Agent-based Dynamic Transport Optimization

Klaus Dorer and Monique Calisti
Whitestein Technologies AG
Gotthardstrasse 50
8002 Zürich, Switzerland
Phone: +41 1 205 5500
Fax: + 41 1 205 5509
{kdo,mca}@whitestein.com

Abstract

In this paper we present our agent-based approach conceived to solve dynamic multi-vehicle pickup and delivery problems with soft time windows. While many of the existing research frameworks have been focusing on reaching close-to-optimal solutions, the central theme of our work is the ability to solve real world sized problems in near real time. In order to describe our solving approach and its performance, we introduce a real case scenario in which a logistics company has to dynamically optimize a set of transportation requests. The aim is to show how our agent-based solution produces significantly better results than achieved with manual optimization of professional dispatchers.

1. Introduction

Today, most of the big logistics companies are facing the crucial problem of managing continuously growing amounts of transportation requests/orders and vehicles under increasingly tight time constraints. The main reason is that the transportation market itself has been significantly expanding over the last years and a number of factors determined the merge of various transportation businesses of two or more logistics companies (e.g. Deutsche Post, AEI and Danzas, UPS and Fritz, Kuehne & Nagel and USCO, Exel and Mark VII). The problem is usually solved by distributing the transportation business across a number of regional dispatching centers and increasing the number of dispatchers. This process, however, is complicated by the fact that a considerable amount of communication and remote coordination is necessary to detect opportunities of combining transportation between regional dispatching centers. In particular, delays and coordination costs can heavily impact the ability of optimizing resource consumption (i.e., deployed vehicles, number of dispatching centers) and finally fulfil all orders as expected. Today, for instance, the average utilization of vehicles in charter business is fairly low (55% in the example described in section 4). In this sense, we argue and show that intelligent transport optimization systems have the great potential of reducing overall transport costs by improving the coordination process of distributed dispatchers and the resource consumption (i.e., better usage of available resources than allocation of additional ones).

The problem of organizing the pickup and delivery of transport requests in a timely fashion by deploying multiple vehicles is known in the operations research literature (and in the following) as dynamic m-PDPSTW. We give a short introduction to the dynamic m-PDPSTW problem in section 2. Good and more extensive overviews of the problem can also be found in [14, 11, 4, 1].

Several agent-based approaches have been proposed to deal with this kind of dynamic optimization problems, e.g., [3, 9]. The main motivation comes from the fact that multi-agent systems ideally reflect the distributed nature of the problem and are able to deal with the dynamics of planning and execution in near real-time settings. The work described in this paper introduces an agent-based optimization system that focuses on and deals with real world transportation problems. Real world transportation problems usually differ from 'standard' problem instances in the number of constraints to be obeyed [16] and the instance size. In section 2.2, we describe the constraints that need to be taken into account in order to obtain delivery routes that are drivable in real world. Section 3 describes the optimization algorithms and agent-based design that enables to deal with problem instances with thousands of transportation requests. As of today, we are aware of only few research works that deal with such large sized problem instances, e.g., [7, 6, 5]).

The remainder of the paper is organized as follows: Section 2 covers the problem formulation, the specific constraints and the cost model used for the empirical evalua-

tions. In section 3, our agent-based optimization approach is explained. Section 4 gives some empirical results of a real world optimization problem. Finally, in section 5 we discuss possible future work directions before concluding.

2. Dynamic m-PDPSTW

The dynamic multi-vehicle pickup and delivery problem with soft time windows (dynamic m-PDPSTW) [13, 14] consists of finding optimal routes for serving transportation requests of customers. The problem is called *dynamic*, as opposed to its static version, if the transportation requests are not all known in advance. In this case, new requests are acquired consecutively during the optimization process itself. An even more dynamic version also deals with changes that can occur to transportation requests and/or transportation capacity in a real-time fashion. These changes reflect reactions to information the driver gets when picking up specific orders or by unplanned situations like traffic jams. The problem is 'single-vehicle' if all transportation requests are served by a unique vehicle. Here, we deal with a 'multi-vehicle' problem where multiple vehicles can be used for the delivery. The vehicles may be of different type and have different capacities. As opposed to vehicle routing problems [10, 14], in *pickup and delivery problems* (PDP), vehicles do not necessarily start or end in the same location. Transportation requests may have the same but have usually different pickup and delivery locations. The pickup and delivery of orders has to occur within a specific time window, even though time constraints can be possibly violated up to some tolerated degree. These kind of problems are called PDP with soft time windows.

