Optimal Decision Making in Agent-Based Autonomous Groupage Traffic

Stefan Edelkamp and Max Gath
Center for Computing and Communication Technologies, Institute for Artificial Intelligence,
University of Bremen, Am Fallturm 1, 28359 Bremen, Germany
{*edelkamp,mgath*}*@tzi.de*

Keywords: Operations Research, Decision Making, Multiagent Simulation, Asymmetric TSP, Logistic Processes.

Abstract: The dynamics and complexity of planning and scheduling processes in groupage traffic require efficient, proactive, and reactive system behavior to improve the service quality while ensuring time and cost efficient transportation. We implemented a multiagent-system to emerge an adequate system behavior and focus on the decision making processes of agents that is based on the Traveling Salesman Problem (TSP) with aspects like contract time windows, individual restricted capacities of trucks, premium services and varying priorities of dynamically incoming orders. We present an optimal depth-first branch-and-bound asymmetric TSP solver with constraints on tour feasibility and depot reachability at any step of the process. To evaluate our approach, we use established benchmarks as well as its inclusion in a real-life multiagent-based simulation. Simulated scenarios are based on real customer orders of our industrial partner Hellmann Worldwide Logistics GmbH & Co. KG and are applied on real world infrastructures. The results reveal that efficient optimal decision making in multiagent systems increases the service quality and meets the requirements and challenges in logistics.

1 INTRODUCTION

In this paper we present an approach to optimize the planning and controlling processes in groupage traffic to improve the service quality while ensuring cost- and time-efficient transport processes. To meet the high requirements on complexity and dynamics, we implemented a multiagent system to ensure a flexible system behavior and developed efficient, optimal decision making algorithms for participating agents.

In transport logistics the decision making often relies on efficient solutions to the *Traveling Salesman Problem* (TSP), a touchstone for many general approaches in combinatorial optimization. In our application of a forwarding agency, the TSPs are generated via shortest paths reductions of route networks between pickup or delivery stops. Each order to be served corresponds to one city in the TSP. *Symmetric TSPs* (STSPs), for which edges cost are the same in both directions, are usually optimally solved with algorithms relying on the quality of the Held-Karp lower bound (Johnson et al., 1996).

Due to one-way streets in this paper we consider *Asymmetric TSPs* (ATSPs), where other bounds apply (Miller and Pekny, 1991; Karp and Steele, 1990). Most of them consider the solution of the according *Assignment Problem* (AP), which can be solved by the Hungarian algorithm or refined approaches (Jonker and Volgenant, 1986), followed by some tour patching strategies. ATSPs can be converted into STSPs, but this requires doubling the number of cities. Empirical TSP exploration results often partially refer to the 8th DIMACS Challenge. As an example, *depth-first branch-and-bound* (DFBnB) has been refined for the TSP (Zhang, 2000).

Many additional constraints apply in practice. Besides capacity and time constraints we have a certain mix of *pick-up and backhaul* and *premium services constraints* to be served in the tour, while others non-premium services are optional (but should be maximized). We also consider TSPs with *delivery and backhauls* (Gendreau et al., 1996); not to be mixed with *delivery and pick-up* (Anily and Mosheiov, 1994). For the TSP with release and due *time window constraints* (TSPTW), exact dynamic-programming algorithms exploit elimination tests to reduce the state space (Dumas et al., 1995). Variants can also be applied to TSPTW problems with precedence constraints. Frequently, branch-and-cut algorithms are the method of choice (Ascheuer et al., 2001). Introducing capacities constraints links to constraint solving. With the mix of premium and non-premium tasks we generate a preference problem that includes a combination of hard and soft constraints. Determining the optimal solution for the *vehicle routing problems* (Baldacci et al., 2012) with capacity,

time and premium services constraints, backhauling and dynamic change becomes practically intractable.

