ISSN:2231-0711
www.ijcset.net

Probing Agent Based Anomaly Solving Strategies for Dynamic Distributed Scheduling in Railway Transportation

M.Hema Madhuri[#1], Divya Vadlamudi[#2], Movva N.V Kiran Babu[#3]Bollimuntha.Kishore Babu[#4]
[#1,#4]M .Tech student, Department of CSE, K L University, Andhrapradesh.
[#2]Assistant professor, Department of CSE, K L University, Andhra Pradesh
[#3]Associate Professor, Department of CSE, Mother Theresa Institute of Science &Technology, Sathupally, AP

[#1]metlamadhuri@gmail.com,[#2]divyaonlineid@yahoo.co.in,
[#3]kiranbabuonline@yahoo.co.in,[#4]bkishore002@gmail.com

***Abstract*: The agent computing paradigm is rapidly emerging as one of the powerful technologies for the development of large scale distributed systems to deal with the uncertainty in a dynamic environment. Railway dispatching or scheduling has been usually modeled using classical technologies, such as operations research and constraint programming. But these strategies are suitable to provide static solutions where the information is complete, but they are not suitable for dynamic environment for distributed scheduling. In order to cope up with dynamic environment, several problem solving strategies have been proposed based on agent technology. In this paper, we first investigate the problems in agent based systems for railway transportation. Finally we explore different agent based problem solving strategies for dynamic scheduling in railway transportation.**

***Key words*-agent, agent technology,mobileagent, net manager**

I.INTRODUCTION

Agent can be defined to be independent, problem solving computational entities capable of effective operation in dynamic and open environment[1] .agents are often deployed in environment in which they interact, and work together with other agent including people and software they have possibly contradictory aims such environments are known as the multi agent systems .agents can operate without the direct involvement of humans and others.[2] Agent can be used as a design symbol for designers and developers in the way of structuring an application around autonomous, communicative elements, and elements, and lead to the constructing of software tools and infrastructure to support the design symbol. In this sense, they offer a new and often more appropriate route to the development of complex systems, especially in open and changing environments. Agent technologies span a range of specific techniques and algorithm for dealing with interactions with others in changing and open environment. Agent-based systems are one of the most effective and important areas in research and development to have emerged in information technology in the 1990s.[3]an agent is a computer system that is capable of flexible autonomous action in dynamic, unpredictable, typically multiage domains the characteristics of dynamic and open environments in which, for example, heterogeneous systems must interact, span organizational boundaries, and operate effectively within rapidly changing circumstances and with dramatically increasing quantities of available information, suggest that improvements on traditional computing models and paradigms are required .thus the need for some degree of autonomy, to allow components to respond dynamically to changing circumstances while trying to achieve over-arching objectives, is seen by many as fundamental. Many observers therefore believe that an agent represents the most important new paradigm for software development since object orientation. The concept of an agent has found wide range of sub-disciplines of information technology, including computer networks, software engineering, artificial intelligence, human-computer interaction, distributed and concurrent systems, mobile systems, telematics, computer-supported cooperative work, control systems, decision support, information retrieval and management, and electronic commerce. In practical developments, web services, for example, now offer fundamentally new ways of doing business through a set of standardized tools, and support a service-oriented view of different and independent software components interacting to provide valuable functionality. In the context of such developments, agent technologies have gradually come to the foreground. Because of its horizontal nature, it is likely that the successful adoption of agent technology will have a profound, long-term impact both on the competiveness and capability of IT industries, and on the way in which future computer systems will be conceptualized and implemented. Many researchers and programmers see agents as programs roaming a network to collect business-related data in order help users to buy goods, or implement platform independent code-on-demand, for example this need for