The *solution* of an m-PDPSTW consists of a set of routes including a schedule that specifies the times at which the vehicles have to be at selected locations. The result is also called *delivery plan* and its quality or goodness is given by an *objective function*, which in our case is a cost-based function (see section 2.3 for more details). In our framework, the term *node* is used to indicate the combination of location and schedule, i.e. arrival and departure time of the vehicle at the location. A *leg* is the way between two nodes. A *tour* is a sequence of nodes a vehicle visits. The vehicle is empty at the beginning and at the end of a tour, but not while the tour has not been completed. Tours are sometimes also called active periods or mini-clusters in literature [11]. A *route* is a sequence of $n \geq 1$ tours done by a vehicle. The terms *order* and *transportation request* are used synonymously for a customer request to transport a good from a pickup location (within a specified pickup time window) to a delivery location (within a specified delivery time window).

2.1. Initial Information

As mentioned above, for dynamic settings the transport requests are not all available when starting the route planning and delivery scheduling process, but can arrive consecutively. Some transport requests may have already been served, while new requests arrive. Every transport request specifies:

- Order type.
- Volume (in loading meters).
- Weight.
- Pickup location.
- Pickup time window.
- Delivery location.
- Delivery time window.
- Time at which the order is known to the system.

Additional information is also available for each vehicle:

- Vehicle type.
- Capacity (volume, in loading meters).
- Capacity (weight).
- Start location.
- Availability time (at start location).
- Tariff.

The tariff indicates a cost class to which the vehicle belongs (see section 2.3). A mapping function defines which order types fit to which vehicle type.

2.2. Constraints

The optimization algorithm has to obey a number of constraints during the calculation of routes. Constraints are classified as *hard constraints* and *soft constraints*. Hard constraints express conditions that must be verified. Soft constraints express conditions that might be violated to some degree. The former category includes:

- Load constraints:
  - Precedence (pickup has to be before delivery);
  - Pairing (pickup and delivery have to be done by the same vehicle);
  - Capacity limitation of a vehicle;
  - Weight limitation of a vehicle;
  - Order type and vehicle type compatibility.
- Time constraints:
  - Earliest pickup - maximal allowed early time;
  - Latest pickup + maximal allowed delay;

  - Earliest delivery - maximal allowed early time;
  - Latest delivery + maximal allowed delay;
  - Maximum time a vehicle may wait at a location for new transport requests;
  - Maximum duration of a tour;
  - Maximum duration of a route;
  - Lead time for ordering a vehicle;
  - Maximum driving time of a driver.
- Maximum number of locations on a tour.

Soft constraints violations produce violation costs that impact the optimization process, but that are not included in the final overall solution costs. Violation costs are introduced so that constraints are only violated if it is 'worthwhile', i.e. only if the cost savings achieved by violating a soft constraint exceed the violation costs the constraint violation is allowed. A soft constraint is defined in terms of a start value above (below) which the condition is soft-violated, an end value above (below) which the constraint has a hard violation, fix violation costs assigned if the constraint is (soft) violated and variable violation costs that grow proportional to the amount of soft violation. The fix violation costs can be used to control the number of violations. The variable violation costs ensure that the amount of constraint violation is kept low (see section 2.3). The following soft constraints have been considered in the evaluated domain:

- Earliest pickup time;
- Latest pickup time;
- Earliest delivery time;
- Latest delivery time.

2.3. Cost Model

The major concern of logistics companies is to reduce their costs [2]. Consequently, the objective function used to evaluate the result of optimization is cost-based. The cost model here was chosen on the basis of real data to be able to compare the results of agent-based m-PDPSTW optimization with the real transport plan created manually by professional dispatchers. The cost model distinguishes between three kinds of costs: variable, fix and violation costs.

