangle$ for $c \in C$, $r_i \in E$, $r_i = (v, v')$, and $n = \sum_{1 \leq j < i} d(r_j)$ with $n + d(r_i) \leq \max\{d(t) \mid t \in T\}$
move (c, n)	move (c, v, v', n) holds for some connection $(v, v') \in E$
moving (c, v, v', n)	$\pi(c) = \langle r_1, \dots, r_i, \dots, r_k \rangle$ for $c \in C$, $r_i \in E$, $r_i = (v, v')$, and $\sum_{1 \leq j < i} d(r_j) < n \leq \sum_{1 \leq j \leq i} d(r_j) \leq \max\{d(t) \mid t \in T\}$
moving (v, v', n)	moving (c, v, v', n) holds for some vehicle $c \in C$
free ($c, d(v), n$)	$c \in C$, $v \in h(V) \cup p(V)$, and $0 \leq n \leq \max\{d(t) \mid t \in T\}$ with $d(v) > 1$ and $n' \in u(c, v) \cup \{0 \mid l(c) = v\}$ such that $(n' - 1) \notin u(c, v) \cup \{0 \mid l(c) = v\}$, $n = n' + i * d(v)$ for some $i \geq 0$, and $\{n'' \in u(c, e) \mid e \in E, n' < n'' < n,$ $n'' + d(e) - 1 \leq \max\{d(t) \mid t \in T\}\} = \emptyset$
free (c, n)	$c \in C$ and $0 \leq n \leq \max\{d(t) \mid t \in T\}$ such that $n \notin \bigcup_{v \in h(V) \cup p(V), d(v) > 1} u(c, v)$ or free (c, d, n) holds, where $d = d(v) > 1$ for some $v \in h(V) \cup p(V)$
used (c, n)	$c \in C$ and $0 \leq n \leq \max(\{0\} \cup \bigcup_{v \in V} u(c, v))$ with $n \leq \max\{d(t) \mid t \in T\}$
move (c, v, v')	move (c, v, v', n) holds for some $0 \leq n \leq \max\{d(t) \mid t \in T\}$
mark (c, v', v)	move (c, v'', v) holds for some connections $\{(v', v), (v'', v)\} \subseteq E$ such that $v' < v''$ lexicographically
same (c, c', v, v')	move (c, v, v') and move (c', v, v') hold for $\{c, c'\} \subseteq C$ such that $c < c'$ lexicographically

Table 2. Atoms characterizing solutions to automated guided vehicle routing scenarios

In more detail, the encoding part addressing task assignment and ordering is shown in Lines 3–12 of Listing 2. The choice rule in Line 3 makes sure that exactly one atom of the form `assign (c, t)` holds per task $t \in T$, providing the vehicle $c \in C$ such that $\alpha(t) = c$. Further rules deal with the order $<$ among tasks assigned to a common vehicle, where `share (t, t')` is derived by the rule in Line 5 if $\alpha(t) = \alpha(t')$ for tasks $t < t'$. The choice rule in Line 6 then either picks `order (t, t')`, expressing that $t < t'$, or the rule in Line 7 yields `order (t', t)`, standing for $t' < t$, otherwise. The remaining rules in Lines 9–12 check that the order relation $<$ is acyclic, utilizing modeling methods detailed in (Gebser et al.).

The second encoding part in Lines 16–27 deals with the completion of transport tasks. To this end, the two rules in Lines 16–19 indicate time points n such that the vehicle $c \in C$