2 DISPATCHING IN GROUPAGE TRAFFIC

In *groupage traffic*, several orders to different destinations with less-than-truckload (LTL) shipments are served by the same truck to decrease total cost. In pickup tours, trucks transport loads from their origin to a local depot where the shipments are consolidated to build economical loads. Through LTL networks the load is transported to a depot in the destination area where each good is delivered to its final destination. The process planning complexity is even increased by individual qualities of shipments like weight, volume, priority, and value. Handling the complexity is aggregated by the high degree of dynamics that result also from unexpected events, such as an exact amount and properties of incoming shipments are not known in advance.

Quality of service is an important factor to succeed the economic objectives. The transportation of so-called premium services must be guaranteed with respect to their time windows while considering other hard constraints, e.g., the capacity of vehicles. In order to increase service quality through short transit times and reliable deliveries it is mandatory to handle the high degree of dynamics and complexity of logistic processes with adaptive, reactive system behavior.

By delegating the planning and controlling processes from central decision making systems to decentralized entities, e.g., agents that represent vehicles, the overall problem is split into smaller problem instances that can be solved optimally. Regarding the dispatching processes each vehicle has to find a tour with minimum costs, such that each pickup and delivery stop is visited exactly once and the tour returns to a central depot. This problem can be described by a TSP with stops i, $i \in \{1,2,\ldots,n\}$. All distances between two stops i and j given by $c_{i,j} \in \mathbb{R}$ with $c_{i,i} = 0$ for $1 \leq i,j \leq n$. Feasible solutions are permutations of $(1,2,\ldots,n)$ with the additional constraint that the first city to visit is the current position of the truck and the last city is the depot. Real transport infrastructures are commonly represented by directed graphs, so that we search an optimal tour for an ATSP.

In logistic transport networks participating forwarding agencies have to pay high amounts of penalty if they are not fulfilling the agreed commitments. Therefore, we distinguish hard premium service constrains that must be delivered on date of receipt and soft non-premium services constraints that can be delayed by up to 2 days.

Definition 1. *Pickup and delivery of* premium services *is mandatory and defined by*

$$p_i = \begin{cases} 1, & \textit{if } i \textit{ is a premium stop} \\ 0, & \textit{otherwise} \end{cases} \quad (1)$$

Hence, the priority of premium stops is higher than visiting other stops.

Definition 2. *The optimal tour of the asymmetric TSP must be feasible and fulfill the following requirements ordered by their priorities for the variables*

$$x_{i,j} = \begin{cases} 1, & \textit{if } (i,j) \textit{ is part of the tour} \\ 0, & \textit{else} \end{cases} \quad (2)$$

1. *Maximize the number of transported premium services:* $max \sum_{i=1}^{n} \sum_{j=1}^{n} p_i \cdot x_{i,j}$
2. *Maximize the number of visited stops:* $max \sum_{i=1}^{n} \sum_{j=1}^{n} x_{i,j}$
3. *Minimize the total cost of the path:* $min \sum_{j=1}^{n} \sum_{i=1}^{n} c_{i,j} \cdot x_{i,j}$ *subject to*
   (a) $\sum_{i=1}^{n} x_{i,j} = 1$ *for all* $j \in \{1,\ldots,n\}$
   (b) $\sum_{j=1}^{n} x_{i,j} = 1$ *for all* $i \in \{1,\ldots,n\}$
   (c) $x_{i,j} = \{0,1\}$ *for all* $j,i \in \{1,\ldots,n\}$
   (d) $\sum_{j \in S} \sum_{i \in S} x_{i,j} \leq |S| - 1$ *for all* $S \leq \{1,\ldots,n\}$

We assume that all premium stops have to be traversed such that we have to find the tour with minimum costs that includes all premium services and the maximum number of stops while satisfying all time and capacity constraints. Therefore, the problem changed into a maximizing-minimizing problem.