Variable costs are assigned depending on the length of a route, the amount of transported load and the start location of the specified route. The variable cost of a route is calculated as:

$$c_{var} = \sum_{tours} c_{km} * dc_{route} \quad (1)$$

with c_{km} the distance cost computed as:

$$c_{km} = d_{tour} * l_{max} * tariff(region, d_{tour}, l_{max}). \quad (2)$$

where d_{tour} is the driven distance of the tour, l_{max} is the maximal load (volume) that is transported on a leg of the tour, $tariff$ is the cost class which the tour belongs to. In the real case we analyzed, 7 tariff regions within Germany, 3 distance classes for each region and 16 load classes for each region and distance class have been defined. The values for $tariff$ used during evaluation where derived from historic data. dc_{route} is a discount that is granted to special routes. A discount is granted if a route contains multiple tours (tramp tours). This reflects the issue that in charter business cheaper offers are given when multiple consecutive tours can be offered to a subcontractor. The empty driven kilometers between two tours have to be limited (e.g., 70 km), and the period of time between the end and start of two tours is limited (e.g., 4 h). A higher discount is granted if a route with multiple tours ends close to (< 100 km) the start location (back tours). This saves work of the subcontractor in looking for freight (and driver) for a back tour of the vehicle.

Fix costs may be assigned to a vehicle on a daily basis. This means that each day a vehicle is used fix costs are assigned. In the analyzed case, fix costs were replaced by minimum costs. If the vehicle's variable costs are below a minimum threshold, the final costs of the route are the minimum costs independent of the duration of the usage. The total cost of a route is then calculated as:

$$c = \max(c_{min}, c_{var}) \quad (3)$$

where c_{min} is the configured minimum costs of a vehicle.

Violation costs are not included in the total cost of a route, but are used by the optimizer during the solving process. They arise when soft constraints are violated, as mentioned above. Each soft constraint violation causes a fix violation cost independent of the amount of violation. This way the user can influence the number of constraint violations. If, for example, the fix violation cost is configured to be 100 cu[1], the cost savings the optimizer achieves by allowing a soft constraint violation has to be above 100 cu. Violating a constraint for less cost savings is not possible. A soft constraint violation also causes a variable violation cost increasing with the amount of violation. If, for example, the variable violation cost is set to 50 cu/h then the cost savings for a 3 hour delay has to be at least 150 cu. This way the average amount of violation can be controlled.

3. Agent-Based Optimization

The optimization process is incrementally reflecting the dynamics of the underlying m-PDPSTW settings. Whenever a new transport request is available to the system, the

1 cu stands for currency unit

```
For all vehicles
  Check availability
  For all pickup insertion points
    For all delivery insertion points
      Check constraints and schedule
      Calculate quote
      If cheapest quote -> store quote
Assign order to cheapest quote
```

Figure 1. Order insertion algorithm.

current delivery plan is updated. This is done in a two-phase-based approach. First, a new valid solution is generated including the new transportation request. Then, the obtained solution is improved by cyclic transfers of transportation requests. The next two sections explain these two steps and section 3.3 describes how these algorithms can be shared across multiple agents.

3.1. Solution Generation

The first step when receiving a new order is the generation of a new valid solution. The algorithm used for this is a sequential insertion of transportation requests [7]. For all available vehicles it is checked if they are able to transport the order and what additional costs arise. One of the available vehicles is a new empty charter vehicle assumed to be available at the pickup location. The process of sequential insertion is summarized in Figure 1. Each time a transportation request is added to a route either 0, 1 or 2 new nodes have to be added depending on whether the pickup and/or the delivery nodes of the request are already part of the route or not. In this way, multiple pickups and deliveries per location are possible. Based on the insertion algorithm, it is also possible that a vehicle visits the same location several times. Finally, for all combinations of existing nodes a quote is generated for inserting a new pickup and delivery node.