mobile agents is acknowledged, and builds on European strengths, but mobility brings added security problems. The research effort concentrates on how to guarantee termination, security or exactly-once protocols. To protect against malicious hosts, agents should contain time limit validity, and electronic money with an expiration date. A key issue that needs to be addressed here is administrability of mobile agent systems, e.g., authorization policies; this has been a major reason why mobile agents have not yet been taken up by the mainstream. Note also that hosts need to be protected as well as agents End users already encounter the situation that, while ample bandwidth is available on the backbones of network service providers, their experience is limited by the constraints of the infamous last mile. Mobile agents may improve the end user experience by offloading application-specific filtering, media adaptation, and other pre-processing to a node with high bandwidth connectivity. This is particularly interesting for mobile phones and portable devices. One of the commercial application areas in which the added value of mobile agents is very high, is large-scale distributed or decentralized system integration with highly adaptive and dynamic business logic. Existing solutions are generally centralized, pulling everything onto one platform, limiting the complexity and changes that can be handled. A decentralized agent approach divides and conquers complexity by pushing a large part of the business logic out onto source systems so that much monitoring and aggregation can be done on each. This distributes workload and increases robustness because the local processing can be performed independently of other systems, resulting in fewer and more relevant interactions with these systems, at a higher level of abstraction. In turn, mobility, mainly single hop, is the answer to the increasing need for flexibility and adaptability in business logic. Agents can easily be deployed to source systems, carrying new database drivers, code to interact with new application or file types, or new data processing rules. Software is updated at the component-level, at runtime, proving a level of dynamism and flexibility that goes far beyond current release policies. Agent communication and behavior capabilities complete the picture, being very well suited to high-level service-based Interactions, the decentralized implementation of business logic, and for adapting and handling change in their environment. A nice property of the dynamic, component-level approach is that it naturally fits step-by-step system integration, with each step resulting in added value for the business. This is a particularly significant advantage in the current economic climate, in which many companies have seen mega-projects, fail. For example, Global IDs Inc in the US offers a next-generation product suite for data integration based on the Tryllian mobile agent platform. Their data integration products are capable of simultaneously monitoring many hundreds of enterprise systems for relevant changes in data or metadata, by deploying mobile agents onto those systems. The agents tap into local databases or applications, keep track of changes, can pre-process data and only forward relevant events or structured derived data to centralized collectors – in real time if required. The mobility of the agents allows highly customized functionality, which can be dynamically updated. Thus, the business user can change the business rules that are being executed at any point in time, while only relevant drivers and adapters are transferred to a source system. Agents can assess the impact of changes in the business rules and handle that impact throughout the integration process in this Paper we see an agent-based approach and its applications in railway transportation.

II.AGENT-BASED SYSTEMS FOR RAILWAY TRANSPORTATION

Efficient transportation is it of persons or goods is a key issue in today's industrial world. Because of the immense amount of transportation tasks, it is necessary to use the available resources most effectively. Thus, computer aided or entirely controlled scheduling systems are key technologies in the forthcoming century. Due to the constantly increasing demand of short-term train schedules, automatic tools for decision support that can be used by Transport Operators in the management of freight trains traffic are more and more necessary. [4] This is particularly true when the number of trains running on a railway line and the availability of tracks are only known on a day by- day basis. Decision support systems can prove very useful to maximize tracks demand granting and optimize the traffic flow. Railway dispatching or scheduling has been usually modeled using classical technologies, such as operations research and constraint programming. These technologies are suitable to model static situations where the information is complete, but they lack the ability to cope with the dynamics and uncertainty of freight-train-traffic management. The complexity of dispatching and scheduling problems in the transportation domain has attracted researchers from the multiagent community [5] an MAS system that models a society of transportation companies whose goal is to deliver a set of dynamically given orders satisfying the given cost and time constraint. The distributed scheduling is achieved by giving individual vehicles local planning capability, and the global scheduling plans are generated from local decisions and problem-solving strategies. Railroad freight haulage as it is performed today is depicted in Figure 1: a company that wants to ship something via railroad to its customers delivers the freight to the local freight center (usually a railroad station) where it is stored until enough freight from other companies has arrived to justify a train to the regional freight center. At the regional freight center, containers from other local freight centers that have the same direction are assembled and sent to the next interregional freight center where another re-assemblance process takes place. The decomposition of the trains is achieved in reverse order [6].

