Can agents measure up? A comparative study of an agent-based and on-line optimization approach for a drayage problem with uncertainty

Tamás Máhr*†[1], Jordan Srour[2], Mathijs de Weerdt[3], and Rob Zuidwijk[4]

[1,3]Delft University of Technology, PO Box 5031, 2600 GA Delft, The Netherlands, {T.Mahr, M.M.deWeerdt}@tudelft.nl
[2,4]Rotterdam School of Management, Erasmus University, Burg. Oudlaan 50, 3062 PA Rotterdam, The Netherlands, {JSrour, RZuidwijk}@rsm.nl

May 7, 2009

Abstract

Experiments studying the behavior of agent based methods over varying levels of uncertainty in comparison to traditional optimization methods are generally absent from the literature. In this paper we apply two structurally distinct solution approaches, an on-line optimization and an agent based approach, to a drayage problem with time windows under two types of uncertainty. Both solution approaches are able to respond to dynamic events. The on-line optimization approach utilizes a mixed integer program to obtain a feasible route at 30-second intervals. The second solution approach deploys agents that engage in auctions to satisfy their own objectives based on the information they perceive and maintain locally. Our results reveal that the agent-based system can out perform the on-line optimization when service time duration is highly uncertain. The on-line optimization approach, on the other hand, performs competitively with the agent-based system under conditions of job arrival uncertainty. When both moderate service time and job arrival uncertainties are combined, the agent system outperforms the on-line optimization; however, in the case of extremely high combined uncertainty, the on-line optimization outperforms the agent-based approach.

Keywords

dynamic vehicle routing, pick-up and delivery problem with time windows, on-line optimization, multi-agent system

1 Introduction

Total maritime container traffic grew to approximately 417 million twenty foot equivalent units (TEUs) in 2006 (BTS, 2007). This phenomenal growth of containerized freight, since the container's introduction in 1956, has led to similar growth in the drayage industry. Drayage commonly refers to the transport of containerized cargo to and from port or rail terminals and inland locations. Unfortunately, the drayage portion of a door-to-door container move tends to be the most costly part of the move. Morlok and Spasovic (1994) indicate that up to 40% of the cost for a 900 mile container move can be attributed

*Corresponding author. Phone: +36 30 7252799
†Authors are listed in alphabetical order.

to the 50 mile drayage portion of the move. There are a variety of reasons for this disproportionate assignment of costs, including a great deal of uncertainty at the interface of modes. For example, trucks moving containers to and from a port terminal are often uncertain about the time a designated container will become available due to disruptions in the physical and administrative processes involving, among others, the shipping line, stevedore, and customs. In order to generate a profitable route for multiple containers, in one day, the drayage industry must adopt planning systems that can anticipate such uncertainty. This paper compares two structurally different planning approaches for drayage operations in an uncertain environment.

Agent-based solutions have been shown to perform well in uncertain domains (Fischer et al., 1995). That is, in domains where the problem is continually evolving. The key objective of this paper is to study the performance of an agent-based solution and an on-line optimization approach with respect to handling uncertainty in the context of a drayage company at the Port of Rotterdam, the Netherlands. In our case, a container transport company with a fleet of 40 vehicles must pick up containers from terminals at the Port of Rotterdam, transport the containers to a customer location in the hinterland arriving within a given time window, wait with the container until it is loaded or unloaded, and then return the full or empty container to another port terminal.

Given this method of operations, the problem may be described, in a static manner, as a pick-up and delivery problem with time windows (PDPTW). Reality, however, reminds us that this problem is anything but static and as such we study this problem in an on-line context taking into account two types of uncertainty – service time uncertainty and job arrival uncertainty. Job arrival uncertainty is the most basic type of uncertainty, as a job cannot be planned for or served until it is made known to the planning system. Furthermore, while the presence of a job may be known in advance, the amount of time required to pick it up from the terminal, service it at the customer location, and process it at the return terminal can be highly variable.

The remainder of this document describes the foundational literature on the PDPTW under conditions of uncertainty as solved by both optimization-based and agent-based solution approaches (Section 2). In Section 3, we provide a detailed description of our experimental design including a description of the solution approaches, the data used, and the two types of uncertainty examined. Results, on the performance of both systems across several scenarios of varying service time and job arrival uncertainty, appear in Section 4. A discussion of these results and suggestions for future research conclude this article.

2 Related Work

As noted in the introduction, at the heart of this research is a case study in drayage. This case can be described in operations research terms as a truckload pick-up and delivery problem with time-windows (PDPTW). In the PDPTW, a fleet of vehicles, capable of carrying only one job at a time, must pick up a job from one location and drop it off at another location within a specified time period. Finding the assignment of jobs to trucks that minimizes costs (in the form of total distance, empty distance, or operating costs) is the solution goal. Recently, the literature has seen a burst of survey papers on the topic of pick-up and delivery problems. These papers collectively detail the basic problem while classifying the multiple variants (Berbeglia et al., 2007; Parragh et al., 2008a,b; Gribkovskaia and Laporte, 2008; Cordeau et al., 2008). Using the classification scheme of Berbeglia et al. (2007) our problem may be more specifically described as a one-to-one pick-up and delivery problem with time windows and multiple unit-load vehicles.

While it is easiest to describe and classify these problems in a static manner, these problems may in reality be studied from either a static or dynamic perspective (Ghiani et al., 2003). Static vehicle routing problems (VRPs) assume that all relevant problem information or input data is known ahead of time and may be exploited in the solution process. On the other hand, the input data for dynamic (also known as on-line (Jaillet and Wagner, 2006) or real-time (Yang et al., 1999)) VRPs is revealed over time and solution mechanisms are designed to react and evolve accordingly. In this regard there is little time to plan for or optimize decisions – there is only time to react. Given this reactive nature of on-line solution mechanisms, one could argue that all on-line algorithms are in effect heuristics – solution mechanisms

designed to yield a good feasible solution rapidly.

Dynamism, also referred to as uncertainty, can have different sources. The type of uncertainty classically studied in vehicle routing is new job arrival. Other types of uncertainty, often studied in the related field of scheduling, include variable activity durations or variable resource failures. A recent review of dynamism in the field of scheduling was published by Herroelen and Leus (2005). For our study, we focus on two types of uncertainty – variable service times and job arrivals revealed over time.

To summarize the position of our work in the literature, we examine two structurally distinct solution approaches–an optimization-based solution approach and an agent-based solution approach – to the dynamic truckload pick-up and delivery problem with time windows under two types of uncertainty. The structurally differentiating feature of the two solution approaches is the level of control – centralized versus decentralized. Specifically, an optimization-based solution approach, focusing on a single objective and using full system information, exemplifies centralized control. In comparison, agents, optimizing their own unique objectives using information they perceive and maintain locally, exemplify decentralized control. The following two subsections describe in greater detail both optimization-based and agent-based approaches to the Dynamic Vehicle Routing Problem (DVRP).

2.1 Optimization-based Approaches for Vehicle Routing

When studying centrally controlled optimization-based approaches for vehicle routing, the role of dispatcher serves as a natural metaphor. A dispatcher is responsible for using all known information (such as job pick-up and drop-off locations, job size, job time windows, fleet size, vehicle capacity, and vehicle locations, etc.) to develop a feasible routing of vehicles to service all jobs at the least cost. In practice, the one characteristic the dispatcher does not posses is clairvoyance. The dispatcher is unaware of most future demands or service times until they occur. Thus, previously optimized plans must be updated to accommodate new demands as they arrive to the system or longer/shorter service times as they are realized; possibly moving the previous plan to a point far from optimal.

Solving dynamic vehicle routing problems from a centralized perspective has been an active area in the literature for over 20 years. Psaraftis' 1988 survey of results in this problem domain is valuable as it clearly lists the primary differences between the static and dynamic versions of the VRP. Chief among these differences is the essential nature of time coupled with the need for information update mechanisms and fast computation times for solutions.

In general, solution approaches to the dynamic vehicle routing problem tend to be premised on innovative manipulations of the static version of the vehicle routing problem. Both static and dynamic problem types may depend on either stochastic or deterministic input data (Powell et al., 1995). Stochastic approaches are designed to exploit statistical information gleaned from data on past problem instances. Alternately, deterministic approaches use only known information to solve the routing problem at specific decision instances. In this study, neither the agent-based nor the optimization-based approach has access to stochastic or forecasted data. We therefore focus our literature review on deterministic approaches.

The decision instances used in deterministic approaches may be designated as the moment of a single job arrival or when a pre-set number of jobs arrive or a specified amount of time has passed. In this way no unknown information is assumed at each decision epoch. The method by which the problem is solved at each decision epoch is the main differentiator between the methods described in the past ten years.

Regan et al. (1995, 1996), for example, propose a set of rule-based heuristics for load acceptance and assignment decisions. These heuristics are sometimes termed *incremental approaches* as they incrementally change the problem at each decision instance without fully resolving the problem. The use of routing heuristics in an on-line setting stems from their history and success in an off line context. Both Cordeau et al. (2002) and Laporte et al. (2000) provide comprehensive surveys on routing heuristics. Two primary branches of classical heuristic approaches are the *constructive methods* and the *improvement methods*. From the first group, a classical example is the *insertion heuristic*, a polynomial-time heuristic without performance guarantees (Solomon, 1987). This algorithm is widely used to create an initial solution that is further optimized by improvement methods.

In the realm of improvement methods, Thompson and Psaraftis (1993) introduce cyclic transfers of jobs among vehicles as one way to improve an initial solution. In their method, a *b-cyclic k-transfer* shifts

k jobs from each vehicle to the next one in a circular permutation of b vehicles. This is a very general framework that can express many different and complicated exchanges of jobs between vehicles. As a result, the space of cyclic transfers is too big to consider a full search. Subsequently, Breedam (1994) and Kinderwater and Savelsbergh (1997) introduce simpler moves such as: load reallocation (moving loads from one vehicle to another), load exchange (exchanging loads between vehicles), and crossover (mixing the routes of two vehicles).

In contrast to these heuristic approaches, Yang et al. (2004) demonstrate the superiority of an exact mixed integer programming formulation of the PDPTW, solved in a rolling horizon framework at each decision instance. They compare their reoptimization approaches to three heuristic approaches (a simple round robin assignment, an insertion heuristic, and a reordering approach). This comparison reveals that the reoptimization approaches systematically outperform the heuristic approaches by about 10%. This superior re-optimization approach, has its origins in the paper by Yang et al. (1999). Mahmassani et al. (2000) further this line of work by examining a hybrid rule-based heuristic and optimization approach. Most recently, Chen and Xu (2006) investigate a dynamic column generation technique as a means to handle a set-partitioning-type formulation at each decision epoch.

While re-optimization or plan/re-plan approaches are appealing as they tend to closely mirror manual operations (i.e. the situation where a dispatcher plans and re-plans routes as new jobs arrive), there are also drawbacks. Powell et al. (2000) highlight the myopic nature of these approaches. For example, optimal solutions implemented early in the day may no longer be optimal in light of additional information that arrives later in the day. Furthermore, the term "re-optimization" may be misleading as the solution to the exact mathematical programming formulation may only be feasible (rather than optimal) at each decision instance. This phenomenon can be exacerbated when the problem size is large as plan/re-plan approaches tend to suffer from the burden of rapidly responding to new information, especially in cases where an optimization-based algorithm is used.

In order to address these issues of large problem size, the idea of breaking the problem into component parts is appealing. Ghiani et al. (2003) provide a review of real-time vehicle routing strategies with a particular emphasis on parallel computing strategies. Their review focuses on different control and communication structures (e.g. master-slave, etc.) for efficiently searching the solution space arising from a centralized problem formulation. While they present a useful method for classifying parallel computing strategies for the VRP, they overlook computing possibilities that begin with a decentralized problem formulation. For example, it may be beneficial to reframe the vehicle routing problem based on roles within the problem, i.e. jobs and vehicles. Dividing the problem in this manner renders the solution approach into a fully decentralized approach. The following section describes decentralized agent based approaches in the context of vehicle routing.

2.2 Agent-based Approaches for Vehicle Routing

Modeling a vehicle routing problem as a decentralized system is primarily motivated by the decentralized nature of the environment in which these problems occur. When a company needs to organize transportation for a set of jobs, it has to deal with the customers the jobs came from, the drivers who will execute the actual transportation, the legal and social environment of the company, and last but not least the company's need to make a profit. If one does not want to abstract away from this setting then a decentralized model considering all the players forms a natural metaphor.

Such a model is provided by multi-agent systems (MAS) (Wooldridge, 2002). Multi-agent systems consist of a group of autonomous decision makers (artificial agents) that are capable of interacting with each other,wile pursuing a goal. If the agents share a common goal, they can cooperate to achieve it. When the goals are contradicting, the agents are competitive, they may hide sensitive information from each other, and try to achieve their goals individually. Applications of agent systems span from vehicle routing and logistics in general (Dorer and Calisti, 2005), through business process management (Hutzschenreuter et al., 2008), procurement and contracting (Jakob et al., 2008), aerospace applications (Scerri et al., 2008), energy management (Das et al., 2008), to security applications (Rehak et al., 2008).