Each time a new node is added to a route the order of nodes could be optimized. This corresponds to solving a traveling salesman problem (TSP) and is intractable in the general case [8]. In our approach, we do not solve the general TSP. Instead, the order of existing nodes is never changed and for every new node all points between existing nodes are checked for insertion of the new node. In an empirical evaluation with a case of 200 orders the comparison of an optimal algorithm for solving the TSP and our approximation algorithm has shown that only in 2 cases our algorithm produced sub-optimal routes with just 17 additional kilometers ($< 0.2\,{}^{0}/_{00}$ of the complete distance traveled). Focusing on real world sized problems this gap from the optimum is considered to be acceptable compared to the gain

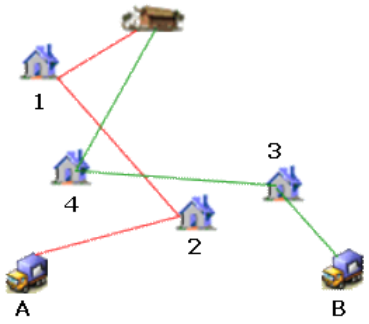

Figure 2. Example for a suboptimal solution of the insertion algorithm for 4 orders.

in runtime performance ($\frac{n!}{2n}$ possible solutions for the traveling salesman problem compared to $\frac{n(n+1)}{2}$ possible solutions for the simplified insertion of an order with n the number of locations).

If no vehicle can transport the order, which means all vehicles would have to break hard constraints, the orders service level is lowered. This means the order may be transported with configurable soft constraint violations allowing violations of pickup and delivery times. If still no vehicle is found the order remains unallocated. This is usually the case if order data is invalid in a sense that the problem is over-constrained (e.g. impossible driving times, volume or weight above limits of vehicles).

3.2. Optimization

Sequential insertion with requests for quotes to all vehicles potentially produces suboptimal solutions, see the example in Figure 2. Order 1 is the first arriving at the system and it is assigned to vehicle A's route. Order 2 is again optimally assigned to vehicle A's route since it produces least additional costs (and kilometers). When order 3 arrives vehicle A is fully loaded. Therefore, a new vehicle B is used for order 3 and later order 4.

To get closer to the optimal solution a further optimization step is performed. It is done by cyclic transfers between vehicles [15]. A cyclic transfer is an exchange of orders between routes. More specifically in a $b-cyclic\,k-transfer$ k orders are cyclically exchanged between b routes [11]. In our case k defines the maximum number of orders transferred. A 3-cyclic 3-transfer might then be moving 3 orders from first to second route, 1 order from second to third and 2 orders from third to first route. Figure 3 shows how the suboptimal example of Figure 2 is improved by a 2-cyclic 1-transfer. Order 2 is transferred from route A to B while order 4 is transferred from route B to A. The optimization procedure has to determine which b-cyclic k-transfers should be

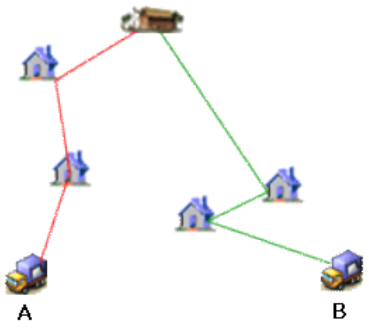

Figure 3. Optimal solution after a 2-cyclic 1-transfer.