Listing 2. Encoding of task assignment, completion, and automated guided vehicle routing

```

1  % task assignment
3  {assign(C,T) : vehicle(C)} = 1 :- task(T).
5  share(T1,T2) :- assign(C,T1), assign(C,T2), T1 < T2.
6  {order(T1,T2)} :- share(T1,T2).
7  order(T2,T1) :- share(T1,T2), not order(T1,T2).
9  check(T1,T2) :- tasks(T1,T2), not order(T1,T2).
10 check(T1,T2) :- tasks(T1,T2), check(T1).
11 check(T2) :- task(T2), check(T1,T2) : tasks(T1,T2).
12 :- task(T2), not check(T2).
14 % task completion
16 pass(C,T,s(1),V,D,N) :- assign(C,T), at(C,V,N), halt(V,D), task(T,M),
17                          subtask(T,s(1),V), not wait(T,N), N < M.
18 pass(C,T,s(I),V,D,N) :- assign(C,T), at(C,V,N), halt(V,D), task(T,M),
19                          subtask(T,s(I),V), done(T,s(I-1),N), N < M.
21 done(T,S,N+D) :- pass(C,T,S,V,D,N), task(T,M), not move(C,N), N+D <= M.
22 done(T,S,N+1) :- done(T,S,N), not task(T,N).
23 :- task(T,M), subtask(T,S), not done(T,S,M).
25 wait(T2,N) :- order(T1,T2), task(T1,M1), task(T2,M2), time(N),
26               subtask(T1,s(I)), not subtask(T1,s(I+1)),
27               not done(T1,s(I),N), N < M1, N < M2.
29 % vehicle routing
31 at(C,V,0) :- vehicle(C,V), time(0).
32 at(C,V2,N+D) :- move(C,V1,V2,N), edge(V1,V2,D).
33 at(C,V,N+1) :- at(C,V,N), park(V,D), time(N+1), not move(C,N).
34 at(C,V,N+1) :- pass(C,T,S,V,D,N), not done(T,S,N), not move(C,N).
35 :- node(V), time(N), #count{C : at(C,V,N)} > 1.
36 :- vehicle(C), time(N), #count{V : at(C,V,N)} > 1.
38 {move(C,V1,V2,N) : edge(V1,V2,D), time(N+D)} < 2 :- vehicle(C), time(N).
39 move(C,N) :- move(C,V1,V2,N).
40 :- move(C,V1,V2,N), not at(C,V1,N).
42 moving(C,V1,V2,N+1) :- move(C,V1,V2,N).
43 moving(C,V1,V2,N+1) :- moving(C,V1,V2,N), time(N+1), not at(C,V2,N).
44 moving(V1,V2,N) :- moving(C,V1,V2,N).
45 :- moving(V1,V2,N), moving(V2,V1,N).
47 free(C,D,N) :- at(C,V,N), stay(V,D), not at(C,V,N-1).
48 free(C,D,N+D) :- free(C,D,N), time(N+D), not move(C,N).
49 free(C,N) :- free(C,D,N).
50 free(C,N) :- vehicle(C), time(N), not at(C,V,N) : stay(V,D).
51 :- move(C,N), not free(C,N).

```

with $\alpha(t) = c$ visits the halt node given by a subtask $t[i]$, while any preceding subtasks according to \prec as well as the sequence of subtasks of t are already completed at time n . In case $i = 1$, the latter condition is in Line 17 checked by the absence of yet incomplete tasks t' such that $t' \prec t$, and otherwise the completion of $t[i - 1]$ up to time n is required by preconditions in Line 19. The rule in Line 21 then derives $\text{done}(t, s(i), n + d(t[i]))$, signaling the completion of $t[i]$, provided that $n + d(t[i])$ does not exceed the deadline $d(t)$ for t and the vehicle c to which t is assigned halts at time n . Note that the occurrence of a halt in the route of c is determined by the absence of $\text{move}(c, n)$ from a stable model, an atom that would otherwise indicate that c takes some connection at time n . The condition that all transport tasks have to be completed within their respective deadlines is further taken care of by the rule in Line 22, which successively propagates the completion of a subtask $t[i]$ on to the deadline $d(t)$, along with requiring the completion of any subtask at $d(t)$ by means of the integrity constraint in Line 23. Finally, the rule in Lines 25–27 derives $\text{wait}(t, n)$, an atom inspected in Line 17, in view of a yet incomplete subtask $t'[s(t')]$ of some task t' such that $t' \prec t$ at time $n < d(t)$, where considering the last subtask $t'[s(t')]$ of t' only is sufficient because the subtasks of t' must be completed in order.