In vehicle routing, a simple agent model may contain job agents and vehicle agents pursuing an assignment by minimizing some given objective. More complex models may include agents for the company,

the planners, or for customers having more than one job. Interactions between the agents constitute a major part of the solution mechanism.

In their frequently cited paper, Fischer et al. (1995) argue that such multi-agent models fit the transportation domain particularly well. Their main reasons are (similar to those mentioned above) that (i) the domain is inherently decentralized (trucks, customers, companies etc.); (ii) a decentralized MAS architecture can cope with multiple dynamic events; (iii) commercial companies may be reluctant to provide proprietary data needed for global optimization while agents can use local information; and (iv) inter-company cooperation can be more easily facilitated by agents.

To illustrate their idea, the authors provide a detailed MAS architecture for transportation problems that evolve over time thereby exhibiting job arrival uncertainty. This architecture makes a distinction between a higher and a lower architectural level. At the higher level, company agents negotiate transportation requests to eliminate ill-fitting jobs. On the lower level, truck agents (clustered per company) participate in simulated market places, where they bid on offered transportation jobs. Truck agents use simple insertion heuristics to calculate their costs and use those costs to bid in auctions. Although the heuristics used by the truck agents to calculate bids are rather crude, Fischer et al.'s research (1995) suggest that in dynamic problems (problems with high uncertainty), such methods survive better than sophisticated optimization methods.

Their bi-level approach recognizes that one shortcoming of a fully decentralized system is that agents only have access to local information. The need to find a balance between the omniscience of a centralized model and the agility of a decentralized model, was similarly recognized by Mes et al. (2007). They also introduce a higher level of agents, but with a different role than the high-level agents of Fischer et al. (1995). Mes et al.'s two high-level agents (the planner and the customer agent) gather information from and provide information to agents assigned beneath them. The role of the higher level agents is to centralize information essential for the lower level agents to make the right decisions.

Some researchers have gone even further in proposing centralization in agent-based models. These researchers concentrate problem information in some agents for the purpose of making better decentralized decisions. In one of the few models that has been applied in a commercial company, Dorer and Calisti (2005) cluster trucks geographically, using one agent per cluster. This way, one agent plans for multiple trucks. They use insertion heuristics to initially assign jobs to trucks, and then use cyclic transfers (Thompson and Psaraftis, 1993) to enhance the solution. In a similar but slightly different approach, Leong and Liu (2006) introduce a planner agent that has complete information about the other agents. In their method, planner agents coordinate the improvement procedures performed by the job and vehicle agents considering global objectives, such as the minimization of the number of vehicles used, and the total traveled distance. The authors analyze the performance of their model on a selection of Solomon benchmark sets, and show that it performs competitively on those sets.

As noted previously, however, the move towards centralization can hinder the ability of the agents to react quickly on local information. Given the uncertain environment of our problem, we are interested in the competitiveness of a system with fully decentralized agents. One example of a fully decentralized agent approach in the transportation domain is that of Bürckert et al. (2000). They propose a more detailed (holonic) agent model. In this model, they distinguish between truck, driver, chassis, and container agents that have to form groups (called holons) to serve jobs. Already formed holons use the same techniques to allocate tasks as Fischer et al., but the higher agent level is omitted, since they model only a single company case. The main focus of their research is computer-human cooperative planning, which makes their contribution interesting in spite of the fact that a thorough comparison of their approach to other ways of dealing with this problem is missing.

In most of the approaches mentioned above, agents use simple heuristic techniques to make decisions. In the related domain of production planning, Persson et al. (2005) embed optimization in the agents to improve local decisions. They show that optimizing agents outperform the heuristic agents, but they also show that central optimization still outperforms the optimizing, yet decentralized, agents.

While Persson et al. concentrated on making optimal decisions within agents, there is still a need to combine these individual solutions in an optimal way. For example, in the transport problem context, when jobs are assigned to trucks sequentially, at every assignment the truck with the cheapest insertion

gets the job. Later, however, it might turn out that it would have been cheaper to assign the same job, together with newly arrived jobs, to another truck. From the truck perspective this means that trucks that bid early and win assignments might not be able to bid later on more beneficial (better fitting) jobs. This problem is called *the eager bidder problem* (Schillo et al., 2002) and several researchers propose alternative techniques to deal with this problem. Kohout and Erol (1999) introduce an enhancement process between pairs of agents. The process mimics the well-known 'swapping' or two-exchange improvement techniques (Cordeau et al., 2001). Kohout and Erol implement this swapping process in a fully decentralized way, and show that it yields significant improvement.

Perugini et al. (2003) extend Fischer et al.'s contract-net protocol to allow trucks to place multiple possibly-conflicting bids for partial routes. These bids are not binding–trucks are requested to commit to them only when one of the bids is accepted by a job agent. Since auctions are not necessarily cleared before other auctions are started, agents have a chance to "change their mind" if the situation changes. This extension helps to overcome the eager-bidder problem and thereby produces better results.

Another possible way to tackle the same problem is to use leveled commitment contracts, as introduced by Sandholm and Lesser (2001). Leveled commitment contracts represent agreements between agents that can be withdrawn. If a truck agent finds a new job that fits better, it can decommit an already committed job and take the new one. Hoen and Poutré (2003) employ truck agents that bid for new jobs considering decommitting already assigned ones. They show that decommitment yields better plans in a single-company cooperative case.

Returning to Fischer's reasoning, however, the primary reason for using decentralized agent solutions is that they are usually expected to outperform central optimization methods in problem instances with high levels of uncertainty. Researchers seemingly take this for granted and allocate their research efforts to demonstrating the value of their decentralized algorithm against other decentralized algorithms. Experiments studying the behavior of agent based methods over varying levels of uncertainty in comparison to optimization methods are generally absent from the literature. This is especially true in the logistics and transportation domain where the lack of appropriate comparisons between agent-based approaches and existing techniques appears to indicate a belief on the part of agent researchers that agent-based systems outperform traditional methods (Davidsson et al., 2005).

In this work, we dare to question this underlying assumption professed by agent enthusiasts. If advanced swapping and decommitment techniques are used, can fully decentralized agents really perform competitively with (or better than) centralized optimization in highly uncertain settings? Can the time gained in doing local operations compensate for the loss of not considering crucial global information? In our opinion these questions have not been fully answered. Our goal is to scrutinize these prevalent assumptions in the agent literature by studying an agent-based system in comparison to an optimization based approach for a real-world dynamic transportation problem. In the following section we describe our experimental design employed to perform this comparison.

3 Experimental Design

Through cooperation with a Dutch logistics service provider (LSP), we received the inspiration for our problem, as well as the data required to test our ideas. The Dutch logistics service provider, participating in this study, dedicates a portion of its business to transporting refrigerated ("reefer") containers from/to the Port of Rotterdam to/from various customer locations in the Netherlands. Approximately 40 trucks transport an average of 65 containers per day in this operation. The containers arrive on container ships arranged by customers. They are off-loaded at sea terminals, where trucks must then pick them up. The containers are then transported to their destination at the customer, where they are emptied. The trucks provided by the Dutch LSP wait at the customers' site until the containers are emptied, they then return the container to a sea terminal. The return terminal may be the same terminal from which the container originated or it may be a different terminal. For export containers the sequence is the same, the only difference is that the containers are not emptied, but loaded at the customer's location. At each location there are time windows within which trucks can make their visits. At sea terminals the time windows correspond to the opening hours of the terminal. At customer sites, the time windows are defined by

the customers. Each day the LSP must plan a set of routes capturing as much business as possible at minimum cost.

Our objective is to compare the performance of a traditional optimization based approach to an agent based approach in determining a feasible set of routes for one day of execution within this PDPTW. In order to examine the advantages/disadvantages of these two methodologies, we used data from the LSP to generate experimental datasets exhibiting two sources of uncertainty (service time duration uncertainty and job arrival time uncertainty). The performance of each solution is compared in terms of empty distance traveled, the lost profit of rejected jobs, and the lateness of the vehicles in reaching each job pick-up, delivery, and return location. This section describes the two solution approaches, the datasets, and the sources of uncertainty.

3.1 Solution Approaches

The primary objective of the planning methods is to route a uniform fleet of forty trucks on the Benelux road network at lowest cost without violating time windows. Costs consist of time traveling empty plus the penalty for rejected jobs. Jobs may be rejected when they cannot be served within the time restrictions. The penalty for rejecting a job equals the loaded time of that job. The loaded time of a job is the time from the start of the pick-up action to the end of the return action–including all loading, unloading, and traveling time. This is an appropriate penalty for a rejected job as it represents the profit lost in not serving the job. As we simulate only one planning day, in each instance, rejected jobs are simply rejected, although in practice they are reconsidered for service the next day. We now describe the mechanisms by which each solution approach solves this operational problem.

3.1.1 Agent-based Solution Approach

In building our agent system, our goal was to define a fully decentralized solution approach, in which no information regarding different agents or different sub-problems is concentrated at a higher level. Additionally, we wanted to equip our agents with state-of-the-art mechanisms to successfully solve the routing problem.

Following the traditions of Fischer et al. (1995), we modeled every truck as an agent. But, instead of defining company agents on a higher hierarchical level to sell jobs to truck agents on auctions, we modeled the containers themselves as agents. Each container agent sells itself on an auction to the truck agents. Truck agents use an extended insertion heuristic to bid on the auctions, and a container-exchange heuristic to improve the current solution. Container agents also actively try to improve the current solution by a reallocation heuristic.

Auctioning Containers Container agents organize auctions immediately after the container is announced to the planning system. If containers are revealed to the system at well-spaced intervals, then the auctions are fully sequential. If, however, multiple containers are announced at about the same time, those auctions are held in parallel. This parallelism is the result of our decision to model every container as an agent. This way auctions are held by separate agents instead of a central company agent. Although this requires extra coordination to handle parallel auctions, introducing a central entity to hold sequential auctions would have been against our design goal of having a totally flat MAS architecture.

The container agents collect quotes for transportation via these auctions. The bids truck agents send in for auctions contain the cost they would incur should they win the auction and transport the container. This includes the *loaded time* of the container, plus the *extra costs* of driving to that container empty, and driving to the next container also empty. A container agent only accepts bids that are less than its reservation price. The reservation price is the sum of the loaded time and the rejection penalty of the container (thus twice the loaded time). In this way, containers that have an extra cost, higher than their rejection penalty, are rejected.

Container agents implement a single-shot second-price closed-bid (a.k.a. Vickrey) auction. This auction type is popular in the literature because of its simplicity (Hoen and Poutré, 2003). We use this mechanism because, by setting the price to the second-best bid, the market position of the container

is implicitly communicated to the winning truck. This information is used by truck agents in making decisions, as explained later. Having the second-best bid as the price also ensures that the truck agent realizes a profit. If the winner is the only truck agent bidding on the auction, or the second-best bid is higher than the reservation price, the container agent sets the price of the contract to the reservation price.

The winning truck agent can accept the contract only if its plan is unchanged since the time of the bidding. If the plan has changed in the interim, due to winning another container, for example, the truck agent must re-calculate the transportation costs for this container considering the new plan. If the new costs are less than or equal to the price in the contract, the truck agent accepts the second container. Otherwise it has to reject it, since transporting the container in this new situation costs more than the price it would receive for the job. If the winner rejects the container, the container agent will try to close a deal with the second, the third, etc. truck agents in a similar manner.

When a container agent succeeds in making a contract with a truck agent, it sends a message to the other containers to notify them about the changed state of the contracted truck agent. This information is crucial for rejected containers that can try to re-auction themselves in the hope that they will have some options now that the situation has changed. Every container agent has a latest possible auctioning time. After that time it is not possible to pickup and transport the container within the given time windows. Container agents try to re-auction themselves only if this latest possible auction time has not yet passed. To avoid an avalanche of auctions from rejected container agents every time a contract has been made, re-auctions take place at randomly chosen intervals with an exponential distribution and a mean value of one-tenth of the time remaining for the container to be successfully scheduled.

Insertion with Substitution When a truck agent bids on a container, its bid includes the additional cost it would incur should it win the auction and transport the container. To calculate this additional cost, a truck agent has to compute the difference between the cost of executing its plan with the new container included and excluded. In general, a truck agent needs to solve a TSP-like problem in order to find the optimal order of the containers in its plan. Similar to other agent based methods, our agent system uses a fast heuristic, rather than a full optimization scheme. The fast heuristic our truck agents employ has appeared in previous work (Mahr et al., 2008). For the convenience of the reader, we describe our insertion and substitution heuristic again here. Algorithm 1 summarizes the extended insertion algorithm.

