Various Approaches to the Application of Answer Set Programming in Order-picking Systems with Intelligent Vehicles

Steffen Schieweck[1,2], Gabriele Kern-Isberner[1] and Michael ten Hompel[2]
[1]*Chair 1 Computer Science, TU Dortmund, Otto-Hahn-Str. 12, Dortmund, Germany*
[2]*Chair of Materials Handling and Warehousing, TU Dortmund, Joseph-von-Fraunhofer-Str. 2-4, Dortmund, Germany*

Keywords: Answer Set Programming, Multi-agent Systems, Application, Hybrid Systems.

Abstract: Intelligent, self-driving vehicles on public roads are widely noted in the media nowadays. In warehouse and production systems, such vehicles have been common for a number of years, even though their intelligence has only come to awareness in the recent years. Those systems are especially tailored to work in volatile environments which change (to some degree) every few weeks or months. Thus, programming is required to be as flexible as possible while still providing high efficiency. Answer set programming is a well known paradigm which has received remarkable attention in the recent years. In this paper several approaches are presented and evaluated to apply answer set programming to an order-picking system with intelligent vehicles. The interconnected planning tasks are the dispatching of vehicles to driving jobs and the assignment of customer orders to picking stations.

1 INTRODUCTION

The currently established continuous conveying systems are tailored to work in environments where steady and well-predictable demand occurs. Intelligent, self-driving vehicles are the answer to a global market which has become highly unpredictable and fast-moving. However, the intelligence of those vehicles is restricted by the tasks and solutions the designer and programmer initially envisaged. As a result, more costly programming has to be conducted whenever a new task or solution is introduced. Answer set programming (ASP) is well known for its simplicity and efficiency for programming and finding optimal solutions and thus serves an ideal tool for the systems mentioned above. In this paper we discuss several approaches to the fusion of intelligent vehicles for order-picking with ASP and approach the question if this might be beneficial.

The planning task which is implemented with ASP has the restriction of not requiring any physical adaption of the system. Still, we aim for a significant increase of performance. A combinatorial problem for such order-picking systems is vehicle dispatching, meaning the assignment of driving jobs to vehicles. In the specific context of order-picking systems, the question arises to which picking station a customer order is assigned. All of the items of a customer order must be transported to the same picking station which have limited capacity. Thus, due to limited capacity a driving job may not be able to select (see section 3) which makes the decisions interconnected. Also, they must be completed at the same time and raise potential for a holistic optimization of the system, e.g. concerning the utilization of the picking stations.

2 FUNDAMENTALS

In the following we will provide some fundamentals of the presented work. The concept of answer set programming will be introduced briefly (further details may be found in e.g. (Dovier et al., 2009) and (Gebser, 2013)). After that, cellular transport systems are described. A (more) formal definition of the planning task is presented in chapter 3.

2.1 Answer Set Programming

An answer set program $\mathcal{P}$ consists of a number of rules of the form

$$r: \ H \leftarrow A_1, ..., A_n, \ not\, B_1, ..., not\, B_m. \qquad (1)$$

where $H, A_1, ..., A_n, B_1, ..., B_m$ are literals and "*not*" is a so-called *default negation* operator. H is $head(r)$

and $A_1,...,A_n, not\, B_1,...,not\, B_m$ is $body(r)$, respectively. H holds if $pos(r) = A_1,...,A_n$ is true and $neg(r) = B_1,...,B_m$ are false or not known. A rule without body is called a fact and holds without any precondition. A rule without head is a constraint and excludes the set defined in $body(r)$ from the *answer set*.

An encoding of an answer set program contains a number of such rules which may (and will) interconnect. A valid answer set satisfies all of the given rules. While the most basic structure of a rule is as described, rules may look differently to achieve specific grounding and solving behaviors (Gebser, 2013). With todays ASP-grounders and -solvers, one may specify objective functions to select an optimal answer set from the set of valid ones.

A set $\mathcal{S}$ of literals is a model (an answer set) of $\mathcal{P}$, if $H \in \mathcal{S}$ whenever $pos(r) \subseteq \mathcal{S}$ and $neg(r) \cap \mathcal{S} = \emptyset$ for every $r \in \mathcal{P}$. $\mathcal{S}$ is a stable model of $\mathcal{P}$, if $\mathcal{S}$ is the $\subseteq$-minimal model of $\mathcal{P}^{\mathcal{S}}$ where $\mathcal{P}^{\mathcal{S}}$ is the reduct of $\mathcal{P}$ relative to the set $\mathcal{S}$ as defined by (Gelfond and Lifschitz, 1988; Gelfond and Lifschitz, 1991)

$$\begin{aligned} \mathcal{P}^{\mathcal{S}} := \{H \leftarrow A_1,...,A_n \mid \\ H \leftarrow A_1,...,A_n, not\ B_1,...,not\ B_m \in \mathcal{P}, \\ \{B_1,...,B_m\} \cap \mathcal{S} = \emptyset\} \end{aligned} \quad (2)$$

A rational agent can gain knowledge from $\mathcal{P}$. He considers any literals $\mathcal{P}^{\mathcal{S}}$ *true* and the remaining literals *false*.