The actual routes of vehicles are tracked by the third encoding part in Lines 31–51. To begin with, the rule in Line 31 derives $\text{at}(c, l(c), 0)$, expressing that any vehicle $c \in C$ starts from its initial location $l(c)$ at time 0. The new location $v' \in V$ resulting from a move $(v, v') \in E$ at time n is reflected by the rule in Line 32 yielding $\text{at}(c, v', n + d((v, v')))$. A stop at some park node $v \in p(V)$ is addressed by the rule in Line 33, deriving $\text{at}(c, v, n + 1)$ from $\text{at}(c, v, n)$ in the absence of any move made by c at time n . Moreover, the rule in Line 34 takes care of a halt to complete some subtask $t[i]$ by deriving $\text{at}(c, t[i], n + 1)$ in case no subtask preceding $t[i]$ is yet incomplete and $t[i]$ itself is not already completed at time n , provided that the vehicle c with $\alpha(t) = c$ does not make any move at n . As a consequence, atoms of the form $\text{at}(c, v, n)$ in a stable model match times $n \in u(c, v) \cup \{0 \mid l(c) = v\}$ for vehicles $c \in C$ and locations $v \in V$, where halts can only be made for completing assigned subtasks. Given this, the integrity constraint in Line 35 makes sure that $u(c, v) \cap u(c', v) = \emptyset$ for any $c' \in C \setminus \{c\}$, and the one in Line 36 asserts that $u(c, v) \cap u(c, v') = \emptyset$ for any $v' \in V \setminus \{v\}$, thus constituting a redundant/entailed state constraint that may still be helpful regarding solving performance (Gebser et al. 2012).

Unlike stops at halt or park nodes, a move $(v, v') \in E$ by some vehicle $c \in C$ at time n such that $n + d((v, v'))$ is still of interest is signaled by an atom $\text{move}(c, v, v', n)$, picked by means of the choice rule in Line 38. The rule in Line 39 then provides the projection of such an atom to $\text{move}(c, n)$, and the integrity constraint in Line 40 requires c to be at the location v at time n . Moreover, atoms $\text{moving}(c, v, v', n')$, derived by the rules in Lines 42–43 for $n < n' \leq n + d((v, v'))$, yield time points $n' \in u(c, (v, v'))$. Their projection to $\text{moving}(v, v', n')$ in Line 44 is inspected by the integrity constraint in Line 45 to, in case $(v', v) \in E$, make sure that $u(c, (v, v')) \cap u(c', (v', v)) = \emptyset$ holds for any $c' \in C \setminus \{c\}$. The remaining rules in Lines 47–51 check that the moves of a vehicle c respect non-atomic durations $d(v) > 1$ of halt and park nodes $v \in h(V) \cup p(V)$. To this end, an atom $\text{free}(c, d(v), n)$ is derived by the rule in Line 47 when c is not already at v at time $n - 1$, and then propagated on in steps of the halt or park duration $d(v)$ by the rule in Line 48 as long as c does not make any move. The projection to $\text{free}(c, n)$ in Line 49 along with the rule in Line 50, also deriving $\text{free}(c, n)$ in case c is not at any halt or