3 CONSTRAINT ATSP SOLVING

To apply *Branch-and-bound* (BnB), we extend depth-first search (DFS) with upper and lower bounds. To determine lower bounds for the ATSP, we transform it into the *Assignment Problem* (AP), that can be solved with the *Hungarian algorithm* (Jonker and Volgenant, 1986) in $O(n^3)$. While the AP is a relaxation of the asymmetric TSP, it can be used as heuristic function for the ATSP. In order to solve the AP, we extend the cost matrix with the distance from the depot to the current node (we want to return to the central depot which is not necessary the starting point). After solving the AP, we subtract it to compute the current lower bound. For increasing the upper bound (which is the sum of the weights in the subgraphs) we determine the minimum value to merge the subgraphs of the AP by comparing the respective columns and rows in the cost matrix and choosing the arcs with minimum costs.

An initial upper bound can be obtained by constructing any solution, e.g., established by a greedy approach. Unfortunately, for lager TSPs the branching process consumes a lot of time to determine a greedy solution. Therefore, we additionally computed the upper bound U at each node by applying Karp-Steel patching (Karp and Steele, 1990). As with standard DFS, the first solution obtained might not be optimal. With *depth-first BnB* (DFBnB), however, the solution quality improves over time together with the global value U until eventually the lower bound $L(u)$ at some node u is equal to U. In this case an optimal solution has been found, and the search terminates.

A *Constraint ATSP* is an ATSP in which additional state constraints are applied. As states are only discarded from the search that do not fulfill the constraints. This weakens but does not invalidate the lower bound. For example time and capacity constraints as well as priorities have to be satisfied.

For time constraints, the due date d_i for latest pickup (or delivery) at each stop $i \in \{1,\ldots,n\}$ has to be met. Each stop may require additional individual processing time δ_i (e.g., for loading), which can be compiled away (the solution value then changes by $\sum_{i=1}^{n} \alpha_i$). In some cases, release date r_i, $i \in \{1,\ldots,n\}$ are given, at which the order at stop i becomes issued. If the arrival is too early, we have to wait for time β_i.

For the TSPTW we choose the next time for traversing the edges to be minimized rather than the total time, which, nonetheless, is progressed for adaption to release and checking with due dates. The travel time between stops π_i and π_j on tour π is denoted by $t_{i,j}$, $i,j \in \{1,\ldots,n\}$. Additional *capacity constraints* for the vehicle yield a *Capacitated TSP* (CTSP).

Definition 3. *Let w_i be the freight weight at stop i and Δ_w be the max. weight of the vehicle. A tour π with stops $\{\pi_1,\ldots,\pi_n\}$ and depot d is* feasible *for the CTSPTW if for all $i = 1,2,\ldots,n$ we have $\sum_{l=1}^{i}(\alpha_i + \beta_i + t_{i-1,i}) \leq d_i$, $\sum_{l=1}^{i} w_i \leq \Delta_w$, and $d = \pi_n$.*

The pseudo-code is shown in Alg. 1. The procedure is invoked with the number of cities n the depot d, cost 0 and upper bound U set to some reasonable estimate (U can obtained using some heuristics; the lower it is, the better the pruning, but in case no upper bound is known, it is safe to set U to ∞). The tour is maintained globally and updated during backtracking. Another global variable *best* keeps track of the actual solution path. DFBnB sorts the set of successors according to increasing L-values is an optional refinement to the algorithm that often aids in accelerating the search for finding an early solution.

Release and due dates in TSPTWs induce a precedence relationship at each city. The relationship implies a partial ordering, so that a city can be selected as a successor of city only if it does not violate the imposed ordering. Given a bitvector of cities visited so far the subsumption check for precedence can be executed by native Boolean operations in $O(1)$. Similarly, premium services are checked on the word level.

```
DFBnB(n,u,g,U)
  tour_depth(u) ← u
  if (depth(u) = n − 1)
    if (g + c_u,d < U)
      best ← tour; U ← g + c_u,d
  else
    for each v_j in nextcities(u)
      if Constraint(v_j)
        if (g + h(v_j) < U)
          call DFSBnB(n, v_j, g + c_u,v_j, U)
```

Algorithm 1: DFBnB for Constraint TSPs.