2.2 Cellular Transport System

In the domain of facility logistics, cellular transport systems received considerable attention in the recent years. They are the embodiment of multi-agent systems for the purpose of in-house transport of goods and handling units such as pallets, bins or cartons. Cellular Transport Systems consist of structural and functional elements like racks, picking stations and lifts and a number of conveying units. The transport units are intelligent and self-organizing which enables them to decide and act autonomously. They may be manifested by single continuous conveying units which are able to work upon "plug-and-play" by detecting the meta-structure of the conveying system and collaborate to achieve conveying objectives (Mayer, 2009). In this paper, a manifestation in form of a fleet of intelligent vehicles is considered which operate in an order-picking system.

The Cellular Conveyor System has been developed jointly by *Fraunhofer-Institute for Material Flow and Logistics IML* and *Dematic GmbH*. Its unique feature is the vehicles capability of moving on the various levels of rack as well as on the ground

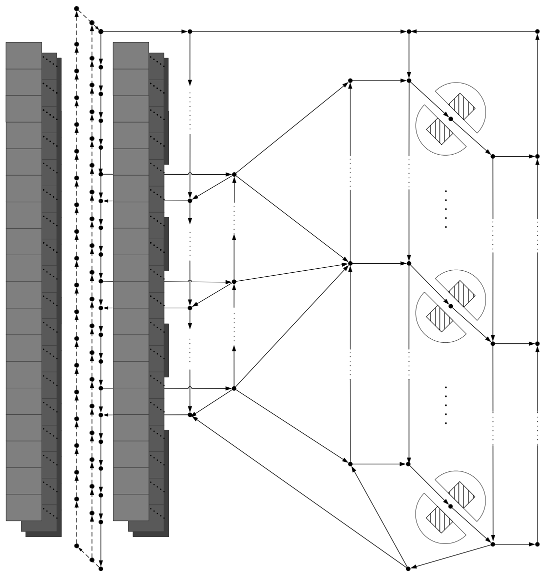
Figure 1: Example of routing graph.

floor. The system consists of one (or multiple) racks, picking stations, article bins, order bins, vehicles and lifts which transport the vehicles vertically. The article bins are stored in the rack(s). If a specific stock keeping unit (*sku*) is ordered, the related article bin is retrieved by a vehicle and transported to one of the picking stations. The system is automated by a large degree. Nonetheless, as in most order-picking systems, the isolated pick-and-put procedure of skus from the article bins to the order bins is conducted by workers (order pickers) who are located at the picking stations. After picking has been completed the vehicle transports the bin back into the rack and stores it. The lifts are required to reach the various levels of the rack. They are located at the beginning and the end of an aisle. The vehicles move on dedicated virtual paths which compose to a graph (see Figure 1). The edges of the graph are unidirectional. Thus, no reservation procedure for edges is necessary. As an additional consequence, one of the lifts is dedicated for the upbound transport of the vehicles and the other lift for the downbound transport.

The control architecture of the vehicles consists of three layers which are connected by UDP-sockets. The sensor and actuator layer is at the lowest level and responsible for tasks which are strongly coupled to the hardware and time-critical (e.g. position control, safety, sensor data acquisition). The operational layer is on the mid-level and responsible for tasks like localization, path planning and collision avoidance. The autonomous behavior is located at the strategical level and implemented via software agents. Those agents are capable of communicating with other agents in the system (Kamagaew et al., 2011).

Currently, such an external system is responsible for job management (aka dispatching) (Kamagaew et al., 2011). Dispatching is conducted using the FIPA protocol (Foundation for Intelligent Physical Agents FIPA, 2002). The external system releases a broadcast to all vehicles once a new order-line arrive. The available (idle) vehicles bid for the order-line and the highest bidder is assigned the offered driving job. Hence, if the system runs under full utilization (which is desirable for economical reasons), the operation results in a simple first-come-first-served (*FIFO*) procedure.

2.3 Related Work

The assignment of driving tasks to vehicles has been studied intensely in the domain of operations research. The *vehicle scheduling problem* decides when, where and how any vehicle in a system shall act. Routing is included in this problem. The problem may be tackled online or offline. The offline problem can theoretically be solved to optimality using a multiple traveling salesman model which has NP-complexity. Real-life scenarios require online solving due to uncertainty of the environment states. This may be realized with a rolling horizon, in which the planning is conducted for a limited future timespan (Le-Anh and de Koster, 2006).

Vehicle dispatching strategies only assign driving jobs to vehicles. Often, the planning horizon has a timespan of zero. This results in simple rules from which the assignment is made. Especially in multi-agent systems, the choice weather the decision is made central or distributed is of interest (Vis, 2006).