Listing 3. Encoding part for makespan, route length, crossing, and overlap minimization

```

53 % objective criteria
55 used(C,N) :- at(C,V,N).
56 used(C,N-1) :- used(C,N), 0 < N.
57 :~ used(C,N). [1@4,N]
58 :~ used(C,N). [1@3,C,N]

60 move(C,V1,V2) :- move(C,V1,V2,N).

62 mark(C,V1,V) :- less(V1,V2,V), move(C,V2,V).
63 mark(C,V1,V) :- less(V1,V2,V), mark(C,V2,V).
64 :~ mark(C1,V1,V), move(C2,V1,V), C1 < C2. [1@2,C1,C2,V]
65 :~ move(C1,V1,V), mark(C2,V1,V), C1 < C2. [1@2,C1,C2,V]

67 same(C1,C2,V1,V2) :- move(C1,V1,V2), move(C2,V1,V2), C1 < C2.
68 :~ move(C1,V1,V2), move(C2,V2,V1), C1 < C2, V1 < V2. [1@1,C1,C2,V1,V2]
69 :~ move(C1,V2,V1), move(C2,V1,V2), C1 < C2, V1 < V2. [1@1,C1,C2,V1,V2]
70 :~ same(C1,C2,V1,V2), V1 < V2. [1@1,C1,C2,V1,V2]
71 :~ same(C1,C2,V2,V1), V1 < V2. [1@1,C1,C2,V1,V2]
72 :~ same(C1,C2,V1,V2), move(C1,V2,V1), V1 < V2. [1@1,C1,C2,V2,V1]
73 :~ same(C1,C2,V1,V2), move(C2,V2,V1), V1 < V2. [1@1,C1,C2,V2,V1]
74 :~ same(C1,C2,V2,V1), move(C1,V1,V2), V1 < V2. [1@1,C1,C2,V2,V1]
75 :~ same(C1,C2,V2,V1), move(C2,V1,V2), V1 < V2. [1@1,C1,C2,V2,V1]
76 :~ same(C1,C2,V1,V2), same(C1,C2,V2,V1), V1 < V2. [2@1,C1,C2,V1,V2]

```

park node $v \in h(V) \cup p(V)$ with $d(v) > 1$ at time n , indicate all times n at which c may make a move, and the integrity constraint in Line 51 restricts moves to such time points.

While the encoding parts described so far make sure that stable models match solutions to automated guided vehicle routing scenarios, the fourth part in Listing 3 addresses the objective criteria to distinguish optimal routes. To begin with, the rules in Lines 55–56 derive atoms $\text{used}(c, n)$ to indicate that the route of a vehicle $c \in C$ is not finished before time n . The makespan ms is then minimized by means of the weak constraint with the highest priority 4 in Line 57, associating a cost of 1 with each time n . To then minimize the route length rl with priority 3, the weak constraint in Line 58 likewise penalizes $\text{used}(c, n)$, where a cost of 1 for each time n is charged per vehicle c . The rule in Line 60 provides the projection of moves to $\text{move}(c, v, v')$ in order to formulate conditions regarding crossings and overlaps of routes without referring to particular times of moves. In fact, the rules in Lines 62–63 investigate and propagate $\text{move}(c, v'', v)$ on to locations v' such that $(v', v) \in E$ and $v' < v''$ lexicographically, thus signaling the inclusion of some other predecessor location of v in the route of c in terms of $\text{mark}(c, v', v)$. The weak constraints in Lines 64–65 make use of this to detect a crossing at v by the move (v', v) of another vehicle $c' \in C \setminus \{c\}$, while the lexicographical order of distinct predecessor locations of v in the routes of c and c' may likewise be switched. In either case, a crossing at v accounts for a cost of 1 with priority 2, where the set $\{c, c'\}$ in pairs $(\{c, c'\}, v)$ contributing to cn is reflected by requiring that $c < c'$ lexicographically. The remaining rules and weak constraints take care of overlapping connections, where the rule in Line 67

yields same (c, c', v, v') when distinct vehicles $c < c'$ both include (v, v') in their routes. This case as well as taking (v, v') and (v', v) in opposite directions is sanctioned by a cost of 1 with priority 1 in view of the weak constraints in Lines 68–71. If c or c' includes both directions, (v, v') and (v', v) , in its route, an additional cost of 1 is charged by some of the weak constraints in Lines 72–75, by means of using a different term order regarding v and v' than before. Finally, the weak constraint in Line 76 accounts for a cost of 2 if c and c' both take (v, v') as well as (v', v) , in which case the resulting sum of four matches the number of triples $(\{c, c'\}, e, e')$ such that $\{e, e'\} = \{(v, v'), (v', v)\}$ contributing to on . As a consequence, we have that the sums of costs with particular priority coincide with the objectives ms , rl , cn , and on , where priorities follow the criteria’s order of significance.