Theorem 1. *Alg. 1 is optimal for admissible lower bounds, and the above pruning rules.*

Proof. If no pruning was taking place, every possible solution would be generated, so that the optimal solution would eventually be found. Sorting of children according to the L-values has no influence on the algorithm's completeness. The condition $L(v_j) < U$ confirms that the node's lower bound is smaller than the global upper bound. Otherwise, the search tree is pruned, as for admissible weight functions exploring the subtree cannot lead to better solutions than the one stored with U. All further pruning rules (like precedence or premium service pruning), cut off infeasible branches from the search tree so that the solution will be optimal for the TSP. Precedence pruning retains optimality for the TSPTW, while capacity pruning retains optimal for CTSP, etc. □

With h_a we define the heuristic that is derived from solving AP. The Hungarian algorithm for computing the solution as well as lower bound offsets based on the tour being Hamiltonian takes time $O(n^3)$. It can be incrementally be computed in $O(n^2)$, but bookkeeping becomes involved. With h_c we define the heuristic that is defined as the sum of the column minima in the distance matrix. If cities are visited, column minima are subtracted from the sum. The travel distance back to the depot is added on top. We compute h_c incrementally in $O(1)$ time and space. Let u be the successor of v, and m_i be the precomputed minima of columns i, $i = \{1,\ldots,n\}$ in distance matrix D. Moreover, let the heuristic value of the depot d be defined as $h_c(d) = \sum_{i=1}^{n} m_{d,i}$. For each expanded node u we compute $h' = h(u) - m_d$ if $u = d$, and $h' = h(u) - c_{u,d}$, otherwise. For each generated successor v of u we have $h_c(v) = h' - m_v + c_{v,d}$.

An advantage of the tree structure is that the constraints are checked in constant time and space. The

current time, capacity and premium service information are stored at each node. Considering the time efficiency of the algorithm, bit-vectors are realized by one computer word: we have bit-vectors for the visited cities, the relative ordering among them imposed by the time window, and the priority service constraints. All bit-vector operations (setting, clearing of bits, check for subsumption) run in $O(1)$.

Theorem 2. *Let w be the word width of the computer. The incremental solver for the asymmetric TSP with time windows, capacity, and premium service constraints and heuristic h_c runs in $O(n/w)$ per node. It allocates $O((n/w) \cdot n^2)$ space in its initialization phase and no more memory during the search.*

Proof. The space consumed for the distance matrices as well as their compression and copies is $O(n^2)$. The matrices for successor reordering and filtering take $O(n^2)$ and each state on the stack requires $O(n/w)$ space. As the stack is limited by the depth times the number of successor, its memory needs are also bounded by $O((n/w) \cdot n^2)$. All other structures (tour covers, interval sizes, row and column covers and minima, partial ordering bitvectors, successor sets, and visited flags) take at most $O(n)$ space. □

To determine the optimal solution for a TSP with premium services we have to extend the algorithm. All premium stops must be traversed so that we have to find the tour with min. costs that includes all premium services and the max. number of stops while satisfying all time and capacity constraints. Therefore, the problem changed into a max.-min. problem and branching is not applicable by comparing the costs to the lower bound L. We have to find a feasible solution that includes all premium services within max. search tree depth and the shortest distance within that depth.