In step i of our insertion and substitution heuristic, a truck agent computes the cost of inserting container k before container i and the cost of substituting container i by container k. The cost of inserting container k between container i and j is calculated as the difference of the empty-travel times with and without container k: $ins_{ij}^{k} = d_{ik} + d_{kj} - d_{ij}$, where d_{xy} is the time the truck travels empty from the drop off location of container x to the pick-up location of container y. The cost of substituting container i, which is between container h and j in the plan, with container k is defined as: $subs_{i}^{k} = ins_{hj}^{k} + profit_{hj}^{i}$. The first term, inserting container k between h and j is as defined above. The lost profit of container i, $profit_{hj}^{i}$, is computed as the price offered for container i minus the loaded time and the insertion cost of container i. Thus, $profit_{hj}^{i} = price^{i} - d_{ii} - ins_{hj}^{i}$. The loaded time is subtracted because it is also part of the bids trucks submit. The insertion cost element of the lost profit is recalculated for every substitution decision, because the preceding or the subsequent container of i may have been changed since it was included in the plan.

This algorithm is linear (considering the insertion of one new container into the plan of a single truck), but the solution found can be arbitrarily far from the optimal, which might only be found by fully reordering all containers. In addition to calculating costs, our trucks also check time-window constraints, and do not accept any solution that violates time windows. The bid that is finally submitted to the container agent is the sum of the lowest insertion or substitution cost and the loaded time of the container, which we denote by d_{ii}.

The advantage of computing both insertion and substitution costs stems from the fact that insertion alone is very sensitive to the order of job arrivals. By allowing trucks to substitute earlier-committed containers, we reduce this ordering-sensitivity problem. Furthermore, the substitution cost expression we

use supports the replacement of containers that had many similar competing bids. The price of such a container, computed from the second-best bid, is only a little bit higher than the associated costs. This means that the profit is low, therefore it becomes a good candidate for replacement. At the same time such a container can easily find another truck for a similar price, since there was at least one truck that bid very close to the winner. Along the same lines, the proposed calculation prevents the substitution of containers that could have difficulties in finding another truck for a similar price.

Algorithm 1 Insertion and substitution of jobs

1. Iterate over the plan and collect the insertion and substitution costs corresponding to positions in the plan.
2. Sort the merged list of insertion and substitution costs and positions by increasing order of costs.
3. Iterate over the list of costs and positions, and
   (a) if the position indicates a substitution and the insertion cost of the new container is not less than ϵ less than the insertion cost of the container currently in the problem, then drop this alternative,
   (b) if it is not possible to insert or substitute the new container at the given position without violating its time windows, then drop this alternative,
   (c) if the time windows of the subsequent containers are violated by the insertion or substitution of the new container, then drop this alternative, or
   (d) else return the position and the cost as the cheapest feasible insertion or substitution position.

If a truck wins an auction where its bid corresponds to an insertion position, then it simply inserts the new container to the specified position. If the bid corresponds to a substitution position, then the truck first releases the container in that position and then inserts the new container to its place.

The released container starts a new auction just as it did following its arrival time. This new auction may, of course, result in another substitution, which invokes a new auction, and so on. The seemingly endless chain of substitutions is limited by the rule that truck agents accept containers for substitution only if they fit better (by at least ϵ) in their plan than the one that is to be replaced. This ϵ can be chosen in such a way as to guarantee a bound on the length of substitution chains. For example, if ϵ is chosen as a fraction of the maximum possible improvement, the length of the resulting substitution chain is bounded by a constant. With any choice of ϵ, the end of a substitution chain is either an insertion or rejection, which occurs if all truck agents reject the container agent in the auction due to time-window infeasibility, or if all bids are higher than the reservation price of the container.

In addition to the basic task of bid calculation, both the container and truck agents are endowed with additional techniques to improve the initial solution.

Random Reallocation Whenever the plan of a truck agent changes, in principle any container agent has a chance that the truck agent with the new plan can now transport it at a lower cost than their current truck agent. Rejected container agents receive notification about such events, therefore they can try to close a deal with the truck agent with the changed state. Since it is also useful for container agents that already are in possession of a contract to try to find better options, they use a randomized algorithm to search for those options.

Every container agent has a timer that fires in exponentially distributed random intervals with a mean value of μ_r. Whenever the timer goes off and the container agent has a contract, it tries to reallocate itself. Reallocations are randomized this way to minimize the chance of two re-auctions happening at the same time. In our simulations, with 65 container agents per day, we selected μ_r to be one hour. This yields approximately one reallocation per minute. To commence reallocations, the container agent, whose timer has fired, sends a message to the truck agent, with which it has a contract, in order to prevent the

truck from transporting it. The truck agent puts the container *on hold*, and reports the current insertion cost of the container to the container agent. Then the container agent re-auctions itself among the other truck agents to see if any of them offers a better price than its current insertion cost. The container agent only accepts new offers that are better than this limit. If the container agent finds a cheaper option, it breaks its current contract and makes a new one with the new winning truck agent. If not, it sends a message to the current contracted truck agent in order to remove the hold, allowing the truck to transport it. The steps of this algorithm are summarized in Algorithm 2.

Algorithm 2 Reallocation

1. The container agent sends a message to its contracted truck agent to prevent the truck to transport it.
2. If the transportation of the container has not started yet, the truck agent puts the container on hold, and sends back its current insertion cost.
3. The container agent re-auctions itself among the other truck agents and collects offers that are better than its current insertion cost.
4. The container agent sorts the list of collected offers from best to worse, and iterates through the list.
   (a) Notify the truck agent that it won the auction.
   (b) If the truck agent accepts the new contract, the container agent notifies the previous truck agent that it leaves. The previous truck agent removes the container from its plan, and the algorithm stops.
5. If there are no offers left (or the list was empty because no new offer was better the current one) the container agent sends a message to the currently contracted truck to remove the on-hold status.

By periodically attempting a reallocation, container agents check new possibilities that arose from changes in the plans of trucks since the last auction. In our agent system, truck agents also actively try to improve the solution. The next section discusses a decentralized algorithm that exchanges containers between pairs of trucks.

Exchange of Containers An efficient way to improve solution quality in vehicle routing problems is to try to exchange jobs between trucks as noted in Section 2. Our decentralized algorithm of container exchanges searches a neighborhood of 2-cyclic k-exchanges in a randomized fashion. Like container agents, truck agents also have a timer set to fire at exponentially random intervals with a mean value of μ_e. For similar reasons explained at the reallocation algorithm, μ_e is chosen to be one hour in our experiments. When a truck agent's timer goes off, it initiates a container-exchange procedure. It first checks if there are any containers in its plan that are not executed yet. These containers are available for exchange with another truck. The truck agent puts the first exchangeable container on hold, to prevent it from being executed. Then it copies the relevant part of the plan and sends it to another truck. In principle, truck agents can apply sophisticated heuristics to choose a partner truck. Geographical coordinates, personal preferences of trucks, or business considerations of the company can be taken into account. However, these heuristics rely on the availability of appropriate information regarding all (or a subset of the) trucks. Maintaining such information implies the aggregation or centralization of data. As one of our goals was to design and test an agent system in which no information is concentrated, our truck agents do not discriminate in choosing a partner truck; selection of a partner truck is made randomly.

The truck that receives the "not-yet-executed" chunk of plan, produces a similar sub-plan from its own plan. It also puts the first exchangeable container on hold, and then performs a full search on the k-exchange neighborhood of the two plan segments. First it tries to exchange single containers, then chains

of two, three, etc. containers in any combination. In the case of our test instances, the plan segments are never longer than three containers. This search returns the exchange combination of containers that yields the highest saving for the *combination* of these two trucks. If a solution is found, it is reported back to the initiator truck.

If the initiator receives a certain container-exchange combination from the responder truck, it implements the exchange in its plan, and sends new contracts to newly assigned container agents. The last step of the initiator is to send the expired contracts of the exchanged containers to the responder truck. To conclude the procedure, the responder also implements the container exchanges in its plan, and sends new contracts to the newly received container agents. The container-exchange algorithm can thus be explained as a four-step negotiation between the initiator and the responder trucks. Algorithm 3 summarizes all the steps.

Algorithm 3 Container Exchange

The Initiator truck

1. puts the first exchangeable container on hold,
2. copies the segment of its plan that contains only exchangeable containers, and
3. sends this segment to a randomly chosen other truck.

The Responder truck

1. also puts its first exchangeable container on hold,
2. makes a copy of the exchangeable part of its plan,
3. performs a full search on the k-exchange neighborhood of the two plan segments, and
4. sends back the best exchange combination to the initiator, if one is found that leads to better plans.

The Initiator truck

1. implements the proposed exchange,
2. sends new contracts to the newly acquired container agents, and
3. sends back the expired contracts of the exchanged container agents to the responder.

The Responder truck

1. implements the proposed exchanges in its plan, and
2. sends new contracts to the newly acquired container agents.

The above described container-exchange algorithm implements a randomized search on the 2-cyclic k-exchange neighborhood of the container transportation problem. The search is performed in a decentralized fashion, and it requires cooperative truck agents. Truck agents are required to reveal parts of their plan to other truck agents. The responder agent searches for an exchange that maximally decreases the costs of the two truck agents together. Such considerations prohibit the direct application of this algorithm in a competitive agent system.

Let us now analyze the run-time requirements of the algorithms the agents use. In order to be able to react quickly, the run-time complexity of all these algorithms is kept polynomial. In the insertion and substitution algorithms, computing the insertion and substitution costs is linear in the number of containers (N), ordering the costs is $O(N \log N)$, and checking the time constraints is $O(N^2)$. The container exchange algorithm consists of a full search of the exponential k-exchange neighborhood, but we limit the size of the neighborhood by three (in our experiments because this is the maximum number of containers a truck can transport per day), thus the algorithm runs in quasi-linear time. During the

auctions, all K trucks submit a bid, the bids are computed in $O(N^2)$ time, and clearing an auction is polynomial in the number of trucks ($O(K \log K)$). The run-time of this approach thus mainly depends on the number of auctions. Auctions are held for three different reasons. First, there are auctions that container agents organize when they arrive. In the worst case, the sum of all such auctions is in the order of ($O(\frac{N^2}{K})$). Second, there are auctions caused by substitutions. The bound on the length of substitution chains depends on the ϵ parameter of Algorithm 1. In our experiments, we chose ϵ to be small enough to allow any substitution that strictly improves the solution, but dependent on the maximum possible cost of a container. Consequently, the number of auctions in a substitution chain is limited by a constant C. The number of substitution chains is at most the number of successful initial auctions, thus in the worst case there are CN auctions caused by substitutions. Finally, auctions may be held during a reallocation attempt. These happen a constant number of times because such attempts are made periodically regardless of the number of containers. The worst case is that each of these auctions start a substitution chain, which generates auctions in the order of $O(N)$. To summarize, the number of auctions is polynomial ($O(\frac{N^2}{K})$) in the number of containers. Considering the complexity of the algorithms and the number of auctions, we can conclude that the agent solution runs in polynomial time ($O(\frac{N^4}{K} + N^2 \log K)$), and scales well both in the number of trucks and jobs.

The insertion, substitution, random reallocation, and the exchange of containers all contribute to the quality of the routes produced for each truck. In particular, these methods only change the plan if doing so leads to a strict decrease of the costs. Each of these algorithms has been used previously in multi-agent transportation planning systems, but to the best of our knowledge, their combination is unique. From this we conclude that the approach presented here is a very good representative of agent methods, and as such is a good candidate for comparison to a centralized optimization-based approach.

3.1.2 Optimization-based Solution Approach

Our objective in selecting a comparative approach, to the decentralized agent based approach, was to choose a proven approach that depends on the centralization of full system information. At the core of the optimization-based solution approach, that we selected, is a mixed integer program (MIP) for a truck-load pick up and delivery problem with time windows. This MIP is passed to CPLEX for solving (ILOG, Inc., 1992). The formulation used here is based on one put forth by Yang et al. 1999. This model was selected because of its ability to provide competitive results in comparison to fast running heuristics (Yang et al., 2004). The on-line optimization approach used in our experiments works by invoking the MIP at 30-second intervals. We selected 30-second intervals in order to give the MIP sufficient time to find a good feasible solution while also remaining competitive with the event driven agent-based solution approach. As the on-line optimization approach may only find a feasible (not necessarily optimal) solution in each 30-second interval, this approach acts more like a heuristic in the on-line context. We therefore believe that the centralized on-line optimization approach makes a good comparison to the decentralized agent approach. The complete description of our modifications to and use of Yang et al.'s MIP in our on-line optimization scheme is the focus of this section.