tried. Adding a new transport request changes a single vehicle's route r_1 (orders with a volume above a vehicle's capacity are assumed to be split before reaching the optimizer). Assuming that at a time t before the order insertion an optimal solution is found the only point for improving transfers is r_1. Therefore, r_1 is initiating transfer requests to all other vehicles. The number of k-transfer requests between two routes is determined by the number of groups g of orders of a route r with at most size k $|g_{r,k}|$. All combinations of pairs of groups between the two routes produce a transfer request. The number of requests therefore grows considerably and limits the applicability of the algorithm to domains with few loads (< 10) per vehicle or to small k values. From the so created transfer neighborhood the most cost-saving b-cyclic k-transfer is performed, i.e. the transfer that reduces the overall costs most. This changes route r_1 and b other routes r_i. This hill climbing process is continued with all changed routes until no more cost-saving exchanges can be done. In our case, only 2-cyclic transfers where performed for performance reasons (see section 4.3). The workflow of a 2-cycle k-transfer hill climbing algorithm is described in Figure 4.

```
Store changed route r1 in list l
 For all other routes ri
  Check compatibility of vehicles
  For all g r1,k
    For all g ri,k
      Perform exchange
      Check r1 constraints and schedule
      Check ri constraints and schedule
      Calculate cost savings
      If highest cost savings
        Store option
 If option found
  Perform best exchange
  Add changed routes to l
 Else
  Remove r1 from l
Repeat until l is empty
```

Figure 4. 2-cycle k-transfer algorithm

3.3. Agent Design

Solving a dynamic m-PDPSTW can be distributed across multiple agents. This is desirable mostly to achieve scalability of performance with growing sizes of problem instances. Several agent designs are possible to distribute the work.

The most radical one is to represent each vehicle by an agent [3, 9]. Solution generation by sequential insertion is then handled by a contract-net interaction protocol [3]. The optimization algorithm could be changed to not look for the best of all possible transfers, but trigger a transfer between two vehicles whenever it improves the objective function. This reflects the issue that multiple vehicle's routes might change concurrently. Therefore, deciding for a transfer only after having asked all other vehicles has a higher probability that this transfer is then no longer possible. In this changed optimization algorithm the hill-climbing is no longer along the steepest slope and possibly more transfers are necessary to get to the same solution. Also, the optimization may end in a different local optimum. The mechanism of initiating transfer requests remains the same except that no list is necessary to store the changed vehicles (routes) for sequential optimization. Instead vehicle agents with a changed route start an optimization process in parallel to the other active vehicle agents.

The advantage of this fully distributed approach is its fine granularity and high scalability. The main disadvantage stems from a considerable overhead in computation time and resource usage. The overhead in computation time is mostly due to more expensive agent communications compared to simple java method calls in a centralized solution. The overhead in resource usage depends on the footprint of an agent in the used agent system.

The agent design chosen for our work reflects the way logistics companies today manage the complexity of this domain. Transportation business is usually divided into regions/clusters. Transportation requests arriving at a cluster are first tentatively allocated and possibly optimized within that cluster. If the order's pickup or delivery location is in a different cluster, the other cluster is also informed and asked to handle the request if it can do it in a cheaper way. In our framework, distinct agents represent different regional clusters. All vehicles starting in the region of an agent are managed by its `AgentClusterManager`. Incoming transport requests are distributed by a centralized `AgentDistributor` according to their pickup location

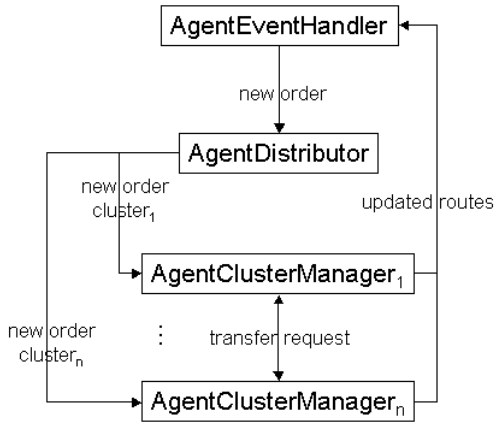

Figure 5. Cluster-based agent design.

(see Figure 5). Sequential order insertion can be done as described in section 3.1. The only difference is that 'all vehicles' in this case restricts to all vehicles of a cluster. A sensible extension is to forward a transportation request to another cluster in case it may not be transported within its cluster. The quality of the solution is expected to decrease with the number of defined clusters since order insertion does not take vehicles of other clusters into account. This has to be 'repaired' by optimization transfers. The optimization algorithm within a cluster is the same as described in section 3.2. Additionally, vehicles with routes spanning over other clusters may also initiate transfer requests among clusters. This can be achieved by adding a vehicle temporarily to the list of vehicles in another cluster and remove it from the original cluster. Then it will be part of the normal optimization process and can be moved back afterwards.