Existing research dealing with the theoretical foundations is diverse. Among them are optimization models (Yang et al., 2004), heuristics (Bartholdi and Platzman, 1989), disposition strategies (de Koster et al., 2004) but also more complex methods such as fuzzy logic (Benincasa et al., 2003) and ant colony algorithms (Saidi-Mehrabad et al., 2015) are used.

Cellular transport systems are a rather new field of research. Therefore, publications are sparse and mainly engage with the physical design of the systems (Guizzo, 2008; Kamagaew et al., 2011). To our knowledge, no publications from other authors exist which deal with the combination of ASP and cellular transport systems.

3 TASK DEFINITION

An order-picking system has the purpose of the assembly of customer orders. In a warehouse, a set M of skus is held in inventory. A customer order $O \subset M$ consists of a number $n_{P,O}$ of skus which are condensed to $n_O \leq n_{P,O}$ order-lines l in which skus with the same identity compose to one order-line[1]. The system has a number of picking stations S, storage positions R and vehicles F. The picking stations have equal capacity c_S. The incoming customer orders are stored in a list of orders L. Given the time t_O of an order O the orders are sorted such that $t_1 \leq t_2 \leq ... \leq t_{q-1} \leq t_q$.

The vehicles move on a graph $G = (V,E)$ with a set of nodes V and a set of edges E. $V = \{V_R; V_S; V_W\}$ where V_R are nodes which identify storage locations in the rack, V_S are nodes which identify picking stations where bins need to be delivered to and V_W are waypoints without further functionality. The edges are unidirectional. On G, the vehicles need to travel a distance d to fulfill the systems purpose.

The planning task is the assignment $l_v = (v,l)$ of a vehicle v to an order-line and the related assignment $o_s = (O,S)$ of an order to a picking station. We aim to maximize the number of satisfied order-lines per time N_l:

$$z_1 : \max N_l \tag{3}$$

Also, a balanced utilization u_s between the picking stations is favored:

$$z_2 : \min \sum_S |u_{avg} - u_s| \tag{4}$$

with

$$u_{avg} = \frac{1}{S} \sum_{i=1}^{S} u_i \tag{5}$$

One vehicle can only transport one bin at a time. The capacity c_S of a picking station may not be exceeded by the number of orders O assigned to the same picking station at a time.

For the planning task, the agents are given a horizon H with size n. Informally speaking, they are given the opportunity to select an order-line from a *pool* which consists of the next n unfulfilled order-lines of L. Note that, if the same article is part of multiple orders in H, the vehicle may approach multiple picking stations between retrieval and storing to satisfy multiple order-lines with one cycle. Thus, we define a driving job j which is selected from H. Every unique storage position is one $Pos \in H$. Every job j has one pickup node $V_p \in V_R$, one or multiple delivery nodes $V_d \subseteq V_S$ and one storing node $V_r = V_p$. If

[1]E.g. if a customer orders 500 business cards, $n_O = 1$.

j has multiple delivery nodes it will be referred to as an eos-job (*economy of scale*). The latter definition of the storing node implies that bins are transported back to the same location from which they were extracted. Practically speaking, we assume a fixed storage policy. Finally, we define that all of the order-lines l of a customer-order O must be transported to the same picking station

$$V_{d,l_1} = V_{d,l_2} = ... = V_{d,l_n} \qquad \forall l_i \in O \tag{6}$$

4 SYSTEM DESIGN

The developed approaches are tailored to work in a realistic scenario and will be evaluated as such (see section 5). To cope with the complexity real-life scenarios provide, the implemented planning agents operate with limited knowledge.

First, the rating of the order-lines $l \in H$ will be based on the driving distance $d_p \subset d$ to the pickup nodes V_p. The remaining steps to complete the driving task are not considered for the rating as they do not depend on the assignment of the vehicle. Also, every order-line needs to be completed at some point of time. As a consequence, those steps have no influence on the overall system performance. We assume a strong correlation between driving distance and driving time. Second, for the assignment of orders to picking stations only orders are assumed to reserve capacity

- for which at least one driving job has been started yet and
- which have unsatisfied order-lines remaining.

The first assumption is trivial, as no order bin is required at the picking station before the first order-line has been delivered. The second assumption implies that any driving job which has already been started will reach a possibly capacity-critical picking station before the currently assigned one. Otherwise, the currently assigned will have to start a new attempt for delivery as no capacity for its order bin is available. A new attempt will result in a detour on G such that other vehicles may reach the picking station and some time passes until the current vehicle reaches the picking station again.

For all of the implementations, a blackboard architecture has been implemented for the multi-agent system. The blackboard contains information about the incoming customer orders (denoted as L). Also, information about the status of the order-line and the corresponding order are published on the blackboard. Depending on the approach, specific agents have read and write privileges for the blackboard. For example, an agent may annotate for an order-line to be completed and add an assignment o_S.

In this paper three approaches are described and evaluated:

- distributed planning with hybrid encoding (section 4.1)
- distributed planning with numbering concept (section 4.2)
- central planning with hybrid encoding (section 4.3)

For all of the implementations the ASP grounder and solver *clingo* (Gebser et al., 2014) in version 4.5.4 of the *Windows* build has been used.