To speed up search for a first solution in depth $n-1$, the computation of lower bounds is disabled and a subtree is cut if the current tour is not a feasible tour. The following rule prunes the tree using bitvectors and Boolean operations. Before starting the search, the premium services are sorted by their capacities. The number of premium services included in the best solution is saved and updated if a better solution is found. Next, we compute the difference Δ between the number of currently included premium services and the amount of premium services in the best-known solution. Afterwards, the weight of Δ lightest not considered premium services is accumulated and we check if they exceed the maximum capacity of the truck. The solution remains optimal and complete because a feasible solution with more premium services exists. The pruning rule is applied to regular orders accordingly if all premium services are included within the tour. The application of this pruning rule is optional, because it is done in $O(\Delta)$ time (for summing up the weight of not considered orders). Nevertheless, it speeds up the algorithm if good solutions are known in advance, the maximum capacity of the truck is reached, and no more orders can be operated. In this case big subtrees are pruned early. If we find a first feasible complete tour objectives 1. and 2. in Def. 2 are fulfilled. The remaining objective is to reduce the total cost and the problem has changed to a classical TSP with constraints. As a result, the computation of lower and upper bounds described above is activated. In this case, less nodes are expanded because the searching process is goal-directed. If heuristic h_a instead of h_c is activated, the efforts at each node are higher.

4 AGENT-BASED DISPATCHING AND SIMULATION

To cope with the dynamics and complexity of logistic processes, we are implementing autonomous groupage traffic with agent technologies. The agent system induces a proactive, reactive, and adaptive system behavior. Agent-based commercial systems are used within the planning and controlling processes of containerized freight (Dorer and Calisti, 2005). Team formation and interaction protocols have been designed for efficient resource allocation (Schuldt, 2011). Agent-based systems have optimized planning and controlling processes within dynamic environments (Fischer et al., 1995). Other ranges of application have been provided for industrial logistic processes (Skobelev, 2011).

In our setting agents represent trucks and orders. Whenever a new transport request has to be aced upon, a new agent is created that represents an order. The goal of the agent is to find a proper transport service provider to pickup or deliver the shipments with respect to the time windows and premium service constraints. The agent starts the contract-net protocol for negotiating with available transport service providers. All operating trucks are represented by an agent as well. The truck agents evaluate the proposals by determining its additional costs for transporting. This is done by solving the TSP optimally that is specified in Def. 2 while considering time and capacity constraints with the algorithm described in Sec. 3. In order to schedule new orders also while transporting other shipments, the truck has to consider its current capacity constraints and position. For example, picked up shipments reduce the capacities and the position of the vehicle affects the determination of short-

est ways and tours. Consequently, we link the planning and decision making processes of the agents directly with their execution behaviors and consider all relevant observed changes of the environment as well as the internal state of the agent within the decision making and tour planning. On the other hand, new plans can effect the executed actions of the agents. Therefore, the truck agent checks during driving, if the next stop has changed and if necessary he adapts the tour. In real processes as well as in the simulation the handing processes (boarding and deboarding of shipments) must not be interrupted. This requirement is satisfied by not adopting plans that manipulates the running handling processes.

To transport a premium service instead of conventional orders or another premium service by driving a shorter distance, already accepted orders may not be included in the new plan. If these orders have not been boarded the truck agent sends a message to the agent that acts on behalf of the corresponding order. Afterwards, the order agent negotiates with other transport service providers again. Potentially, this results in a series of computation and communication intensive negotiations between agents to achieve small improvements. To weaken this effect (especially if several shipments are processed consecutively within a short time window and the global allocation changes significantly) the agent waits for a certain period of time before it starts the negotiation procedure. For optimal decision making agents have to solve a TSP for each proposal. Consequently, numerous TSPs have to be solved in the planning and controlling processes.

Applying simulation for evaluating multiagent systems before their deployment in real applications is an accurate cost and time reducing method. Our system PlaSMA (`http://plasma.informatik.uni-bremen.de`) extends the JADE framework (Bellifemine et al., 2007) and provides a discrete time simulation that ensures a correct conservative synchronization with time model adequacy, causality, and reproducibility. The transport infrastructure within the simulation environment is modeled as a directed graph. Nodes represent traffic junctions or logistic sources and sinks, while edges represent different types of ways, e.g., roads, motorways, trails, and waterways. To model sound planning and controlling processes in the logistic domain, we extended our system to import infrastructures from OpenStreetMap (OSM) databases. The directed graph includes information about the real worlds speed limits, the distance as well as the type of an edge. Particularly within large infrastructures determining the shortest path between nodes is an essential, costly, and time-consuming procedure within the decision making process of an agent. However, computing the distance matrix between cities is essential for solving the TSP on a shortest path reduced graph. Consequently, we implemented single-source shortest paths search (Dijkstra, 1959) with a memory-efficient joint representation of graph and heap nodes. Computing a distance matrix requires many shortest-path queries with a fixed starting node and is time-critical. Hence, we adapted the search procedure and saved the last visited nodes within a hash map as well as in the heap as long as the start node has not changed. While processing new search requests we check in constant time, if the shortest path to the node was already found and retrieve the corresponding node from the heap. Otherwise, shortest path search is continuing at the last expanded node. We extended the search with a cache.