The main advantage of this design is its direct correspondence to today's business and its good scalability. Its computational overhead is also much lower than with the first mentioned approach. The disadvantage compared to a centralized and the above solution is that optimization within a cluster and among clusters has to be handled slightly differently. The solution quality could turn out to be worse compared to a fully centralized approach since information is not globally available.

4. Empirical Results

The empirical tests were run for a European logistics company with the aim of evaluating the potential cost savings of the introduction of a transport optimization system. The dataset contains roughly 3500 real-business transportation requests. The constraints and cost model have been used as described in section 2. Parameterization was set up so that the resulting routes where accepted by the dispatchers as realistic.

	Savings (agent-based)
cost	11.7%
kilometer	4.2%
vehicles	25.5%

Table 1. Savings achieved by agent-based compared to manual dispatching.

The primary goal of the logistics company has been to reduce their costs. Secondary or sub-goals have been to reduce the number of used vehicles and the amount of driven kilometers, and increase the utilization of the vehicles.

The agent server used for the evaluations is the Living Agents Runtime System (LARS) of the Living Systems GmbH. Tests were run on a 2 GHz Pentium machine running on Linux. For the results presented in section 4.5, four 800 MHz Pentium PCs running also Linux were used.

4.1. Comparison to Manual Dispatching

A number of measures have been taken to make the results of agent-based optimization comparable with the results achieved by the professional dispatchers. We used the same underlying geo-coding information system (distance and drive time information) as was used by the dispatchers. The average cost values for tariff classes (see section 2.3) have been obtained by the real costs that incurred in the corresponding tariff classes. Soft time windows and therefore soft constraint violations were only allowed if order data did not allow an in-time pickup or delivery. The resulting delivery plan was checked by dispatchers for feasibility and drivability.

Table 1 shows the comparison of the results. A total of 11.7% cost savings was achieved, where 4.2% of the cost savings stem from an equal reduction in driven kilometers. Another 2.2% is achieved by increasing the number of cost-saving tramp traffics (see section 2.3) by 380%. The rest stems from buying cheaper vehicles. The cost-based optimization prefers routes that start in cheap regions. An additional important achievement, is that the number of vehicles used is 25.5% lower compared to the manual solution. This is due to a higher utilization of the vehicles and an on average longer usage of a single vehicle. In this sense, the cost savings would even be higher if fix costs for the vehicles would arise, which is not the case in charter business, but possibly in other transportation settings.

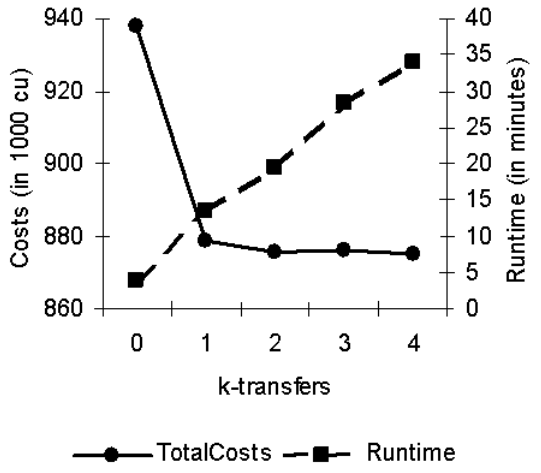

Figure 6. Influence of transfer parameter k on result quality and runtime.

4.2. k-transfers

The evaluated logistics business was a so called *part load business*. This means that multiple orders are usually loaded on a vehicle. Therefore the optimization result should improve with a growing number k of orders involved in a single transfer. Figure 6 shows the optimization results for the above described example when growing k. $k = 0$ means that no transfers are allowed. The corresponding result is therefore the result of the sequential insertion algorithm described in section 3.1. For $k = 1$ a single order may be moved from one vehicle to another or two vehicles may exchange a single order. The achieved cost savings over $k = 0$ is 6.7%. Another 0.3% savings are achieved with $k = 2$ and 0.1% with $k = 4$. For $k = 3$ the result is minimally ($< 1\,‰$) worse compared to $k = 2$. This happens since all results are potentially local optima.