Example 4

The optimal solution to the automated guided vehicle routing scenario from Example 1 indicated in Figure 2 and described further in Example 2 is such that the vehicles c_1 and c_2 finish their last halts at $t_1[3] = t_2[3] = 2$ at the times 55 and 49. Hence, a stable model representing this solution, obtained for the facts in Listing 1 along with the encoding in Listings 2 and 3, contains the atoms $at(c(1), v(2), 55)$ and $at(c(2), v(2), 49)$. These in turn imply atoms $used(c(i), n_i)$ for $i \in \{1, 2\}$, $0 \leq n_1 \leq 55$, and $0 \leq n_2 \leq 49$. The weak constraints with priorities 4 and 3 thus yield 56 or 106 distinct terms, respectively, of the form $[1@4, n_i]$ or $[1@3, c(i), n_i]$, where the numbers of terms match $ms + 1 = 55 + 1 = 56$ and $rl + i = 104 + 2 = 106$. Note that the addends 1 or i to ms and rl are due to atoms of the form $used(c(i), 0)$, which can be deterministically derived in view of the initial locations of vehicles and do not affect the preference order of solutions. Regarding crossings, as detailed in Example 3, the connections taken by c_1 and c_2 trigger weak constraints associated with the terms $[1@2, c(1), c(2), v(1)]$, $[1@2, c(1), c(2), v(4)]$, and $[1@2, c(1), c(2), v(7)]$, leading to the sum $cn = 3$ of costs. Moreover, six unidirectional overlaps of the routes of c_1 and c_2 are indicated by terms of the form $[1@1, c(1), c(2), v(v), v(v')]$ for $(v, v') \in \{(1, 2), (2, 3), (3, 4), (4, 5), (5, 6), (1, 6)\}$, among which the lexicographically ordered pair $(1, 6)$ stands for the connection $(6, 1)$ taken in the opposite direction. However, such term ordering matters for bidirectional connections only, where $(v, v') \in \{(1, 4), (4, 7)\}$ yield three distinct terms each: $[1@1, c(1), c(2), v(v), v(v')]$, $[1@1, c(1), c(2), v(v'), v(v)]$, and $[2@1, c(1), c(2), v(v), v(v')]$. That is, the weak constraints with priority 1 lead to twelve terms in total, ten of them accounting for a cost of 1 and two having a cost of 2 (written before ‘@’), summing up to $on = 14$. ■

4 Empirical Results

A preliminary field study by production engineers at Mercedes-Benz Ludwigsfelde GmbH investigated viable options for implementing a new control system for automated guided vehicles used in car assembly, where one goal consists of setting the routes for such vehicles up automatically rather than by hand. This study raised interest in the open-source control system software *openTCS* (see <http://www.opentcs.org>), developed by Fraunhofer IML. While *openTCS* offers a generic platform for automated guided vehicle control, it is not geared to ship a solver for hard combinatorial tasks such as the multi-objective optimization of routes off the shelf. Instead, the system is extensible and allows for integrating customized components, which is where our ASP approach to vehicle

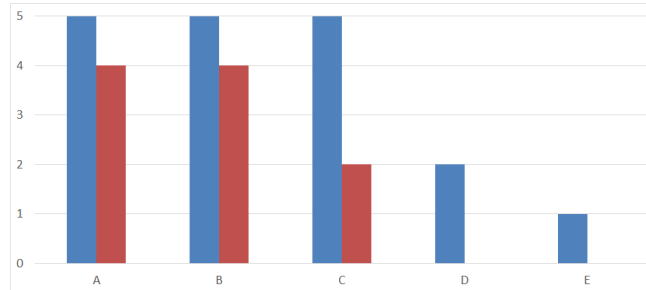


Fig. 3. Use cases solved by means of ASP (in blue) and the default scheduler of *openTCS* (in red)

routing is envisaged to come into play in the future. For reference, however, we contrast our results to those obtained with the default scheduler of *openTCS*, which implements a simple round-robin procedure to assign tasks to and pick shortest routes for vehicles.