5 EVALUATION

In the first setting (all experiments were run on an Intel i7 processor with negligible RAM requirements; see Theorem 2) we generated a fully-connected random graph of k nodes with edge weights in $[0..C]$. With $C = 10$ our solver can handle problems with 500 cites. For $C = 100$, and up to 100 cities the experimental outcome shows that there are rare unfortunate cases that need hours for computation, but generally remain tractable. For smaller values of n (the number of cities) and more complex TSPs, our incremental solver with constant time per node was often more effective than computing a lower bound by solving the Assignment Problem (Jonker and Volgenant, 1986) at each node.

Next, we extended the solver with time windows and focus on the performance of applied heuristics. As release and due times are more difficult to generate randomly, we took an existing benchmark set of TSPTW problems (Dumas et al., 1995). The results in Table 1 examine the search efforts for DFBnB with h_a and h_c (see Sec. 3). While the number of nodes is substantially lower for h_a, the solving time is sometimes (but not always) larger than for h_c, indicating that the more constraint the problem is, the worse the AP approximation and the better the search with a simpler heuristic. Note that the best results were obtained with a combination of both heuristics, using the more expressive one in the top part of the tree and the less expressive one in the bottom part of the tree.

To investigate the interactions between agents we implemented several scenarios within PlaSMA. The transport infrastructure contains 124,462 nodes and 292,521 edges. Orders and transport requests were

Table 1: Results in Dumas' TSPTW Benchmark (Exp. nodes and CPU time for the two heuristics are shown).

Prob.	Cost	E_a	T_a	E_c	T_c
n20w20.001	378	49	$< 1s$	2784766	$< 1s$
n20w20.002	286	97	$< 1s$	3234936	$< 1s$
n20w20.003	394	138	$< 1s$	4944477	$< 1s$
n20w20.004	396	156	$< 1s$	2331312	$< 1s$
n20w20.005	352	41	$< 1s$	4017260	$< 1s$
n20w40.001	254	38022	3s	103087541	18s
n20w40.002	333	88	$< 1s$	11388523	2s
n20w40.003	317	1409	$< 1s$	21158796	3s
n20w40.004	388	7676	1s	35117607	6s
n20w40.005	288	10287	2s	20801644	3s
n20w60.001	335	40810	14s	223904879	43s
n20w60.002	244	97144	7s	81367918	15s
n20w60.003	352	399127	27s	31292739	5s
n20w60.004	280	4055453	258s	1245195466	238s
n20w60.005	338	105393	10s	104049862	18s
n20w80.001	329	316992	35s	288653549	56s
n20w80.002	338	260552	36s	166880630	33s
n20w80.003	320	15959	3s	208526467	42s
n20w80.004	304	1258898	80s	373077547	73s
n20w80.005	264	5224435	438s	1660621704	332s
n20w100.001	237	1635101	52s	1232596799	279s
n20w100.002	222	68954	7s	2203174867	531s
n20w100.003	310	13382035	765s	2586795810	538s
n20w100.004	349	34289	2s	1213551958	266s
n20w100.005	258	688887	44s	2308713055	548s