The Mixed Integer Program Before introducing the notation and mathematical formulation for this problem, we begin with a small example to illustrate exactly how Yang et al.'s MIP works to exploit the structure of this truckload pick-up and delivery problem with time windows. Imagine a scenario with three trucks and four jobs. The model of Yang et al. is constructed such that it will find a set of least cost cycles describing the order in which each truck should serve the jobs. For example, as depicted in Figure 1, the outcome may be a tour from truck 1 to job 1, then job 2, then truck 2, then job 3, then back to truck 1. This would indicate that truck 1 serves job 1 and 2, while truck 2 serves job 3. The cycle including only truck 3 indicates that truck 3 remains idle. Similarly, the cycle including only job 4 indicates that job 4 is rejected.

Given this problem description, we designate the following notation for the given information.

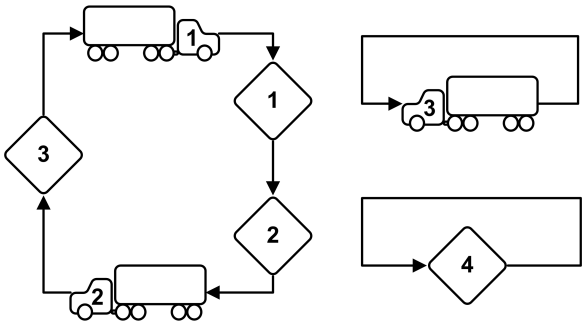

Figure 1: Cycles in the MIP solution structure.

K the total number of vehicles available in the fleet.
N the total number of known demands.
d_{ij} as introduced in 3.1.1, the travel time required to go from demand i's return terminal to the pick-up terminal of demand j. Note, if $i = j$ then the travel time d_{ii} represents the loaded distance of job i; this distance includes the time from pick up at the originating terminal to completion of service at the return terminal.
d^k_{0i} the travel time required to move from the location where truck k started to the pick-up terminal of demand i.
d^k_{iH} the travel time from the return terminal of demand i to the home terminal of vehicle k.
v^k the time vehicle k becomes available.
τ_i^- earliest possible arrival at demand i's pick-up terminal.
τ_i^+ latest possible arrival at demand i's pick-up terminal.
M a large number set to be $2 \cdot \max_{i,j}\{d_{ij}\}$.

Note that τ_i^- and τ_i^+ are calculated to ensure that all subsequent time windows (at the customer location and return terminal) are respected.

Given the problem of interest, we specify the following two variables.

x_{uv} a binary variable indicating whether arc (u, v) is used in the final routing; $u, v = 1, \ldots, K + N$.
δ_i a continuous variable designating the time of arrival at the pick-up terminal of demand i.

Using the notation described above, we formulate a MIP that explicitly permits job rejections, based on the loaded distance of a job.

$$\begin{aligned} \min \quad & \textstyle\sum_{k=1}^{K}\sum_{i=1}^{N} d^k_{0i} x_{k,K+i} + \sum_{i=1}^{N}\sum_{j=1}^{N} d_{ij} x_{K+i,K+j} \\ & + \textstyle\sum_{i=1}^{N}\sum_{k=1}^{K} d^k_{iH} x_{K+i,k} \end{aligned}$$

such that

$$\sum_{v=1}^{K+N} x_{uv} = 1 \qquad \forall u = 1, \ldots, K+N \tag{1}$$

$$\sum_{v=1}^{K+N} x_{vu} = 1 \qquad \forall u = 1, \ldots, K+N \tag{2}$$

$$\delta_i - \sum_{k=1}^{K} (d_{0i}^k + v^k) x_{k,K+i} \geq 0 \qquad \forall i = 1, \ldots, N \tag{3}$$

$$\begin{aligned} &\delta_j - \delta_i - M x_{K+i,K+j} + \\ &\quad (d_{ii} + d_{ij}) x_{K+i,K+i} \\ &\quad \geq d_{ii} + d_{ij} - M \end{aligned} \qquad \forall i, j = 1, \ldots, N \tag{4}$$

$$\tau_i^- \leq \delta_i \leq \tau_i^+ \qquad \forall i = 1, \ldots, N \tag{5}$$

$$\delta_i \in \mathbb{R}^+ \qquad \forall i = 1, \ldots, N \tag{6}$$

$$x_{uv} \in \{0, 1\} \qquad \forall u, v = 1, \ldots, K+N \tag{7}$$

In words, the objective of this model is to minimize the total amount of time spent traveling without a profit generating load. Specifically, we wish to minimize the penalty incurred from rejecting jobs, time spent traveling empty to pick up a container, between containers and when returning to the home depot. This objective is subject to the following seven constraints:

(1) Each demand and vehicle node must have one and only one arc entering.

(2) Each demand and vehicle node must have one and only one arc leaving.

(3) If demand i is the first demand assigned to vehicle k, then the start time of demand i (δ_i) must be later than the available time of vehicle k plus the time required to travel from the available location of vehicle k to the pick up location of demand i.

(4) If demand i follows demand j then the start time of demand j must be later than the start time of demand i plus the time required to serve demand i plus the time required to travel between demand i and demand j; if however, demand i is rejected, then the pick up time for job i is unconstrained.

(5) The arrival time at the pick up terminal of demand i must be within the specified time windows.

(6) δ_i is a positive real number.

(7) x_{uv} is binary.

Mathematically this model specification serves to find the least-cost (in terms of time) set of cycles that includes all nodes given in the set $\{1, \ldots, K, K+1, \ldots, K+N\}$. We define $x_{uv}, (u, v = 1, \ldots, K+N)$ to indicate whether arc (u, v) is selected in one of the cycles. These tours require interpretation in terms of vehicle routing. This is done by noting that node $k, (1 \leq k \leq K)$ represents the vehicle k and node $K+i, (1 \leq i \leq N)$ corresponds to demand i. Thus, each tour that is formed may be seen as a sequential assignment of demands to vehicles respecting time window constraints.

The model described above is used to provide the optimal (yet realistically unattainable) lower bound for each day of data in the experiments concerning job arrival uncertainty. We denote this approach as the static *a priori* case. In this case, we obtain the route and schedule as if all the jobs are known and we have hours to find the optimal solution. Thus, not only is this lower bound realistically unattainable due to a relaxation on the amount of information available, but also due to a relaxation on the amount of time available for CPLEX to obtain the optimal solution.

Optimization in an On-line Context In order to provide a fair comparison with the agent-based approach, the MIP is manipulated for use in on-line operations. In our on-line approach, this MIP is invoked at 30 second intervals. At each interval, the full and current state of the world is captured, and then encoded in the MIP. This "snapshot" of the world includes information on all jobs that are available and in need of scheduling, as well as the forecasted next available location and time of all trucks. The MIP is then solved via a call to CPLEX.

The decision to use 30 second intervals was driven by the desire to be comparable to the agent-based approach while still providing CPLEX enough time to find a feasible solution for each snapshot problem. While this interval may seem extremely short for finding a feasible solution, it is not as damaging as one might think. First, in the baseline dataset, we see that even with all of the jobs arriving in one clump at the start of the day, the on-line optimization rejects no jobs. Second, in most datasets, the snapshot problem being solved at each decision instance is significantly smaller than the full problem size.

The solution given by CPLEX, at the end of the 30-second interval, is parsed and any jobs that are within two intervals (i.e. 60 seconds) of being late (i.e. missing the time specified by δ_i in the latest plan) if travel is not commenced in the next interval are permanently assigned. Any jobs that were designated for rejection in the solution are permanently rejected only if they are within two intervals of violating a time window; otherwise they are considered available for scheduling in a subsequent interval.

If CPLEX cannot find an initial feasible solution in any one interval then the plan from the last feasible interval is invoked and parsed to fix assignments and rejections as described. If CPLEX cannot find an initial feasible solution for three consecutive intervals (90 seconds) then one job is selected for rejection based on the following hierarchy:

1. a job that arrived since the last feasible plan was made.
2. a job with a loaded distance less than or equal to 13000 seconds.
3. a job that is 30 minutes away from the end of its time window.
4. a random job.

The procedure continues solving problem instances and parsing solutions in this fashion until the end of the working day at which point all jobs have been served or rejected.

3.2 The Data

In this subsection, we describe how our data was inspired and fed by the operations of the Dutch LSP. Recall that the LSP is transporting comparatively high-value reefer containers. As such, the trucks always wait at customer sites for the containers to be (un)loaded, and they never exchange containers. We therefore handle each pick-up, delivery, and return sequence as one job. Note, more than one job starts and ends at the same terminal locations. Moreover, some customers have more than one job serviced in a day. Nevertheless, we handle each job separately, as if they all belonged to different customers. Each job is specified by two data vectors–one spatial and one temporal.

The spatial vector contains the location of the pick-up terminal, the customer site, and the return terminal. This data was derived from a set of operational data tables provided by the LSP. In all, we were given data from January 2002 to October 2005 as well as from January 2006 through March 2006. The tables represented jobs that were planned to be served on a given day. Unfortunately, the exact timing of the jobs each day was nearly absent from the data. Further problems were presented by some of the addresses that referred to postal boxes instead of real customer or terminal locations; therefore these had to be pruned from the data. Nevertheless, after a preliminary review of the data, we could conclude that on average 65 jobs were served in a day, at customer and terminal locations associated with less than 25 distinct zipcodes. The rare timing information suggested that the jobs were served uniformly throughout the day. Using these parameters, we extracted a random sample of appropriately defined jobs from the original data-set in order to generate a set of 33 days with 65 jobs per day using the locations in the sample. Figure 2 depicts the geography of the Netherlands and the full set of locations represented in our data.

Figure 2: All locations in the Netherlands. Black markers indicate customer locations; grey markers indicate terminal locations; and the white marker indicates the home terminal of the LSP

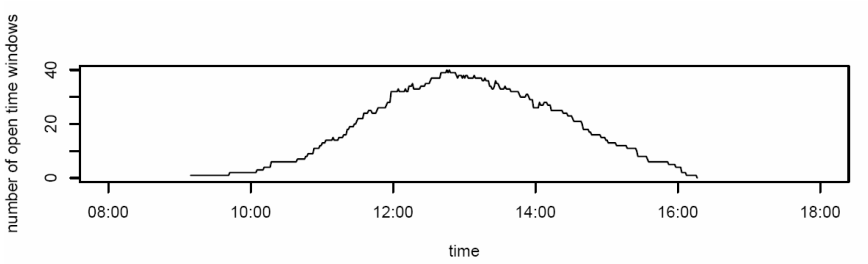

Figure 3: Number of open time windows for all jobs throughout the working day.

The temporal vector is comparatively more complex–containing three data types: data on time windows, data on service times, and data on job arrival. The data on time windows includes the terminal operating time windows and the customer time window. The data on service times includes the service time required at the three job locations. Finally, the data on job arrival includes one element – the time the job is announced in the planning system. As mentioned earlier, such timing information was sparsely recorded in the data tables, therefore this part of the job descriptions is entirely generated based on the experiences of the human planners.

To standardize the data for our experimental purposes we specified time windows at all locations as follows: terminals are open for pick up between 6am and 6pm, and for return between 6am to 5:59am on the next day. The wide return time windows reflect the practice that trucks can bring containers to the terminals on the following day, if they were too late on the same day. These time windows are the same for all jobs. Delivery time windows are set to two hour intervals, and their start times are distributed uniformly over the working day between 8am and 5pm. Figure 3 displays the number of open time windows for all jobs at any time point of the working day. Since time windows open regularly and stay open for two hours, the number of open time windows gradually builds up and reaches the top between 12am and 2pm. After that, the number of open time windows, and therefore the number of jobs requiring service before the end of the day decreases.

The service time data type refers to the time trucks need to complete service at the different locations. When a truck arrives at a sea terminal or a customer, it spends some time to pick up, to deliver, or to return a container. The length of this time depends on various factors. Picking up a container for example, can be delayed by customs clearing, paperwork, or problems with putting the container on the truck. Emptying a container at the customer can be quick if the customer is ready to unload the goods, but it can be delayed if a warehouse is very busy. Similarly, when a container is returned, technical issues may delay the trucks. In discussions with the LSP, it seems that the human planners, by experience, allocate one hour for picking up, one hour for delivering, and half an hour for returning a container. As such, the baseline dataset (referred to as R_0) sets all service time values to these times for all jobs.

The job arrival data type refers to the time a container is made known to the planning system. Before this time, the planning methods do not know about the job, after this time, the locations and the time windows are revealed. In the baseline dataset all job arrivals occur at the start of the day, 6am.

In all instances that we generated, we added a homogeneous fleet of 40 trucks starting at a base location close to Rotterdam. Although the number of trucks used by human planners varies each day, we chose to use 40 trucks, because this proved to be enough to solve each problem *a priori*, in the baseline dataset. Using more trucks would not yield different results. Using fewer trucks would yield a higher rejection rate, with possible differences in the kind of jobs rejected by the two methods.