In order to evaluate the practical applicability of control systems, the engineers at Mercedes-Benz Ludwigsfelde GmbH defined 18 use cases based on a factory layout with 25 locations and 35 connections, resembling the storage areas, assembly lines, and transport routes at the physical car factory. Such use cases are run in simulation to test the entire functionality of a control system, where task assignment and vehicle routing are incorporated at the high level. According to the main test targets, the use cases (available at <http://www.cs.uni-potsdam.de/wv/projects/daimler/resources-iclp18.tar.xz>) are grouped into five categories: A) communication and feedback, B) task assignment and execution, C) routing and traffic management, D) special conditions and breakdowns, and E) full production cycle. While test cases in the first four groups focus on particular scenarios of small size, i.e., up to three tasks and vehicles, the use case in category E emulates a full production cycle of roughly 20 minutes in real time, involving ten transport tasks with 39 subtasks in total to be accomplished by four automated guided vehicles.

For computing optimal routes with *clingo* (version 5.2.2), we represented the scenarios of the use cases in terms of facts as described in the previous section, thus focusing on task assignment and vehicle routing rather than system control. As indicated by the blue bars in Figure 3, we succeeded in obtaining provably optimal routes in all of the 18 use cases, where the runtimes of *clingo* on a desktop machine equipped with Intel i7-6700 3.40GHz CPU ranged from split seconds to ten seconds at most for each of the 17 small scenarios in the first four groups. While these small instances stem from use cases primarily created to test the reaction of a control system to some isolated situation, the scenario in E aims at managing all of the transport tasks recurring within each full production cycle, a complex operation that has so far been carried out by human engineers before a production process starts or is resumed with an updated configuration, respectively. In fact, optimizing the routes for automated guided vehicles over the full production cycle is computationally challenging, even for a system like *clingo* that is geared to hard combinatorial problem solving, and it took about 5 hours to compute a provably optimal solution. Its makespan of 225 clock cycles amounts to five times as many seconds in real time, roughly 20 minutes, the route length 891 corresponds to about 75 minutes of operation, shared between four vehicles whose computed routes involve 39 crossings and 141 overlaps in total. While 5 hours of computation time can certainly not be afforded by a control system that has to

react to incidents in real time, the dimension of what has to be anticipated in comparable scenarios is nevertheless encouraging, as it tells us that routes for accomplishing recurrent transport tasks in realistic production processes can be programmed in an automated fashion, going along with optimality that cannot be established by human engineers alone.

The red bars shown in Figure 3 exhibit that the default scheduler shipped with *openTCS* fails to come up with a feasible solution in 8 of the 18 use cases, which is due to its greedy approach to assign a task and lock locations along the shortest route of a vehicle one by one. E.g., such an approach is bound to fail on the example scenario in Figure 2, where each of the vehicles c_1 and c_2 must necessarily visit node 4 in order to complete any subtask, which locks the respective other vehicle out from visiting node 4, while a single vehicle cannot alone complete both transport tasks, t_1 and t_2 , within their deadlines. This observation along with the fact that *clingo* can timely handle the small scenarios of use cases in the first four groups clearly motivate its integration into *openTCS* as a component in charge of task assignment and vehicle routing, which we are working on.

5 Discussion

The problem of assigning transport tasks to and routing automated guided vehicles in car assembly reveals parallels to Generalized Target Assignment and Path Finding (GTAPF; Nguyen et al. 2017) as well as Temporal Planning (cf. Fisher 2008). Similar to automated guided vehicle routing scenarios, GTAPF allows for more tasks than there are agents, sequences of subtasks, and deadlines for tasks, while its framework assumes that tasks are completed according to some total order and it does not feature (non-atomic) durations. Unlike that, durations play a fundamental role in Temporal Planning, but standard representations such as the Planning Domain Definition Language (PDDL; Fox and Long 2003) lack constructs for conveniently modeling task assignment and ordering. We thus stick to a native ASP encoding, and a corresponding approach likewise turned as advantageous for planning by means of tabled logic programming (Zhou et al. 2015).