provided by the Bremen office of Hellmann Worldwide Logistics GmbH & Co. KG. We started a reverse geocoding process to map the address information to coordinates and determined the nearest neighbor node in the map, to link the addresses with graph nodes. The real weight, premium service constraints, latest delivery times as well as the incoming dates are attached with the order. Since exact delivery times are not available, only the date is considered during evaluation. As a result, we modeled the dynamics by generating new orders successively until all orders of this day have been dispatched. In real transport processes, vehicles with interchangeable units are sent to stops where numerous shipments have to be handled. Consequently, we did not consider orders if more than six orders had to be picked up at the same stop. While the dynamics of the planning and controlling processes of delivery tours are limited due to the fact that shipments have to be loaded at the depot in the morning and cannot be changed anymore, we only consider pickup orders and simulate the planning processes simultaneously to the execution of plans.

We assume that tours start at 7am at the depot and the vehicle must return not after 4pm. Moreover, the average velocity of a truck is set to 60 km/h, all trucks have a maximum possible capacity of 7.5 tons and a truck starts only one tour per day. The handling and waiting periods at incoming goods departments is half an hour.

Table 2: CPU time in ms for solving the TSP as well as for shortest path searches during the matrix computations.

#Trucks	#TSP	time for solving TSP in ms	time for matrix computation in ms
5	6,122	199,043	4,912,094
10	11,893	324,566	16,588,955
20	22,457	604,675	56,845,930
25	26,751	972,579	75,882,546
30	31,542	2,246,511	94,921,678
40	40,759	7,979,063	144,742,544
60	56,152	79,704,147	225,749,232

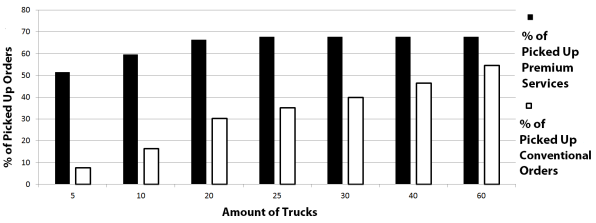

Figure 1: Black/White bars show the percentage of picked up premium/conventional services in the selected scenario.

We modeled seven scenarios. In each scenario 1,265 orders are distributed within a whole week while the number of trucks is varying. About six percent of the total amount of modeled orders are premium services. Table 2 shows the computation times elapsed during each simulation run for computing the distance matrix with the shortest-path algorithm as well as the time for solving the TSP specified in Def. 2 while cons. The results reveal that more computation time is required for the reduction of the real world infrastructure graph to a minimum distance matrix between relevant nodes than for optimally solving of the TSP. Consequently, the shortest path matrix computation is the most expensive part of the decision making process of the agent, if we are considering the computational effort. Moreover, Table 2 indicates that the amount of TSPs rises with the number of available trucks. This is obvious, because the TSP solver is an essential part within the decision making process of each truck.

Fig. 1 shows the percentage of picked-up premium and conventional orders within each scenario. Consequently, the agent system serves premium services with higher priorities. Even with 5 trucks nearly 50% of all premium services are sill picked up, while processing more than 90% of conventional orders are postponed to other days. An increase of the number of available trucks leads to processing more premium services as well as more conventional orders. If about 70% of all premium services were satisfied, only transported conventional orders increased. However, not more than 70% of all premium services are transported even if enough trucks are available. We

assume, this is explained by the scenario assumptions in combination with the customer orders. For example, if the max. velocity of trucks is set to 60 km/h, it is not possible to pick up premium services with a distance of more than 60 km at 8am, if the trucks start the transporting process at 7am.