4.1 Distributed Planning

Some versions of the distributed approach with hybrid encoding have been discussed in (Schieweck et al., 2016). We will describe the superior version of the encoding and its operation in the overall system in the following.

The implementation follows the trend towards distributed systems by increasing the vehicles autonomy and enabling them to take their own decision. This approach has been selected due to its high analogy to the current systems architecture, its flexibility and low expected computing times. As soon as a vehicle completes a driving task it requests a new driving job. Assuming full utilization of the system (which is the most critical state) a number of driving jobs is available. At that point of time, the vehicle makes the assignments for both l_v and o_S, if required. The vehicles planning agent extracts the relevant information from the blackboard and translates them to an ASP-instance (see Listing 1). The instance contains information about the available order-lines (`order_pos/3`), the current assignments of orders to picking stations (`order_pickst/2`) and the vehicles position (`veh_position/2`).

In the encoding the driving jobs (`pos/2`) are extracted in line 8. After, we ensure that the vehicle is assigned exactly one driving job in the atom `pos_veh/2`. In line 10 existing assignments of orders to picking stations are transferred to the new atom `as_order_pickst/2` to differ between new and existing assignments. If a driving job j is selected, at least one of its corresponding order-lines l must be assigned to the vehicle (line 11). The selection of the order-line requires assignment of the related order, if no assignment exists already (line 12). The assignment is then conducted in lines 13 and 14 with respect to the capacity restriction `c_pickst` of the picking stations.

```
1                                       instance
2  order_pos(69,33,436). order_pos(70,34,322). order_pos(76,36,241). order_pos(82,39,446).
3  order_pos(83,39,124).
4  order_pickst(33,2). order_pickst(34,3). order_pickst(36,1). order_pickst(39,8).
5  veh_position(4,573).
6
7                                       encoding
8                                    pos(P) :- order_pos(_,_,P).
9                     1{pos_veh(P,V): pos(P)}1 :- veh(V).
10                         as_order_pickst(O,S) :- order_pickst(O,S), pickst(S).
11     1{job_veh(Ix,V): order_pos(Ix,_,P)} :- pos_veh(P,V).
12                                 as_order(O) :- job_veh(Ix,V), order_pos(Ix,O,_),
                                                 not order_pickst(O,_).
13     1{as_order_pickst(O,S): pickst(S)}1 :- as_order(O).
14                                             :- pickst(S), c_pickst+1{as_order_pickst(O,S)}.
15
16                  dist(A,B,@distance(A,B)) :- veh_pos(_,A), pos(B).
17                              pos_veh(P,V,C) :- pos_veh(P,V), veh_position(V,A), dist(A,P,C).
18                                  numJobs(K) :- K=#count{Ix: job_veh(Ix,V)}.
19                           numOrderPickst(X) :- pickst(S), X=#count{O: as_order_pickst(O,S)}.
20                       maxNumOrderPickst(Y) :- Y=#max{X: numOrderPickst(X)}.
21
22                               #minimize{C@3 : pos_veh(P,V,C)}.
23                               #maximize{K@1 : numJobs(K)}.
24                               #minimize{Y@2 : maxNumOrderPickst(Y)}.
```

Listing 1: Excerpt of encoding for distributed planning with exemplary instance ($n = 5$).

In line 16 to 20 the rating of the driving jobs is implemented. Note that in line 16 we use the @-directive to compute the distance of the vehicle to the various bins with a Dijkstra-Algorithm (Dijkstra, 1959). The *Windows* build of *clingo* allows for the inclusion of Lua scripts in the ASP-encoding. Thus, hybrid programming is implemented. Some tests were conducted comparing the Dijkstra-algorithm with the A*-algorithm. For the prevailing structure, Dijkstra provided lower calculation times. The distance to the bins position is minimized with highest priority. Then, the maximum of the number of orders assigned to all picking stations is minimized, resulting in a leveled utilization. To be able to use eos-jobs properly the number of order-lines l completed with this single driving job j is maximized.

4.2 Location Numbering

The proposed approach is an evolution of the one described in section 4.1, resulting from first tests and insights to the aforementioned approach. While in the previous approaches hybrid programming is used, we try to eliminate the routing algorithm in the current approach. This is due to the high expected calculation time of the routing algorithm, especially when it comes to scalability. ASP is tailored for combinatorial, complex problems. The routing problem however, has been investigated many times and existing algorithms are expected to be superior to the implementation of the routing problem in ASP[2]. Thus, a different approach has been developed which makes use of the unidirectional structure of the routing graph. By testing and implementing we hope to find a good approximation of the optimal solution. Also, we hope to gain insights on how costly the routing algorithm is for the current planning task.

As a matter of fact, every driving job starts and ends in the rack. All of V_R are given names such that simple comparison yields an estimation for the ranking of the driving jobs with shortest traveling distance d_p. An example of the numbering for one rack is given in Figure 2. As a consequence, the multi-agent system is extended by a numbering agent which reassigns the names according to the concept. The numbering agent also has an ASP-encoding implemented which requires information about the length of the rack, the number of rack levels and the position of the rack entries and generates the 2-tuple $t = (id_{old}, id_{new})$.