Within the baseline dataset, each job requires, on average, approximately 4.2 hours of loaded distance. When the routing is optimal (in the *a priori* baseline case in which all jobs are known at the start of the day and all service times equal their expected values) the average empty time per job is approximately 25 minutes (or 27 hours total per day).

Table 1: Summary of variation and corresponding bounds on service times for service time uncertainty scenarios.

Experiment	Bound on Pick-up and Delivery Service Times	Bound on Return Service Times
10 minutes	[50min, 70min]	[20min, 40min]
20 minutes	[40min, 80min]	[15min, 50min]
30 minutes	[30min, 90min]	[15min, 60min]
4 hours	[30min, 5hours]	[15min, 4.5hours]

3.3 Uncertainty Scenarios

The objective of this work is to determine how the two solution approaches perform on the given pick-up and delivery problem with time windows under different types of uncertainty. The three main sources of uncertainty encountered by the drayage company are: service time uncertainty, travel time uncertainty, and job-arrival time uncertainty. The complexity of pursuing travel time uncertainty in a correct way, made us decide to use a deterministic estimate of the travel time, and focus only on uncertainty regarding service-time and job arrivals. Note, however, that when periodic changes in the travel time are included in the simulations and taken into account by both approaches, it should not influence the results. The main difficulty in simulating with uncertain travel times is the part that is identifying which trucks, traveling which routes, are subject to the variation. Nevertheless, up to a certain extent, such travel-time uncertainty is not that different from our model of service-time uncertainty in the case where service times are dependent upon each other.

3.3.1 Scenarios with Service-Time Uncertainty

In reality service times vary, forcing plans to be updated in real-time (or more generally, after every pick-up, delivery, or return action). To simulate this source of uncertainty, we define different service times for every pick-up, delivery, or return action drawn from a uniform random distribution. This exact information is, however, hidden from the planning methods, which consider the mean service times as derived from human experience, and encounter the variable service times only during execution. Service time uncertainty may be viewed from two perspectives. The first perspective is that incidents affecting service times are random and occur in a uniformly distributed way across the day and across terminal and customer locations (*independent*). The second perspective comes from recognizing that service time disruption events are most likely clustered in both time and location (*dependent*).

In scenarios with independent service times, we generated durations for all actions as random variables drawn independently from a uniform distribution. The boundaries of the uniform distributions for each scenario were defined with the intention to generate four different scenarios with service-time variations of 10, 20, 30 minutes, and 4 hours. The 10, 20, and 30 minutes are common delays according to the LSP; 4 hours is included to view any effect in exaggeration. Table 1 summarizes the bounds on the service times for each service action type as well as the nominal values used to denote the scenarios. Note that the lower bound of the return actions was not allowed to fall below 15 minutes. This was based on the assumption that returning a container can never be quicker than 15 minutes. Similarly, in the most extreme case, the lower bounds on pick up and delivery actions were minimized at 30 minutes. Furthermore, it is important to note that regardless of the bounds, the planning systems always planned using the values experienced by the human planners.

In addition to generating service times for all actions independently, we also considered scenarios where service times at the same locations are always the same. To achieve this, we generated service times for every location in an instance independently according to a uniform distribution. Then, we always assigned the same service times to actions happening at the same locations. The scenarios were generated according to the same variations as in the independent case (summarized in Table 1). This resulted in four dependent service-time variations scenarios: Sd_{600}, Sd_{1200}, Sd_{1800}, and Sd_{14400}.

The motivation for the scenarios with dependent service times stems from practice. For example, if the cause of a delay at a sea terminal is a faulty crane then it will influence all trucks visiting that terminal. Note, such dependent delays have an effect similar to traffic jams, where trucks traveling toward the same location (and using the same roads) suffer from the same traffic conditions.

Considering this type of uncertainty in the context of the solution approaches, we note that when a truck agent experiences a change in its plan due to the actual service times, it tries to adjust the plan to the new situation. If simply shifting the remaining containers in time solves the problem, then that is done. If any subsequent containers become infeasible due to time-window violations, the truck agent removes the infeasible containers from its plan. The removed containers experience a removal similar to a substitution, and they start new auctions to find another truck agent. In this way, service-time variation is handled and the agents continue their improvement efforts as before.

As a comparison, the on-line approach has a more basic way of handling service-time variation. Whenever an actual service time is revealed, the planner adjusts the timings in the plan of the truck in question. Jobs that are already permanently assigned in a plan are not put back into the pool of unplanned jobs – even if they will now be late. In this way job rejections are minimized at the expense of being late. This is distinctly different than the *a priori* optimal which, with access to the service time data and in accordance with the constraints, will instead reject the jobs that will be late. As a result, a comparison of the on-line and *a priori* optimal solutions, in the case of service time uncertainty, would be inappropriate or unfair. Given these differences we only compare the agent-based and on-line optimization solution approaches for the service time uncertainty experiments ignoring the *a priori* lower bound.

3.3.2 Scenarios with Arrival-Time Uncertainty

In addition to service time uncertainty, human dispatchers are often faced with a great deal of uncertainty regarding the time when a container may enter the planning system. Although the load may be known to the transportation company beforehand (e.g. from ship arrival data), the exact moment that the container will actually be offloaded is often unknown. Some containers are handled on a previous day, or during the night, while others may become available only in the afternoon.

In every problem instance, we defined a job-arrival time for each container. The earliest job arrival was at 6am; containers with such an arrival time we call static jobs. We call them static because they are actually known to the planning systems when planning starts for the day. Containers with an arrival time later than 6am we call dynamic jobs, referring to the fact that the planning systems need to incorporate them into the plans during execution. When a container is randomly selected to be dynamic in a certain problem instance, we set its arrival time to exactly two hours before the start of its customer time window (i.e., four hours before the end of the customer location time window, leaving slightly less than two hours on average before the latest departure time from the pickup location). This choice was made to ensure that the planning methods were confronted with time pressure. On average, the distance to the time window cannot be shorter than two hours, because it would render the instances infeasible. It could be longer, but then the resulting instances would be easier to solve, since there would be ample time for planning.

We generated five different scenarios with varying levels of job-arrival uncertainty. The varying levels of uncertainty were expressed in the percentage of dynamic jobs present in the instances. The baseline dataset represents the zero percent scenario (R_0), where all jobs are known at the start of the working day, 6am. In the 25% scenario (R_{25}), one quarter of the jobs (selected randomly from the 65 containers) arrive two hours before their customer time window, and the rest are static. In the 50% scenario (R_{50}) half of the jobs are static and half of them are dynamic. In the 75% scenario (R_{75}), one quarter of the jobs are static. Finally, in the 100% cases (R_{100}) all containers arrive in a dynamic fashion.

The agents implicitly handle job-arrival uncertainty; they do not start their first auction before a job's designated arrival time. In the on-line optimization approach new arrivals are taken into account in the next epoch. For the job arrival uncertainty scenarios, we compare all three solution approaches (*a priori* optimal, on-line optimization, and agent-based).

Table 2: Scenarios with different sources of uncertainty

Scenario Title	Scenario Description
S_{600}, S_{1200}, S_{1800}, S_{14400}	Scenarios with 10, 20, 30 minutes, and 4 hours independent service time variations.
Sd_{600}, Sd_{1200}, Sd_{1800}, Sd_{14400}	Scenarios with 10, 20, 30 minutes, and 4 hours dependent service time variations.
R_0, R_{25}, R_{50}, R_{75}, R_{100}	Scenarios with zero, 25%, 50%, 75%, and 100% dynamic jobs.
$R_{50}S_{600}$, $R_{50}S_{1200}$, $R_{50}S_{1800}$, $R_{50}S_{14400}$	Scenarios with 50% dynamic jobs, and 10, 20, 30 minutes, and 4 hours independent service time variations.

3.3.3 Scenarios with Arrival-Time and Service-Time Uncertainty

Having new jobs arriving during the day, and experiencing variations in service times are both quite common in practice. In instances with service-time uncertainty (denoted S_* and Sd_*), all containers are static (they all arrive at the beginning of the day). In those instances the only source of uncertainty is service-time variations. Similarly, in instances with dynamic job arrivals (in R_* instances), all service times experienced during execution correspond to the average service times. To study the combined effect of both sources of uncertainty, we defined scenarios where dynamic job arrivals and variable service times are both present.

In conversations with the planners at the LSP, it was revealed that typically 50% of the containers served in one day are dynamic. We therefore based our combined scenarios on the R_{50} instances in which 50% of the containers are dynamic.

We generated four variations of the R_{50} instances with 10, 20, 30 minutes and 4 hour service time variations. The resulting scenarios are denoted as $R_{50}S_{600}$, $R_{50}S_{1200}$, $R_{50}S_{1800}$, and $R_{50}S_{14400}$ respectively. Consequently in these scenarios, in addition to 50% of the containers being dynamic, durations of all service actions are varied independently according to the same uniform distributions as the independent service-time variation case explained in Section 3.3.1.

Table 2 summarizes all the scenarios and their notation as used in this paper.

4 Results

The basis on which both methods are compared across all datasets is total routing cost. Specifically, total routing cost, as defined here, is the sum of the amount of time spent traveling empty, the penalty affiliated with rejecting loads, and the amount of time incurred from delivering jobs outside their appointed time windows. Note, in the design of both systems only time spent traveling empty and job rejection penalties are considered explicitly. Late penalties are a by-product of service time variability in an on-line setting. To demonstrate the effect of this design consideration, the results are presented in two formats. The tables presented in the following subsections include the mean and standard errors of the total routing costs, while the graphs illustrate the split of these mean total costs across their three component costs.

4.1 Service Time Uncertainty Results

The results highlighted here illustrate the capabilities of each system–agents and on-line optimization–to handle independent and dependent service time disruptions. Recall, we do not include the *a priori* optimal solution in this set of results as it would be unfair to compare a system that can never yield a solution permitting time window violations with solutions that allow such violations.

Figure 4 shows, by means of a bar chart, the mean routing costs of the agent and on-line optimization solution approaches for the case of independent service time variability. Of particular interest is the fact that the empty time (hours) attributable to both systems does not vary significantly across the five scenarios; for the on-line optimization the empty time remains consistently at 27.9 hours whereas for the

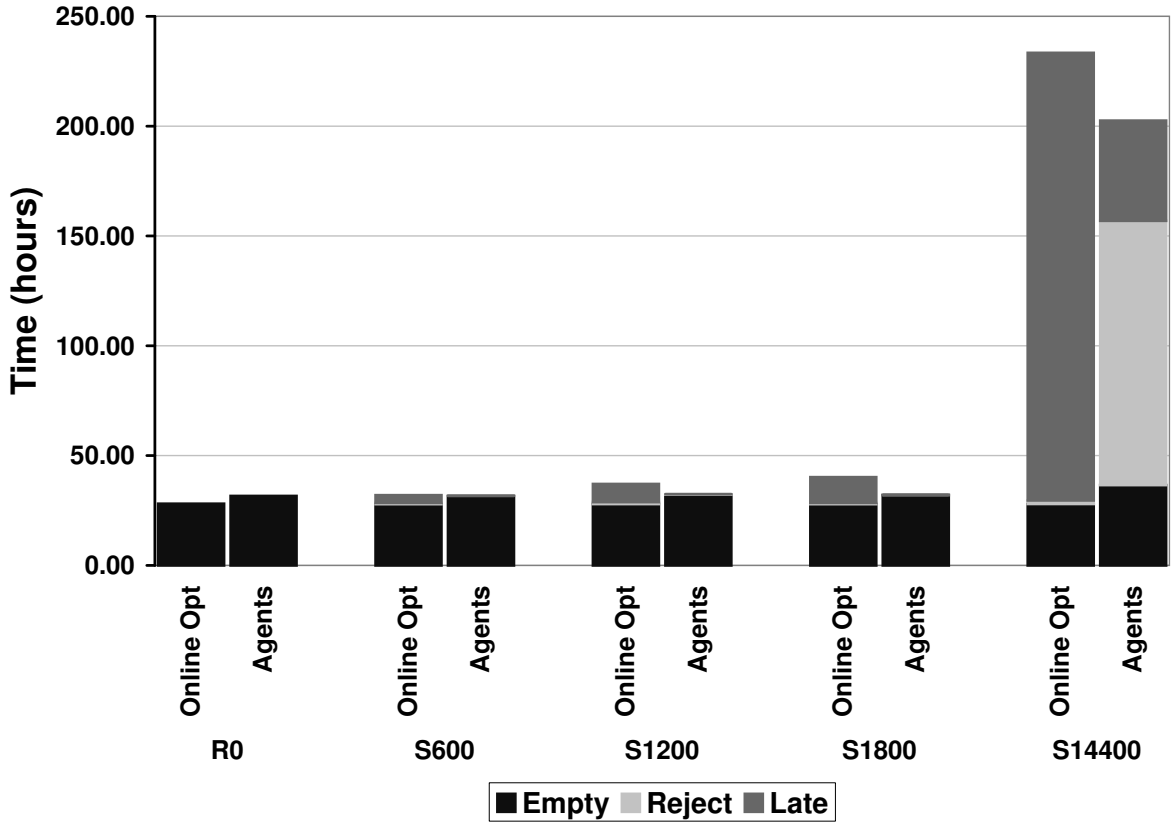

Figure 4: Mean over 33 days of the routing cost components (empty time, rejection penalty, late time) for the agent and on-line optimization solution approaches across five independent service time uncertainty scenarios.