6 CONCLUSION AND OUTLOOK

We developed a dispatching system matching the requirements of forwarding agencies in groupage traffic. To face the dynamics of consecutively incoming orders and the high complexity of logistic processes, we implemented a reactive and proactive multiagent system. The agents link the planning and scheduling processes directly with their actions. Therefore, changes of the environment can be considered during runtime and induce a reactive behavior. We focused on the planning and decision making processes of the agents and developed an efficient TSP solver that is crucial for negotiation with service customers agents. The optimal branch-and-bound TSP solver is time and space efficient: it checks resource, time, and premium service constraints in $O(1)$ time and space per generated node. Moreover, after the allocation of $O((n/w) \cdot n^2)$ words at initialization time for the stack contents and other structures (including copies of the distance matrix) no additional memory is allocated during the search. The performance was proven on artificial graphs with test sets from benchmarks (Dumas et al., 1995) as well as in simulated real world scenarios of an entire week by computing more than $56,000$ TSPs including time windows, capacities, handling times, and priorities.

In further investigations, we will evaluate the multiagent system on multiple computer and enable parallel decision making. As a result, trucks have sufficient computational power to continue negotiating with other trucks and improve the allocations consecutively. Applying the contract-net protocol in combination with the optimal TSP solver, the negotiations will converge in a global optimum (Shoham and Leyton-Brown, 2009, p. 27).

ACKNOWLEDGMENTS

The presented research was partially funded by the German Research Foundation (DFG) within the Collaborative Research Centre 637 "Autonomous Cooperating Logistic Processes: A Paradigm Shift and its Limitations" (SFB 637) at the University of Bremen, Germany. We thank the Bremen office of Hellmann Worldwide Logistics & Co. KG for great cooperation.

REFERENCES

Anily, S. and Mosheiov, G. (1994). The traveling salesman problem with delivery and backhauls. *Operations Research Letters*, 16(1):11–18.

Ascheuer, N., Fischetti, M., and Groetschel, M. (2001). Solving the asymmetric travelling salesman problem with time windows by branch-and-cut. *Math. Programming*, 90:475–506.

Baldacci, R., Mingozzi, A., and Roberti, R. (2012). Recent exact algorithms for solving the vehicle routing problem under capacity and time window constraints. *European Journal of Op. Res.*, 218(1):1–6.

Bellifemine, F., Caire, G., and Greenwood, D. (2007). *Developing Multi-Agent Systems with JADE*. John Wiley & Sons.

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

Dorer, K. and Calisti, M. (2005). An Adaptive Solution to Dynamic Transport Optimization. In *AAMAS*, pages 45–51.

Dumas, Y., Desrosiers, J., Gelinas, E., and Solomon, M. (1995). An optimal algorithm for the travelling salesman problem with time windows. *Operations Research*, 43(2):367–371.

Fischer, K., Mueller, J. P., and Pischel, M. (1995). Cooperative Transportation Scheduling: An Application Domain for DAI. *Journal of Applied Artificial Intelligence*, 10:1–33.

Gendreau, M., Hertz, A., and Laporte, G. (1996). The traveling salesman problem with backhauls. *Comp. & Op. Research*, 23(5):501–508.

Johnson, D. S., McGeoch, L. A., and Rothberg, E. E. (1996). Asymptotic experimental analysis for the Held-Karp traveling salesman bound. In *SODA*, pages 341–350.

Jonker, R. and Volgenant, A. (1986). Improving the hungarian assignment algorithm. *Operations Research Letters*, 5:171–175.

Karp, R. M. and Steele, J. M. (1990). Probabilistic analysis of heuristics. In *The Traveling Salesman Problem, E. Lawler et al. (eds.)*, 181–205.

Miller, D. L. and Pekny, J. F. (1991). Exact solution of large asymmetric traveling salesman problems. *Science*, 251:754–761.

Schuldt, A. (2011). *Multiagent Coordination Enabling Autonomous Logistics*. Springer Verlag.

Shoham, Y. and Leyton-Brown, K. (2009). *Multiagent Systems: Algorithmic, Game-Theoretic, and Logical Foundations*. Cambridge Univ Press

Skobelev, P. (2011). Multi-agent systems for real time resource allocation, scheduling, optimization and controlling: Industrial applications. In *HOLOMAS*, 1–14.

Zhang, W. (2000). Depth-first branch-and-bound versus local search: A case study. In *AAAI*, p.930–936.