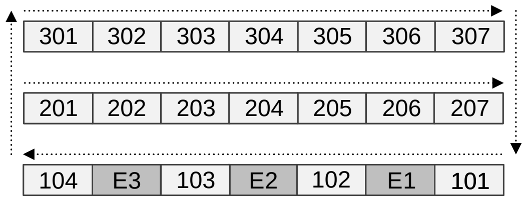

Figure 2: Example of numbering concept.

[2]Again, tests were conducted to indicate this.

```
1  level_pos(N,P/100) :- order_pos(N,_,P).
2  level_veh(V,P/100) :- veh_position(V,P).
3      numPickst(Max) :- Max=#max{S: pickst_node(S,_)}.
4
5     %case 1                    %case 2                     %case3
6     possiblePos1(N) :- ...     possiblePos2(N) :- ...      possiblePos2(N) :- ...
7     selectedBin1(N) :- ...     selectedBin2(N) :- ...      selectedBin3(N) :- ...
8
9      selectedBin(N) :- selectedBin1(N).
10     selectedBin(N) :- selectedBin2(N).
11     selectedBin(N) :- selectedBin3(N), not selectedBin2(_).
12
13       pos_veh(B,V) :- selectedBin(N), order_pos(N,_,B), veh_position(V,_).
```

Listing 2: Excerpt of encoding for distributed planning with numbering concept.

The vehicles planning agent has an encoding similar to Listing 1. Lines 16 and 18 are substituted by Listing 2. Also, the optimization statement in line 22 is no longer necessary. First, the rack levels of the order-lines $l \in H$ are calculated in lines 1 and 2. The number of the picking stations is expressed in the atom `numPickst/1` in line 3. Afterwards, the cases

1. the vehicle is not in the rack,
2. the vehicle is in the rack and at least one driving job $j \in H$ starts on the same level in driving direction of v and
3. the vehicle is in the rack and a driving job $j \in H$ is available.

If case I applies and the capacity restriction for at least one job of the position is fulfilled an atom `possiblePosI/1` is created. Multiple `possiblePosI/1` may exist for each `I`. The minimum number has the lowest traveling distance d_p and is transferred to `selectedBinI/1`. While case 1 excludes the other cases, 2 and 3 may occur in the same instance. Then, `selectedBin2/1` is assigned because the bin can be picked up on the way to the outside of the rack (line 10 and 11). Line 13 is the interface to the existing encoding and translates `selectedBinI/1` to `pos_veh/2`.

4.3 Central Planning

The distributed approaches fit into the existing trend towards local decision making of intelligent units. However, multi-agent systems rarely exist without any central agents which have the ability to coordinate the overall system (see e.g. section 2.2 with the current architecture). As described in chapter 2.2 the existing system contains a central entity for dispatching. A global optimum for a planning problem is expected to have an objective value at least as good as multiple local optima. On the other hand, the search for a global optimum raises higher complexity. The following approach has been developed to test the impact of the global optimum and the capability of ASP to find the optimum with satisfiable computing times.

An exemplary situation in which a central planning approach is superior to a distributed approach is depicted in Figure 3. Vehicle 1 requests a new driving task and has bin a and b available. Bin b induces a shorter distance and will be selected in the distributed approach, even though the difference between a and b is comparatively small. For vehicle 2 bin a remains for which it has to travel through the whole rack and use the lifts twice. A central planning approach can minimize the overall traveling distance for all vehicles. In this case, vehicle 1 is assigned bin a with a slightly larger traveling distance. As a consequence, vehicle 2 can reach bin b which induces considerably less traveling distance.

For the current approach a central dispatching agent is added to the multi-agent system. Just like in the previous implementations the agent has a horizon H of order-lines available from which it conducts the assignments l_v and o_S. Once a vehicle completes a driving job it tries to find a new driving job which is assigned for itself on the blackboard. If no assignment

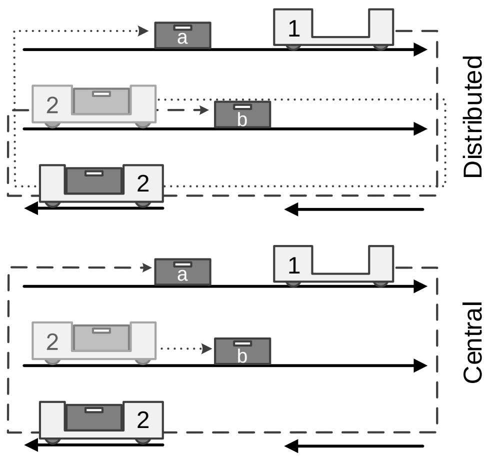

Figure 3: Distributed vs. central planing.

```
1                                    :- pos(P), 2{pos_veh(P,V)}.
2                  sposition(V,Pos)  :- occupied(V,Pos).
3                  sposition(V,Pos)  :- veh_position(V,Pos), not occupied(V,_).
4 dist(SPos,TPos,@distance(SPos,TPos)) :- sposition(_,SPos), pos(TPos).
5                    veh_drive(V,D)  :- pos_veh(P,V), sposition(V,SPos), dist(SPos,P,D).
6
7                     #minimize{D@3 : veh_drive(V,D)}.
```