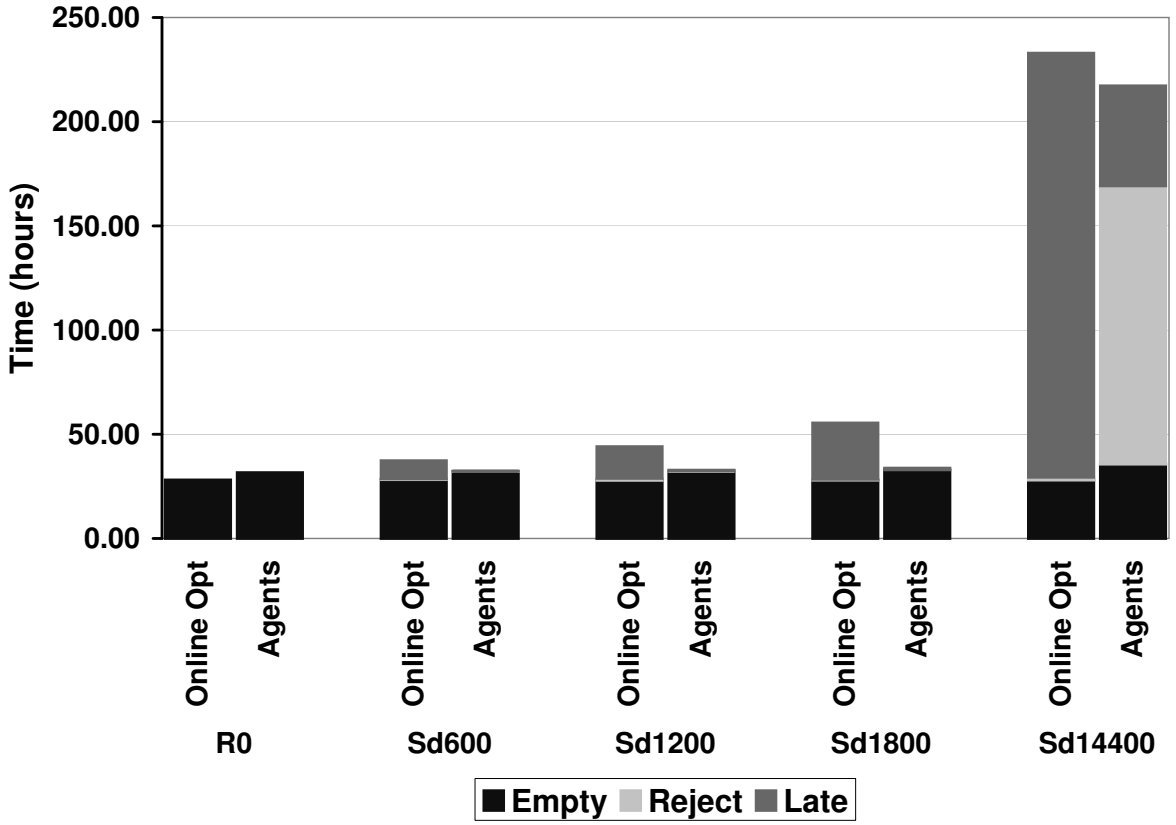

Figure 5: Mean over 33 days of the routing cost components (empty time, rejection penalty, late time) for the agent and on-line optimization solution approaches across five dependent service time uncertainty scenarios.

agents it ranges slightly from 32 to 36 hours. The primary difference, both across scenarios and between systems, can be seen in terms of the mean over the total amount of time late for the jobs' time windows. For the on-line optimization approach the amount of lateness increases sharply across the five scenarios – starting at 0 ranging through 3.5, 8.25, 11.7 and ending at 203.79 hours. The agents, however, have a comparatively small amount of lateness across the five scenarios – starting at 0 ranging through .02, .05, .14, and ending at 47.75 hours in the most extreme scenario. The penalty affiliated with job rejection is, on the other hand, relatively small for both systems until the most extreme case at which point the job rejection penalty of the agents (126.02 hrs) far exceeds that of the on-line optimization (1.62 hrs). We should note here that all of the jobs rejected (27 in total) in the on-line optimization were rejected due to the inability of the system to find a feasible solution within 90 seconds. While this might appear to be a major shortcoming of the on-line optimization, this represents only .0025% of all jobs examined across all S_* instances.

Figure 5 shows the mean routing costs of the agent and on-line optimization solution approaches for the case of dependent service time variability. Similar to the independent service time scenarios, the empty time (hours) attributable to both systems does not vary significantly across the five scenarios. In fact the values for both solution approaches in the dependent scenarios are also nearly identical to those obtained in the independent scenarios; for the on-line optimization the empty time remains consistently at 28 hours whereas for the agents it ranges slightly from 32 to 36 hours. Again, the primary difference, both across scenarios and between systems, can be seen in terms of the mean over the total amount

Table 3: mean ± std. error of total routing costs (lateness plus rejected penalties plus time spent empty) in hours for all service time uncertainty experiments; n = 33.

Dataset	On-line Optimization	Agents	Paired t-test: Difference in means equal to 0?
S_{600}	31.95 ± .68	31.62 ± .33	Fail to Reject, $p = .60$
S_{1200}	37.05 ± .86	32.44 ± .29	Reject, $p < .0001$
S_{1800}	40.17 ± 1.00	32.09 ± .36	Reject, $p < .0001$
S_{14400}	233.36 ± 3.19	202.51 ± 5.17	Reject, $p < .0001$
Sd_{600}	37.33 ± .94	32.33 ± .38	Reject, $p < .0001$
Sd_{1200}	44.06 ± 1.81	32.79 ± .39	Reject, $p < .0001$
Sd_{1800}	55.43 ± 2.65	33.77 ± .39	Reject, $p < .0001$
Sd_{14400}	232.89 ± 4.08	217.20 ± 9.56	Fail to Reject, $p = .05$

of time late for the jobs' time windows. For the on-line optimization approach the amount of lateness increases sharply across the five scenarios – starting at 0 ranging through 8.72, 15.25, 27.02 and ending at 203.53 hours. The agents, however, have a comparatively small amount of lateness across the five scenarios – starting at 0 ranging through .08, .32, .79, and ending at 49.05 in the most extreme scenario. The penalty affiliated with job rejection is, on the other hand, relatively small for both systems until the most extreme case at which point the job rejection penalty of the agents (129.52 hrs) far exceeds that of the on-line optimization (1.36 hrs). Again, we note that all of the jobs rejected (24 in total) in the on-line optimization were rejected due to the inability of the system to find a feasible solution within 90seconds; this represents only .0022% of all jobs examined across all Sd_* instances.

Table 3 summarizes the mean plus/minus the standard error for the total routing costs averaged over the 33 days of data for both the independent and dependent service time uncertainty cases. Also displayed in the table are the results of a paired t-test comparing the mean total cost of the on-line optimization and the agent-based method for each dataset.

These results indicate that the agent approach can deal slightly better with uncertainty regarding the service times than the on-line optimization approach. This result is completely caused by the many late jobs in the on-line optimization approach. The tardiness manifested in the on-line optimization results comes from the assignment process of the approach when used in a rolling-horizon framework. The MIP is calibrated to specify an arrival time at each job in each snapshot problem. If a truck is close to violating the time specified in the previous solution, then the job is permanently fixed and the truck begins execution. In this way many jobs are assigned that later encounter delay during execution as a result of the variable service times. This also explains the many late jobs in the extreme case of 4 hours, where the agent approach incurs a high penalty for rejection, because the agent approach can find no way to squeeze that many jobs into an already delayed schedule.

Furthermore, it is clear that the amount of late time incurred by the on-line optimization solution in the dependent case is higher than that in the independent case. The difference in costs for the agents is however not so different between the independent and dependent service time uncertainty scenarios. In fact, a paired t-test indicates that in all, but the S_{14400} and Sd_{14400} scenarios, there is a statistically significant difference ($p < .0001$) in the means of the independent and dependent service time scenarios for the on-line optimization approach. On the other hand, the agents do not exhibit a statistically significant difference in means (at $p > .01$) for all but the S_{1800} and Sd_{1800} cases.

These results indicate that the on-line optimization tends to suffer in terms of late time when the service time uncertainty has underlying dependencies. This is most likely because jobs that are close together location-wise often end up in the same route. When jobs are on the same route, if one job demonstrates delay in execution, subsequent jobs are immediately assigned as they appear more urgent in the system. From this initial solution many job assignments are then fixed and service is undertaken; only to suffer from even more lateness occurring in execution. From this argumentation, we may expect better performance of the on-line optimization when service times are static. We now turn our attention

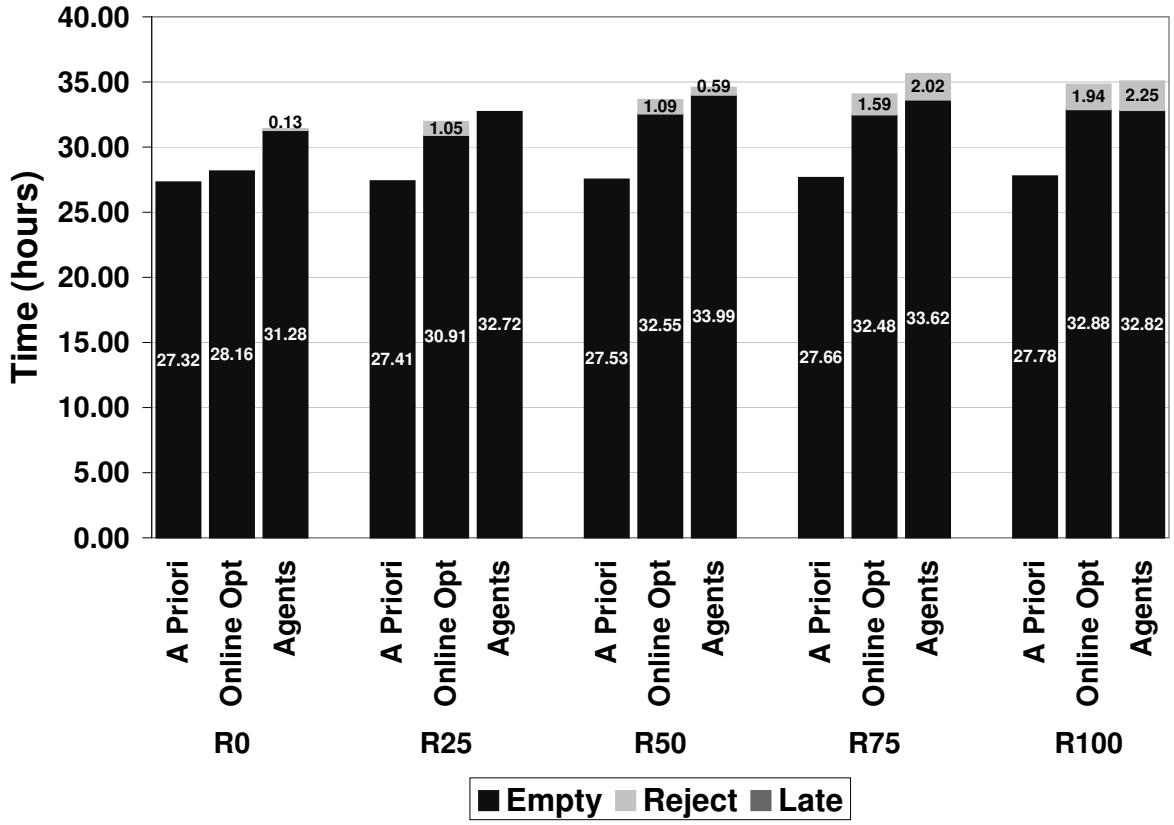

Figure 6: Mean over 33 days of the routing cost components (empty time, rejection penalty, late time) for the agent and on-line optimization solution approaches across five job arrival uncertainty scenarios.

to the results of the job arrival uncertainty experiments to see whether we can support this experimentally.

4.2 Job Arrival Uncertainty Results

Job arrival uncertainty is the most common type of uncertainty classically studied in the dynamic vehicle routing literature. As the instances used to test job arrival uncertainty do not contain jobs that would cause lateness, regardless of their arrival time, it is valid in these instances to compare all three routing systems–the *a priori* optimal (a realistically unattainable lower bound), the on-line optimization approach, and the agent approach.