Early work on using ASP for solving basic multi-agent path finding problems was done by Erdem et al. (2013); also, Nguyen et al. (2017) used ASP for solving GTAPF problems. Action durations and intervals were previously considered in ASP by Son et al. (2004). Notably, we checked that *clingo* requires less than 1 GB of RAM to represent durations and makespan of a full production cycle at the car factory of Mercedes-Benz Ludwigsfelde GmbH, while extensions by difference logic (Neubauer et al. 2017) or integer variables (Banbara et al. 2017) may possibly handle even greater time intervals in a compact way.

The contributions of our work include a transparent and elaboration tolerant formalization of transport tasks evolving in realistic production processes by specifying them in ASP, thus also making off-the-shelf solving systems accessible for computing optimal vehicle routes. As it turns out, our declarative approach allows for handling a scenario covering the full production cycle at the car factory of Mercedes-Benz Ludwigsfelde GmbH, so that it can assist human engineers in setting up routes to be periodically taken by automated guided vehicles. Moreover, scenarios of small size, resembling task reassignment or vehicle rerouting in case of unforeseen circumstances, can virtually be handled in real time, which also makes the future integration of *clingo* as a component of the control system software *openTCS* for task assignment and vehicle routing attractive.

Acknowledgments This work was partially funded by DFG grant SCHA 550/9. We are grateful to the anonymous reviewers for their helpful comments.

References

- BANBARA, M., KAUFMANN, B., OSTROWSKI, M., AND SCHAUB, T. 2017. Clingcon: The next generation. *Theory and Practice of Logic Programming* 17, 4, 408–461.
- CALIMERI, F., FABER, W., GEBSER, M., IANNI, G., KAMINSKI, R., KRENNWALLNER, T., LEONE, N., RICCA, F., AND SCHAUB, T. 2012. ASP-Core-2: Input language format.
- ERDEM, E., KISA, D., ÖZTOK, U., AND SCHÜLLER, P. 2013. A general formal framework for pathfinding problems with multiple agents. In *Proceedings of AAAI’13*. AAAI Press, 290–296.
- FISHER, M. 2008. Temporal representation and reasoning. In *Handbook of Knowledge Representation*. Elsevier Science, 513–550.
- FOX, M. AND LONG, D. 2003. PDDL2.1: An extension to PDDL for expressing temporal planning domains. *Journal of Artificial Intelligence Research* 20, 61–124.
- GEBSER, M., JANHUNEN, T., AND RINTANEN, J. Declarative encodings of acyclicity properties. *Journal of Logic and Computation*, in press.
- GEBSER, M., KAMINSKI, R., KAUFMANN, B., LINDAUER, M., OSTROWSKI, M., ROMERO, J., SCHAUB, T., AND THIELE, S. 2015. *Potassco User Guide*. University of Potsdam.
- GEBSER, M., KAMINSKI, R., KAUFMANN, B., AND SCHAUB, T. 2012. *Answer Set Solving in Practice*. Morgan and Claypool Publishers.
- LIFSCHITZ, V. 1999. Answer set planning. In *Proceedings of ICLP’99*. MIT Press, 23–37.
- NEUBAUER, K., WANKO, P., SCHAUB, T., AND HAUBELT, C. 2017. Enhancing symbolic system synthesis through ASPmT with partial assignment evaluation. In *Proceedings of DATE’17*. IEEE Press, 306–309.
- NGUYEN, V., OBERMEIER, P., SON, T., SCHAUB, T., AND YEOH, W. 2017. Generalized target assignment and path finding using answer set programming. In *Proceedings of IJCAI’17*. IJCAI/AAAI Press, 1216–1223.
- SON, T., BARAL, C., AND TUAN, L. 2004. Adding time and intervals to procedural and hierarchical control specifications. In *Proceedings of AAAI’04*. AAAI Press, 92–97.
- ZHOU, N., BARTÁK, R., AND DOVIER, A. 2015. Planning as tabled logic programming. *Theory and Practice of Logic Programming* 15, 4-5, 543–558.