Listing 3: Excerpt of encoding for central planning.

can be found a new planning cycle is initialized[3]. At that point of time, the planning agent creates a plan for all of the vehicles v in the system and assigns exactly one j for each v and publishes the result on the blackboard. The agent requires the additional information about the status of each vehicle. The status is defined by the occupation and if an occupation exists, the final vehicle position induced by driving job j.

For the comparability of the implementations, n is redefined. Note that a minimum of F order-lines must be available in H to allow for one assignment l_v for each vehicle v. Also, the amount of undesirable, unassigned order-lines in H shall stay the same. Thus, the central planning agent has a number of

$$n_{cen} = n + F - 1 \tag{7}$$

order-lines available.

The central planning agent has a similar encoding as the distributed agents in Listing 1. The differing lines are displayed in Listing 3. In line 1 we ensure that each vehicle receives only one driving job. If a vehicle is occupied, the atom `occupied/2` is added to the problem instance and the starting point for the next job is defined with `sposition/2` (line 2). If the vehicle is not occupied, we assume that it does not move until the next driving job starts (line 3). The necessary distances are computed in line 4. Again, a routing algorithm is used. If an assignment is made the vehicle has to travel the distance `D` which is noted in the atom `veh_drive/2` (line 5). The overall traveling distance to the starting positions is minimized in line 7.

5 EVALUATION

The approaches which are described in section 4 are evaluated in the following. First, the experimental design is described (section 5.1). Second, the results are described and interpreted (section 5.2).

[3]The planning procedure may be started even before. This is the protocol which has been implemented to the simulation.

5.1 Experimental Design

For the evaluation a *Demo3D* simulation model has been created. The simulation model has the purpose of a development and evaluation environment. For being able to validate the functionality of the approaches, their behavior inducted by the encodings could be tested instantly. Also, an interface between the simulation environment and the employed ASP-grounder and -solver *clingo 4.5.4* has been created. *Demo3D* was selected due to its programming structure which is close to agent-based systems. Single entities are assigned proprietary C#-scripts which exchange information with messaging protocols. Thus, the simulation could be implemented similarly to a real-world application. The level of abstraction of the simulation is relatively low. Just like the real vehicles, their digital replications possess proximity sensors which slow the vehicle down (proximity < 3 m) and let it come to an halt eventually (proximity < 1 m).

Working towards comparable results not only the approaches described in section 4 were implemented but also counterparts with classical (imperative) programming as well as the *FIFO* approach. The quality of the implementations is in the following quantified by their improvement compared to the *FIFO* approach and indicated by Δ. The experimental design consists of a *basic system* from which single parameters are varied. The basic system has one rack with a capacity of 500 bins, $S = 5$ picking stations, $F = 5$ vehicles, uniformly distributed demand among the skus

Table 1: Data of simulated system.

Parameter	Value	Unit
Vehicle		
Speed floor	1	m/s
Speed rack	2	m/s
Acceleration/Deceleration	0,5	m/s^2
Time loading/unloading	4,5	s
Lift		
Speed	2	m/s
Acceleration/Deceleration	2	m/s^2
Picking time per order-line	5	s