Figure 6 shows the mean routing costs of the *a priori* optimal, on-line optimization, and agent solution approaches for the case of job arrival uncertainty. Of note is the fact that the empty time (hours) attributable to each system does not vary significantly across the five scenarios, but does show a marked difference when studied between systems. The *a priori* optimal remains the lowest for all three systems with a value between 27 and 28 hours for all scenarios. The slight increase is due to the tightening of the time constraints stemming from the later availability of some jobs. It should also be noted that in scenarios R_0, R_{25}, R_{50}, and R_{75}, CPLEX was unable to find a confirmed optimal solution to the *a priori* problem in three instances before running out of memory. In scenario R_{100} there were four instances for which the optimal could not be found. In these cases the relaxed lower bound was taken as the objective value. On average, the lowest bound was 1.05% from the best known integer solution at the time the model quit running. Thus, the data presented here represents a lowest bound on the routing costs.

Table 4: mean ± std. error of total routing costs (lateness plus rejected penalties plus time spent empty) in hours for all job arrival uncertainty experiments; n = 33.

Dataset	*A Priori* Optimal	On-line Optimization	Agents	Paired t-test: Difference in on-line opt and agent means equal to 0?
R_0	27.32 ± .35	28.16 ± .34	31.41 ± .38	Reject, $p < .0001$
R_{25}	27.41 ± .36	31.96 ± .46	32.72 ± .37	Fail to Reject, $p = .08$
R_{50}	27.53 ± .36	33.64 ± .62	34.58 ± .47	Fail to Reject, $p = .13$
R_{75}	27.66 ± .37	34.07 ± .71	35.64 ± .69	Reject, $p = .01$
R_{100}	27.78 ± .37	34.82 ± .73	35.07 ± .71	Fail to Reject, $p = .61$

The on-line optimization, on the other hand, performs consistently in the range between the *a priori* optimal and the agent system but does experience a steady increase across scenarios–ranging from 28.16 hours when all jobs are known at time zero to 32.88 hours in the case when all jobs are revealed over time. The agents system consistently yields a higher (and in four cases the highest) level of empty time across all five scenarios; ranging from 31.28 hours in the case with the least uncertainty to 32.82 in the case with the most uncertainty. With the exception of the low uncertainty case, both the on-line optimization and the agent system incur a penalty for rejecting jobs. The rejection penalty grows at an increasing rate for the agent system (ranging from 0.13 to 2.25) while it appears to fluctuate in a smaller range for the on-line optimization approach (ranging from 1.05 to 1.94). Interestingly, in the on-line optimization only one job was rejected due to the inability of the system to find a feasible solution within 90seconds.

Overall we can say that the on-line optimization approach consistently outperforms the agent approach. However, the more jobs are uncertain, the smaller the difference becomes–to the point that both systems can be considered to be competitive. This is supported statistically, with a paired t-test indicating that the difference in the means is not significant in the R_{100} case at $p = .61$. A summary of the means plus/minus standard errors, along with the results of the paired t-tests, can be seen in Table 4.

Now that we have examined both service time and job arrival uncertainty in isolation, we turn our attention to a scenario that combines both types of uncertainty.

4.3 Job and Service Time Uncertainty Results

This final set of experiments focuses on the effect of combining job arrival uncertainty at the 50% level with independently distributed service time uncertainty. For this scenario, we are again forced to leave out the results for the *a priori* optimal, because only the on-line optimization and the agents can incur lateness.

Figure 7 shows the mean routing costs of the agent and on-line optimization solution approaches for the case of independent service time variability combined with a 50% job arrival uncertainty level. As in the service time uncertainty cases, the empty time remains relatively constant across all cases for both solution approaches. Again, as in the service time uncertainty scenarios, the primary difference, both across scenarios and between systems, can be seen in terms of the mean over the total amount of time late for the jobs' time windows. For the on-line optimization approach the amount of lateness increases sharply across the five scenarios–starting at 0 ranging through 4.39, 9.07, 13.72 and ending at 144.98 hours. The agents, however, have a comparatively small amount of lateness across the five scenarios–starting at 0 ranging through .12, .42, 1.07, and ending at 57.51 in the most extreme scenario. The penalty affiliated with job rejection is, on the other hand, relatively small for both systems until the most extreme case at which point the job rejection penalty of the agents (132.64 hrs) far exceeds that of the on-line optimization (5.94 hrs). We should note here that only nine of the jobs rejected, in all $R_{50}S_*$ instances of the on-line optimization, were rejected due to the inability of the system to find a feasible solution within 90 seconds.

Overall we can see that agents do better here, and perform similarly to the setting without job arrival uncertainty. However, for the extreme case of a four hour service time uncertainty, the on-line

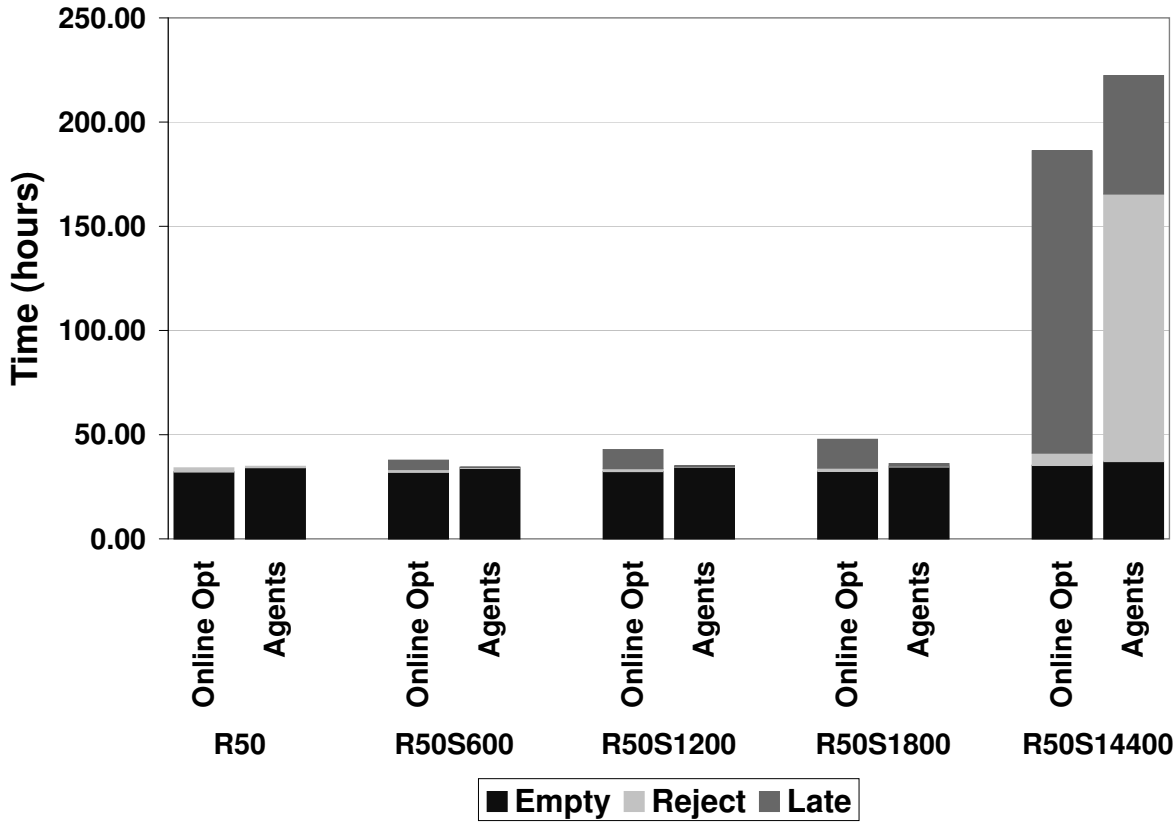

Figure 7: Mean over 33 days of the routing cost components (empty time, rejection penalty, late time) for the agent and on-line optimization solution approaches across five combined job arrival and service time uncertainty scenarios.

Table 5: mean ± std. error of total routing costs (lateness plus rejected penalties plus time spent empty) in hours for all service time combined with job arrival uncertainty experiments; n = 33.

Dataset	On-line Optimization	Agents	Paired t-test: Difference in means equal to 0?
$R_{50}S_{600}$	37.71 ± .75	34.36 ± .44	Reject, $p < .0001$
$R_{50}S_{1200}$	42.79 ± 1.01	35.06 ± .44	Reject, $p < .0001$
$R_{50}S_{1800}$	47.71 ± 1.21	36.04 ± .47	Reject, $p < .0001$
$R_{50}S_{14400}$	186.17 ± 2.94	222.25 ± 4.41	Reject, $p < .0001$

optimization approach has much lower costs. This difference between the agents and on-line optimization is not, however, statistically significant (most likely due to the difference in variance exhibited by both datasets). The means plus/minus standard error as well as the paired t-test results can be seen in Table 5.

What is interesting, however, is that the difference in means between the on-line optimization S_{14400} and $R_{50}S_{14400}$ instances is statistically significant ($p < 0.001$). The moderating effect that job arrival uncertainty seems to have on service time uncertainty is most likely due to the fact that when jobs arrive over time, each problem instance is smaller and can therefore solve to optimality within the 30 second horizon allotted to it. Furthermore, the on-line optimization is less likely to fix long routes early as in the case with only service time uncertainty, since it does not have access to information regarding co-located jobs. Thus, each truck may only be assigned one (or even zero) jobs in the early iterations of the optimization, this helps to keep the trucks from permanently being assigned longer routes of co-located jobs.

5 Discussion

Returning to the question raised in our title – can agents measure up? From the results presented here, we are safe to say: yes, but that is not the end of the story. Not only do the agents perform competitively in comparison to the on-line optimization approach on the spatially realistic drayage case, but they also demonstrate a certain number of strengths in handling uncertainty.

These strengths are most obvious in the experiments focused on service time uncertainty. In both independent (S_*) and dependent (Sd_*) cases the agents consistently yield lower routing costs when the variation in service times is greater than 10 minutes. We attribute this advantage to the limitations of the heuristics agents use to compute a solution. These heuristics provide sub-optimal results when faced with a static problem. In a series of problems however, being suboptimal in the beginning means having more trucks on the move, yielding more options to accommodate changes.

There are, on the other hand, settings in which the on-line optimization approach gives significantly better results. In particular, in settings where job arrival uncertainty is a dominant feature. Admittedly, this is not surprising, as job arrival uncertainty is the type of uncertainty most commonly studied in the literature and the type of uncertainty the on-line optimization approach was truly calibrated for.

Given these findings, one might ask the question: "how much are these differences attributable to the centralized versus decentralized nature of the systems as opposed to the optimization versus heuristic characteristics of the approaches?" We begin to answer this question by noting that the comparison between the two approaches is a comparison of heuristics. While the on-line optimization has the opportunity to find an optimal solution, it more often finds only a good feasible solution within each 30-second interval. As such, neither system can claim to produce an optimal solution. Nevertheless, we cannot, so easily, disentangle the centralization/decentralization and heuristic/optimization facets of each approach. The near optimal solutions put forth by the on-line optimization are highly dependent on the centralization of all problem data; the agility exhibited by the decentralized agents is dependent on fast running heuristics.

As the on-line optimization operates by batching all system information at 30-second intervals, it is also forced to batch routing decisions in the same intervals. This process of centralizing information,

and using all this information in making job assignment as well as rejection decisions yields assignments that may later turn out to be late, because service times may vary. On the other hand, when service times are static, this approach yields assignments that very effectively balance empty distance against job rejection.

In contrast, the agents, simulated in an individualized manner, are not dependent on batched decisions made every 30-seconds. They can react at any time, but they have a limited view. The individualized outlook implies that the truck agents do not recognize the cost of a rejected container. The container agents, in turn, are the victims of the trucks' inefficiency (or service time variability). This yields a high level of rejected jobs, but a minimal level of lateness when service time uncertainty is high. However, when job arrival uncertainty is the only source of uncertainty, the agents lack the ability to globally weigh routing costs against rejection penalties.

The advantage of a global perspective, regarding the ability to balance empty distance against rejection costs, is supported by our results. Paired t-tests, comparing only the combined costs of empty distance and rejection penalties (leaving out lateness), reveal a significant difference in these mean costs for all service time uncertainty scenarios; and for all, but the R_{100}, job arrival uncertainty scenarios. Alternately, the combined service time and job arrival uncertainty scenarios show a significant difference only in the most uncertain case ($R_{50}S_{14400}$). These results lead us to the following conclusion, when the on-line optimization approach performed better, it was by capitalizing on the optimal (or near optimal) balance between routing and rejection costs. In cases the agents performed better, their flexibility provided by their distributed nature was the competitive advantage.