The runtime for optimization of the orders grows as expected with increasing k. The almost linear growth of runtime can be explained by the average number of orders per vehicle $|\bar{o}| = 1.7$. This means that the number of candidate vehicles with 2 or more orders is getting increasingly small with growing k. The absolute runtime of 13:29 minutes for this data emphasizes the near real time property of our optimization approach. In particular real time events like order data change or delays of vehicles are usually inserted and optimized within one second.

4.3. b-cycles

The above results were obtained by using 2-cyclic transfers of orders. There are situations where no exchange between two vehicles may produce cost savings, but exchanges between three or more vehicles will. To estimate the gap in solution quality of 2-cyclic transfers to 3-cyclic transfers we generated a set of 10 million random transport situations with 3 vehicles containing one order each. Each situation was first optimized doing all possible 2-cyclic exchanges with cost savings. The solution was then compared with both possible 3-cyclic exchanges.

An additional optimization was just possible in 0.04% of the total cases saving only 0.02% costs. The gap should be even smaller for $b > 3$, although we did not test it. The increase in runtime for checking b-cyclic exchanges would on the other side be considerable. The additional gain is therefore not considered to be achievable in near real time.

4.4. Soft Time Windows

Some logistics companies today offer different service levels for the delivery of transport requests. They offer the customer the choice of a more expensive 100% in-time pickup and delivery or different cheaper levels with possible delays. This can be modeled using soft time windows.

To estimate the impact of soft time windows a series of optimization runs with increasing allowed delay for pickup and delivery of orders have been performed. Time violations were discouraged by 100 cu fix and 50 cu/hour variable violation costs. So only opportunities for cost-saving delays with a relatively high real cost saving are taken. An example for such an opportunity is that an order may be loaded on an existing vehicle that already drives from the order's pickup to delivery location, but is some minutes late for delivery instead of driving it in an almost empty separate vehicle in time.

Figure 7 shows the results. As expected, the less strict constraints have to be obeyed the lower are the overall costs for transportation. On the other side, the number of constraint violations grows. Logistics companies can use this kind of simulation to estimate the possible cost savings and set up their cost structure accordingly.

4.5. Distributed Optimization

As described in section 3.3 the solution generation and optimization can be distributed across multiple agents and therefore also across multiple agent platforms. The above mentioned results where gained using a single optimization agent. In our current work, we are increasing the number of optimization agents and agent servers involved. Initial results are available running the optimization on 4 agents and 4 servers. The corresponding clusters were disjunct and no inter-cluster optimization was available in these initial runs. Also a smaller dataset with 1200 orders was used.

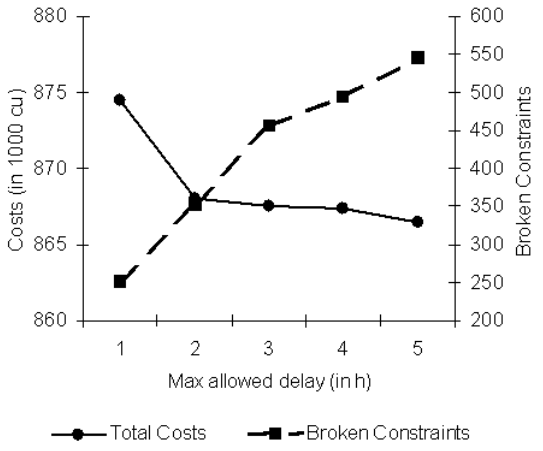

Figure 7. Influence of soft time windows to the optimization result and the number of constraints broken.

Optimization was 9 times faster than using a single agent. Roughly a factor three is achieved by the distribution on 4 servers, another factor three is due to the regional clustering itself. The loss in quality was 1.2%. This should be improved once transfers between clusters is available.