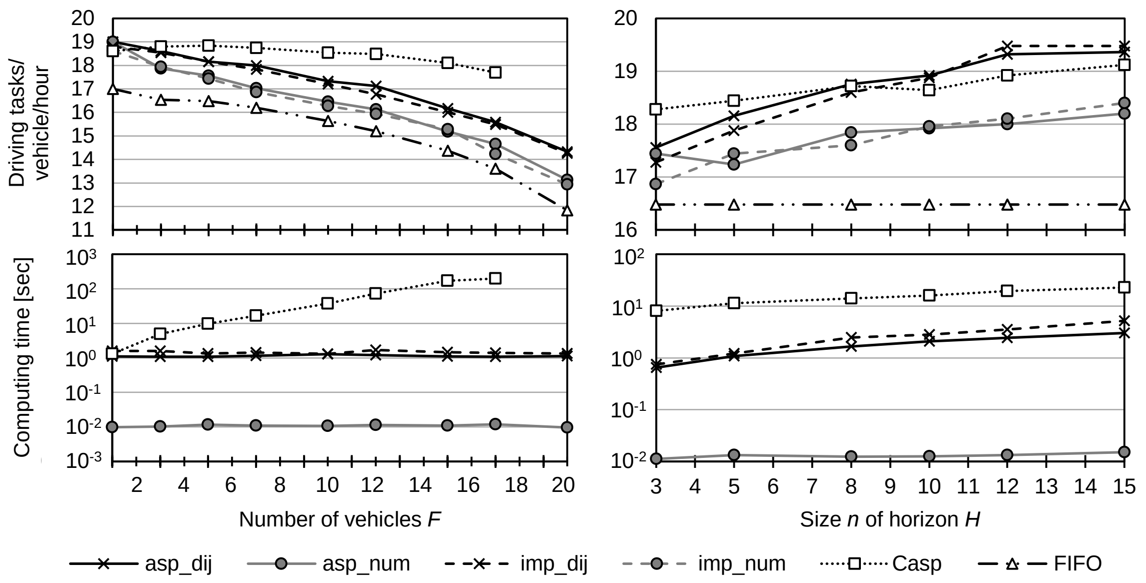

Figure 4: Evaluation of the approaches.

and a horizon with $n = 5$ order-lines. The parameter variations were tested for five hours of simulation time each. The central and distributed planning agents are granted a computing time of 200 seconds to find the optimal solution[4]. After this time, three options may occur:

- An optimal answer set has been found and the computation has been terminated at a time < 200 seconds.
- The solver has not terminated the computation procedure yet and
  - an answer set has been found. The answer set may or may not be optimal.
  - no answer set has been found.

If no answer set has been found the simulation run was terminated.

5.2 Experimental Results

Some key results of the simulation runs are depicted in Figure 4. Note that the computing time (bottom) is presented with a logarithmic y-Axis. We define the number of driving tasks which have been completed by one vehicle per hour on average as the *performance*.

Generally, the performance of a vehicle decreases with a growing number of vehicles in the system (top left). This is due to blocking and congestion of the vehicles on the routing graph. For the variation of the number of vehicles the highest average performance was achieved by the central approach (*Casp*, $\Delta 19\%$). The distributed approaches with hybrid programming achieve $\Delta 13\%$ (ASP, *asp_dij*) and $\Delta 12\%$ (classical programming, *imp_dij*). With the distributed approaches with numbering concept an improvement of 7% (ASP, *asp_num*) and 6% (classical programming, *imp_num*) is achieved. For the distributed approaches the computing time (bottom left) is constant with increasing vehicle number as the instance of every single planning agent is independent of this parameter. No measurable computing times for *imp_num* and *FIFO* were recorded. The computing time for *asp_num* was 0.01 seconds on average. For *asp_dij* and *imp_dij* the times were on average 1.13 and 1.45 seconds, respectively. The increase is caused by the routing algorithm. Separate experiments revealed that the average computing time for the routing on the graph of the basic system is roughly 0.2 seconds. The computing time of the central approach increases almost exponentially with the number of vehicles. As a consequence, the experiments with 20 vehicles did not produce results due to the 200 seconds limit.

With growing size *n* of the horizon *H* the performance (top right) of the vehicles increases. For all of the simulation runs, the central ($\Delta 13\%$) and the distributed, hybrid approaches ($\Delta 13\%$ ASP, $\Delta 12\%$ imperative) achieve the highest performance followed by *asp_num* and *imp_num* ($\Delta 6\%$ each). Remarkably, the performance of *Casp* increases by a smaller amount than the other approaches, even though a central approach is expected to be superior. The limited knowledge has an impact in those scenarios. While in

[4]This is the time a vehicle needs approximately for one driving task.

the distributed approaches the planning is conducted exactly at the time needed, the central planning agent has to take a wider look into the future. However, the limitation of the knowledge stays the same due to rapidly growing complexity of the problem (e.g. in an state-action encoding with consideration of time points). Thus, in the central approach a capacity of a picking station may be considered as occupied by the time of the planning while in real-life operation it has been released in due time. The computing time (bottom right) grows steadily for all of the approaches which use the routing algorithm. The highest increase is noted for *Casp* due to the redefinition of the horizon size n_{cen}. For the distributed approaches the increase is roughly constant to the 0.2 seconds which are required to compute the distance to every $l \in H$. The computing time of *asp_num* only increases from 0.009 seconds to 0.011 seconds with a horizon size of 3 and 15, respectively.

Generally, a slight decrease of performance occurs from the ASP implementations to their imperative counterparts. At this point, the flexible structure of the ASP encodings proves to be advantageous. In the ASP approaches it is possible to prioritize the eos-jobs such that, if two driving jobs induce the same distance and utilization, the driving job which completes the most order-lines will be selected. The imperative implementations have a less flexible structure in which first an order-line is selected and if possible, more order-lines with the same bin are added:

1. Setup of horizon H.
2. Calculate distances d_p.
3. Choose minimum distance order-line l_{min}.
4. Search eos of l_{min} in H with $d_{p,eos} = d_{p,min}$.
5. Assign station(s) o_S with respect to c_S and u_{avg}.
6. (a) If no o_S could be assigned, eliminate l_{min} from H, clear $o_S \forall l_{min}$ and go to (2).
   (b) Else, assign order-line(s) l_v.