Aside from the observable and quantifiable results presented in this comparative study, we would be remiss if we did not highlight a few qualitative differences between agent-based and on-line optimization approaches for vehicle routing. Agents permit modeling a complex system in a rather "ergonomic" way. That is in a way that closely mirrors the problem being solved. This feature of agent-based systems can make them especially appealing to the lay-person. Thus, when we consider the real-world environment of a drayage company, it is easy to imagine a dispatcher embracing a solution approach that directly maps onto their existing operations. Additionally, in a competitive environment, where sharing of sensitive information is an issue, the distributed agent model may be advantageous.

On the other hand, agent technology is new. New technology often represents a risky choice. In contrast, many existing decision support systems depend on optimization techniques. Furthermore, there is a perception that optimization and operations research represents a technique with years of scientific research underpinning it. Additionally, the management level of logistics companies often prefers to specify their goals for the company as encapsulated in an objective function. The notion of emergent behavior or autonomous fulfillment of local goals may appear threatening to those that must select and invest in a new decision support system.

Given both the quantitative and qualitative advantages and disadvantages of both approaches, the future of this research lays in the investigation of how ideas from an agent-based approach can be used in an on-line approach, and vice versa. We expect that a hybrid approach with the ability to exhibit the right behavior depending on the current situation is ultimately the best way of dealing with uncertainty. Furthermore, the agent-based approach can itself be augmented to include learning. In this paper we describe agents that do not make any effort to adjust their behavior based on what they have seen before. One may conjecture that some of the global state could be inferred from past experiences, and that agents with limited information could benefit from learning. An interesting challenge, for future research, would be to find the conditions in which agents can learn and adapt to uncertainties in comparison to a centralized system based on stochastic programming.

In order to find the exact conditions under which one approach outperforms the other, and to understand the fundamental reasons for that behavior, further experiments are required. Not only should we study more general vehicle routing problems (using standard datasets such as the Solomon benchmark sets (Solomon, 1987)), we should also study other sources of uncertainty, such as travel time uncertainty (to see if this indeed shows similar behavior to the dependent service times considered in this paper) and truck break downs. Furthermore, as instances grow larger, the effect of limited time and/or memory may influence performance drastically, and should thus be carefully taken into consideration. Finally,

we would like to see a similar study focused on a comparison of agents with stochastic approaches. We believe the set-up of the experiments in this paper, as well as the results presented, support performing these future experiments.

Acknowledgements

We are indebted to three anonymous referees for their valuable comments. Mathijs de Weerdt is supported by the Technology Foundation STW, applied science division of NWO, and the Ministry of Economic Affairs of the Netherlands.

References

Berbeglia, G., Cordeau, J.-F., Gribkovskaia, I., and Laporte, G. (2007). Static pickup and delivery problems: a classification scheme and survey. *Top*, 15(1):1–31.

Breedam, A. V. (1994). *An Analysis of the Behavior of Heuristics for the Vehicle Routing Problem for a Selection of Problems with Vehicle-Related, Customer-Related, and Time-Related Constraints.* PhD thesis, University of Antwerp.

BTS (2007). Transportation statistics annual report 2007.

Bürckert, H.-J., Fischer, K., and Vierke, G. (2000). Holonic transport scheduling with Teletruck. *Applied Artificial Intelligence*, 14(7):697–725.

Chen, Z.-L. and Xu, H. (2006). Dynamic column generation for dynamic vehicle routing with time windows. *Transportation Science*, 40(1):74–88.

Cordeau, J.-F., Desaulniers, G., Desrosiers, J., Solomon, M. M., and Soumis, F. (2001). VRP with time windows. In *The vehicle routing problem*, pages 157–193. Society for Industrial and Applied Mathematics.

Cordeau, J.-F., Gendreau, M., Laporte, G., Potvin, J.-Y., and Semet, F. (2002). A guide to vehicle routing heuristics. *Journal of the Operational Research Society*, 53(5):512–522.

Cordeau, J.-F., Laporte, G., and Ropke, S. (2008). Recent models and algorithms for one-to-one pickup and delivery problems. In Golden, B. L., Raghavan, S., and Wasil, E. A., editors, *Vehicle Routing: Latest Advances and Challenges*, pages 327–357. Kluwer, Boston.

Das, R., Kephart, J. O., Lefurgy, C., Tesauro, G., Levine, D. W., and Chan, H. (2008). Autonomic multi-agent management of power and performance in data centers. In *Proceedings of the 7th international joint conference on Autonomous Agents and Multiagent Systems (AAMAS 2008)*, pages 107–114, Richland, SC. IFAAMAS.

Davidsson, P., Henesey, L., Ramstedt, L., Törnquist, J., and Wernstedt, F. (2005). An analysis of agent-based aproaches to transport logistics. *Transportation Research Part C*, 13(4):255–271.

Dorer, K. and Calisti, M. (2005). An adaptive solution to dynamic transport optimization. In *Proceedings of the 4th International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2005)*, pages 45–51. ACM Press.

Fischer, K., Muller, J. P., Pischel, M., and Schier, D. (1995). A model for cooperative transportation scheduling. In *Proceedings of the First International Conference on Multiagent Systems.*, pages 109–116, Menlo park, California. AAAI Press / MIT Press.

Ghiani, G., Guerriero, F., Laporte, G., and Musmanno, R. (2003). Real-time vehicle routing: Solution concepts, algorithms and parallel computing strategies. *European Journal of Operational Research*, 151(1):1–11.

Gribkovskaia, I. and Laporte, G. (2008). One-to-many-to-one single vehicle pickup and delivery problems. In Golden, B. L., Raghavan, S., and Wasil, E. A., editors, *Vehicle Routing: Latest Advances and Challenges*, pages 359–377. Kluwer, Boston.

Herroelen, W. and Leus, R. (2005). Project scheduling under uncertainty: Survey and research potentials. *European Journal of Operational Research*, 165(2):289–306.

Hoen, P. J. and Poutré, J. A. L. (2003). A decommitment strategy in a competitive multi-agent transportation setting. In *Proceedings of 2nd International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2003)*, pages 1010–1011. ACM Press.

Hutzschenreuter, A. K., Bosman, P. A. N., Blonk-Altena, I., van Aarle, J., and Poutré, H. L. (2008). Agent-based patient admission scheduling in hospitals. In *Proceedings of the 7th international joint conference on Autonomous Agents and Multiagent Systems (AAMAS 2008)*, pages 45–52. IFAAMAS.

ILOG, Inc. (1992). Using the CPLEX Callable Library and CPLEX Mixed Integer Library.

Jaillet, P. and Wagner, M. R. (2006). Online routing problems: Value of advanced information as improved competitive ratios. *Transportation Science*, 40(2):200–210.

Jakob, M., Pěchouček, M., Miles, S., and Luck, M. (2008). Case studies for contract-based systems. In *Proceedings of the 7th international joint conference on Autonomous Agents and Multiagent Systems (AAMAS 2008)*, pages 55–62. IFAAMAS.

Kinderwater, G. A. P. and Savelsbergh, M. W. P. (1997). Vehicle routing: Handling edge exchanges. In Aarts, E. and Lenstra, J., editors, *Local Search in Combinatorial Optimization*, page 337–360. Wiley, Chichester, UK.

Kohout, R. and Erol, K. (1999). In-time agent-based vehicle routing with a stochastic improvement heuristic. In *Proceedings of the 16th national conference on Artificial Intelligence and the 11th on Innovative Applications of Artificial Intelligence (AAAI/IAAI 1999)*, pages 864–869, Menlo Park, CA, USA. AAAI press.

Laporte, G., Gendreau, M., Potvin, J.-Y., and Semet, F. (2000). Classical and modern heuristics for the vehicle routing problem. *International Transactions in Operations Research*, 7(4-5):285–300.

Leong, H. W. and Liu, M. (2006). A multi-agent algorithm for vehicle routing problem with time window. In *Proceedings of the ACM Symposium on Applied Computing (SAC 2006)*, pages 106–111, New York, NY, USA. ACM Press.

Mahmassani, H. S., Kim, Y.-J., and Jaillet, P. (2000). Local optimization approaches to solve dynamic commercial fleet management problems. *Transportation Research Record*, 1733:71–79.

Mahr, T., Srour, J., de Weerdt, M., and Zuidwijk, R. (2008). Agent performance in vehicle routing when the only thing certain is uncertainty. In *Proceedings of the workshop on Agents in Traffic and Transportation (ATT)*.

Mes, M., van der Heijden, M., and van Harten, A. (2007). Comparison of agent-based scheduling to look-ahead heuristics for real-time transportation problems. *European Journal of Operational Research*, 181(1):59–75.

Morlok, E. and Spasovic, L. (1994). Redesigning rail-truck intermodal drayage operations for enhanced service and cost performance. *Journal of Transportation Research Forum*, 34(1):16–31.

Parragh, S., Doerner, K., and Hartl, R. (June 2008a). A survey on pickup and delivery problems: Part I: Transportation between customers and depot. *Journal für Betriebswirtschaft*, 58(2):21–51.

Parragh, S., Doerner, K., and Hartl, R. (June 2008b). A survey on pickup and delivery problems: Part II: Transportation between pickup and delivery locations. *Journal für Betriebswirtschaft*, 58(2):81–117.

Persson, J. A., Davidsson, P., Johansson, S. J., and Wernstedt, F. (2005). Combining agent-based approaches and classical optimization techniques. In *Proceedings of the European workshop on Multi-Agent Systems (EUMAS 2005)*, pages 260–269.

Perugini, D., Lambert, D., Sterling, L., and Pearce, A. (2003). A distributed agent approach to global transportation scheduling. In *IEEE/WIC Int. Conf. on Intelligent Agent Technology (IAT 2003)*, pages 18–24.

Powell, W. B., Jaillet, P., and Odoni, A. (1995). Stochastic and dynamic networks and routing. In Ball, M. O., Magnanti, T., Monna, C., and Nemhauser, G., editors, *Network Routing*, volume 8 of *Handbooks in Operations Research and Management Science*, pages 141–295. North-Holland, Amsterdam.

Powell, W. B., Towns, M. T., and Marar, A. (2000). On the value of optimal myopic solutions for dynamic routing and scheduling problems in the presence of user noncompliance. *Transportation Science*, 34(1):67–85.

Psaraftis, H. N. (1988). Dynamic vehicle routing problems. In Golden, B. L. and Assad, A. A., editors, *Vehicle Routing: Methods and Studies*, volume 16 of *Studies in Management Science and Systems*, pages 223–248. Elsevier Science Publishers, Amsterdam.

Regan, A. C., Mahmassani, H. S., and Jaillet, P. (1995). Improving the efficiency of commercial vehicle operations using real-time information: potential uses and assignment strategies. *Transportation Research Record*, 1493:188–198.

Regan, A. C., Mahmassani, H. S., and Jaillet, P. (1996). Dynamic decision making for commercial fleet operations using real-time information. *Transportation Research Record*, 1537:91–97.

Rehak, M., Pechoucek, M., Celeda, P., Novotny, J., and Minarik, P. (2008). CAMNEP: agent-based network intrusion detection system. In *Proceedings of the 7th international joint conference on Autonomous Agents and Multiagent Systems (AAMAS 2008)*, pages 133–136, Richland, SC. IFAAMAS.

Sandholm, T. W. and Lesser, V. R. (2001). Leveled commitment contracts and strategic breach. *Games and Economic Behaviour*, 35(1-2):212–270.

Scerri, P., Gonten, T. V., Fudge, G., Owens, S., and Sycara, K. (2008). Transitioning multiagent technology to uav applications. In *Proceedings of the 7th international joint conference on Autonomous Agents and Multiagent Systems (AAMAS 2008)*, pages 89–96, Richland, SC. IFAAMAS.

Schillo, M., Kray, C., and Fischer, K. (2002). The eager bidder problem: a fundamental problem of DAI and selected solutions. In *Proceedings of 1st International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2002)*, pages 599–606, New York, NY, USA. ACM Press.

Solomon, M. M. (1987). Algorithms for the vehicle routing and scheduling problems with time window constraints. *Operations Research*, 35(2):254–265.

Thompson, P. and Psaraftis, H. (1993). Cyclic transfer algorithms for multivehicle routing and scheduling problems. *Operations research*, 41(5):935–946.

Wooldridge, M. (2002). *An Introduction to MultiAgent Systems*. John Wiley & Sons, Chichester, England.

Yang, J., Jaillet, P., and Mahmassani, H. (1999). On-line algorithms for truck fleet assignment and scheduling under real-time information. *Transportation Research Record*, 1667:107–113.

Yang, J., Jaillet, P., and Mahmassani, H. (2004). Real-time multi-vehicle truckload pick-up and delivery problems. *Transportation Science*, 38(2):135–148.