5. Future Work and Conclusion

In this paper we presented an agent-based approach to solve a real-world dynamic pickup and delivery problem with soft time windows. The constraints and parameterization have been chosen to get a realistic delivery plan that can be executed in daily business. The presented solution produced significantly better results compared to the results of professional dispatchers.

These results are currently achieved by a simple hill climbing optimization. Since in a productive environment the optimization is a constant process running the whole day there is usually computation time available during less busy hours of the day. Those less busy times could be used by more sophisticated optimization algorithms like tabu search to further explore the solution space [12].

Our current work focuses on the distribution of the optimization across multiple agents. The distributed optimization has been validated and already deployed for first experimental results. However, a number of improvements are still ongoing and mainly concern enabling transfers between clusters and simplifying the process of configuring distributed agent servers. Finally, we aim to also implement the fully distributed version (as mentioned in section 3.3) to estimate the possible performance and measure the real overhead in runtime and memory usage.

References

[1] T. G. Crainic. Long haul freight transportation. In R. W. Hall, editor, *Handbook of Transportation Science*. 2^{nd} Edition, Kluwer, 2002.

[2] Exel. Cost reduction still the biggest pressure on logistics for the industrial sector. In *http://www.exel.com/mediacentre/-newsreleases/releases.asp?int_articleID=710*, London, UK, 2004.

[3] K. "Fischer, J. P. Müller, and M. Pischel. Cooperative transportation scheduling: an application domain for DAI. *"Journal of Applied Artificial Intelligence"*, 10, 1995.

[4] M. Gendreau and J. Potvin. Dynamic vehicle routing and dispatching. In T. Crainic and G. Laporte, editors, *Fleet Management and Logistics*, pages 115–126. Kluwer, 1998.

[5] M. Hickman and K. Blume. A method for scheduling integrated transit service. In 8^{th} *International Conference on Computer-Aided Scheduling of Public Transport (CASPT)*, 2000.

[6] I. Ioachim, J. Desrosiers, Y. Dumas, M. Solomon, and D. Villeneuve. Clustering algorithm for door-to-door handicapped transportation. *Transportation Science*, 29(1):63–78, 1995.

[7] J.-J. Jaw, A. R. Odoni, H. N. Psaraftis, and N. H. Wilson. A heuristic algorithm for the multi-vehicle advance request dial-a-ride problem with time windows. *Transportation Research*, 20 B(3):243–257, 1986.

[8] R. Karp. Reducibility among combinatorial problems. In R. Miller and J. Thatcher, editors, *Complexity of Computer Computations*, pages 85–104. Plenum Press, New York, 1972.

[9] R. Kohout and K. Erol. In-time agent-based vehicle routing with a stochastic improvement heuristic. In *Proceedings of the 6th National Conference on Artificial Intelligence (AAAI-99); Proceedings of the 11th Conference on Innovative Applications of Artificial Intelligence*, pages 864–869, Menlo Park, Cal., July 18–22 1999. AAAI/MIT Press.

[10] G. Laporte and I. H. Osman. Routing problems: A bibliography. *Annals of Operations Research*, 61:227–262, 1995.

[11] S. Mitrovic-Minic. Pickup and delivery problem with time windows: A survey. Technical Report TR 1998-12, School of Computing Science, Simon Fraser University, Burnaby, BC, Canada, May 1998.

[12] W. P. Nanry and J. W. Barnes. Solving the pickup and delivery problem with time windows using reactive tabu search. *Transportation Research*, 34B:107–121, 2000.

[13] H. Psaraftis. Dynamic vehicle routing: status and prospects. *Annals of Operations Research*, 61:143–164, 1995.

[14] M. W. P. Savelsbergh and M. Sol. The general pickup and delivery problem. *Transportation Science*, 29(1):17–29, 1995.

[15] P. M. Thompson and H. N. Psaraftis. Cyclic transfer algorithm for multivehicle routing and scheduling problems. *Operations Research*, 41(5), 1993.

[16] H. Xu, Z.-L. Chen, S. Rajagopal, and S. Arunapuram. Solving a practical pickup and delivery problem. *Transportation Science*, 37(3):347–364, 2003.