In the case mentioned above weather the order-line with or without eos-job is selected is random. Also, one might note that the priority of eos-jobs is third in the ASP approaches and they are searched for second in their counterparts. This is due to the fixed sequence in imperative programs. In this case, all of the potential l must be found to verify possible c_S.

Furthermore, the computing time of the approaches with the numbering concept is remarkably low. Unfortunately, the performance of the approaches is low as well, even though an improvement compared to the FIFO experiments could be achieved. We find the reason for this in the rating of the storage positions on different levels. While the hybrid approaches always select the positions which are the easiest to reach, *asp_num* and *imp_num* prefer the positions on the lower levels because they have lower numbers (see Figure 2). As a final note, the computing times of the imperative counterparts are certainly induced by some degree by the simulations interface to the routing algorithm. Generally, we expect a well programmed imperative code to be faster than a general purpose ASP solver. However, especially for systems which are designed to work in volatile environments the flexibility and comfort of programming is key which is remarkably good for ASP. This can be elucidated with the small length of the encodings in this paper as well as the small adaptions necessary between the approaches.

6 CONCLUSION AND OUTLOOK

The presented work combines answer set programming with the innovative field of application of intelligent vehicles in an industrial setup. The approaches are tailored to work in a real-life environment and are evaluated as such. The general approach of allowing the planning agents to select driving jobs from a horizon achieves a considerable improvement of performance without any physical adjustment of the system. The approaches to the planning task do not cause an immense complexity. However, for operational planning even seconds become valuable. For most cases, central planning achieved a higher performance than the distributed approaches. The limited knowledge plays a critical role for the performance, especially when the planning horizon is large. Answer set programming as a method promises great capabilities in systems which are made to work in changing and volatile environments. This is due to good computing times and especially easy modeling and high flexibility.

For future steps we recommend the evaluation of further planning tasks to be implemented with ASP and combinations of those. As a final step, the most promising approaches should be implemented and tested with existing vehicles.

REFERENCES

Bartholdi, J. J. and Platzman, L. K. (1989). Decentralized control of automated guided vehicles on a simple loop. *IIE Transactions*, 21(1):76–81.

Benincasa, A. X., Morandin, O., and Kato, E. R. R. (2003). Reactive fuzzy dispatching rule for automated guided vehicles. In *IEEE International Conference on Systems, Man and Cybernetics*, pages 4375–4380.

de Koster, R. B., Le-Anh, T., and van der Meer, J. (2004). Testing and classifying vehicle dispatching rules in three real-world settings. *Journal of Operations Management*, 22(4):369–386.

Dijkstra, E. W. (1959). A note on two problems in connexion with graphs. *Numerische Mathematik*, 1(1):269–271.

Dovier, A., Formisano, A., and Pontelli, E. (2009). An empirical study of constraint logic programming and answer set programming solutions of combinatorial problems. *Journal of Experimental & Theoretical Artificial Intelligence*, 21(2):79–121.

Foundation for Intelligent Physical Agents FIPA (2002). Fipa contract net interaction protocol specification.

Gebser, M. (2013). *Answer set solving in practice*. Morgan & Claypool, San Francisco, CA.

Gebser, M., Kaminski, R., Kaufmann, B., and Schaub, T. (2014). Clingo = asp + control: Extended report.

Gelfond, M. and Lifschitz, V. (1988). The stable model semantics for logic programming. *Journal of Symbolic Logic*, 57(1):274–277.

Gelfond, M. and Lifschitz, V. (1991). Classical negation in logic programs and disjunctive databases. *New Generation Computing*, 9(3-4):365–385.

Guizzo, E. (2008). Three engineers, hundreds of robots, one warehouse. *IEEE Spectrum*, 45(7):26–34.

Kamagaew, A., Stenzel, J., Nettstrater, A., and ten Hompel, M. (2011). Concept of cellular transport systems in facility logistics. In *5th International Conference on Automation, Robotics and Applications*, pages 40–45.

Le-Anh, T. and de Koster, R. B. (2006). A review of design and control of automated guided vehicle systems. *European Journal of Operational Research*, 171(1):1–23.

Mayer, S. H. (2009). *Development of a completely decentralized control system for modular continuous conveyor systems*. Dissertation, Karlsruhe University, Karlsruhe.

Saidi-Mehrabad, M., Dehnavi-Arani, S., Evazabadian, F., and Mahmoodian, V. (2015). An ant colony algorithm for solving the new integrated model of job shop scheduling and conflict-free routing of agvs. *Computers & Industrial Engineering*, 86:2–13.

Schieweck, S., Kern-Isberner, G., and ten Hompel, M. (2016). Using answer set programming in an order-picking system with cellular transport vehicles. In *IEEE International Conference on Industrial Engineering and Engineering Management*, pages 1600–1604.

Vis, I. F. A. (2006). Survey of research in the design and control of automated guided vehicle systems. *European Journal of Operational Research*, 170(3):677–709.

Yang, J., Jaillet, P., and Mahmassani, H. (2004). Real-time multivehicle truckload pickup and delivery problems. *Transportation Science*, 38:135–148.