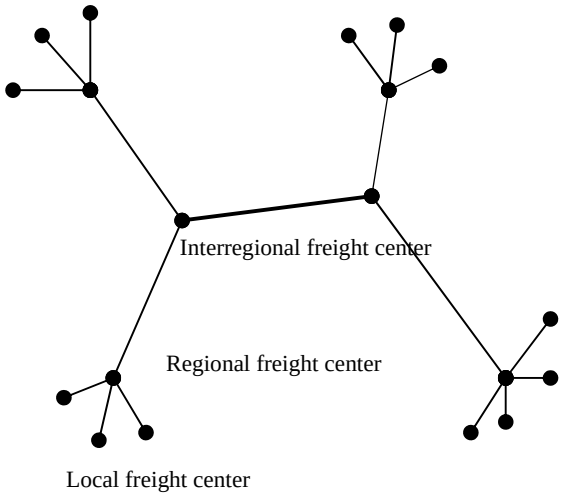

Figure 1: Hierarchical freight haulage

An alternative approach [7] to the classical freight haulage process uses small railroad *transportation modules* (or simply *modules*, e.g. [8] instead of complete trains. Whereas a normal train is made up of one railcar and several freight cars, a transportation module is a unit of an engine and a storage area with approximately the size of an individual freight car. When a company wants to deliver some freight to a customer, it orders a transportation module at a local freight center and loads its goods on this module. The module itself is then responsible to find its way through the railroad network. The problems now, that a location route in a railroad network cannot be used by two independent modules at the same time, i.e. either a route is blocked by a single module or two (or more)modules share a route by hooking together at the beginning of a location route and splitting up afterwards. In order to use the underlying railroad infrastructure most efficiently, the railroad modules should share as many location routes as possible. The main advantage of this approaches that it avoids a central planning authority that schedules all transportation modules. Instead, each module is responsible to achieve its goal which is to deliver its freight to some destination node in the network. Additionally, each module performs some local optimization of the network throughput by sharing as many location routes with other modules as possible. The local optimization process of all modules eventually leads to a high, though usually not optimal, degree of resource efficiency. Besides this major advantage, a decentralized approach implies less coupling operations during the train composition process, a high degree of customer accessibility and lower costs because of the effective location route usage the coordination of the local optimization processes of many thousand modules in a practical scenario is a computationally demanding process and requires a sophisticated algorithmic framework. The major requirements towards a real-world system are listed in the following paragraphs. First of all, a scheduling system should be capable of dealing with an incomplete problem specification. The classical operations research based approaches assume that the entire problem specification is given at the systems start-up time. Unfortunately, this assumption often does not hold in the real world! Transportation companies usually receive customer tasks over time and not only at the beginning of the planning process. Thus, the company cannot wait until all task specifications are available, then start the planning process in order to find an optimal plan and finally start to execute this plan. Instead, the company must overlap the planning process based on the available data and plan execution monitoring. Incoming tasks must then be integrated in the ongoing planning and execution process. The second requirement is highly related to the first point and deals with the problem to establish a proper relation between the system and the real world it is supposed to model. A prominent example for the gap between research and reality are order dispatching systems for haulage companies: usually, the respective systems try to compute optimal routes and schedules for the companies trucks, but they fail to monitor the plan execution process and are thus not able to react to unforeseen situations such as mechanical failure of trucks or traffic jams making it impossible to maintain the original schedule. An exception from this shortcoming is the TELETRUCK system presented in [9] which uses a multi-agent approach for planning and monitoring of transportation tasks. Finally, a system should not necessarily try to find an optimal solution for a given problem. Although optimality is a desirable property of a solution, it is often the case that the computational effort to find an optimal solution is too high for realistic problems. Thus, a mechanism that is capable of finding a rather good solution quickly and then improving this solution if sufficient time is available, is an alternative to classical approaches. Algorithms of this type are usually referred to as *anytime-algorithms*. The name stems from the fact, that these algorithms can be aborted at any time and still yields a solution. The quality of the solution simply depends on the resources assigned to the algorithm.

III PROBLEM DISCRIPTION

The overall goal of the system presented in this paper is to reduce the cost for a given set of transportation tasks in a railroad network. Each task is specified as a tuple consisting of the origin and the destination node, the earliest possible departure time (EDT), the latest allowed arrival time (LAT) and an additional time stamp indicating when the task is announced to the system. Thus, the set of tasks is not fully known to the system at start-up time, new tasks arrive during the planning process and may require a revision of the already assembled plan in order to reduce cost. A typical time profile for incoming tasks is depicted in Figure 2: at start-up time, 2000 tasks are known to the system; during the next 24 hours (=1440 minutes) additional tasks arrive, summing up to a total of 5000 tasks at the end of the day.

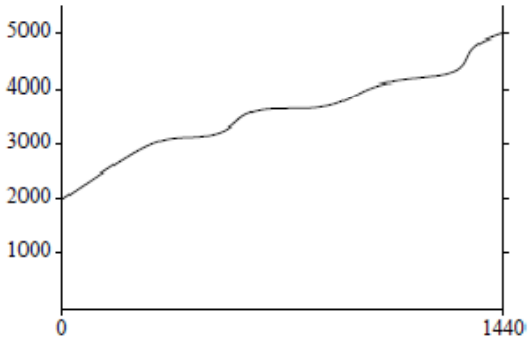

Figure 2: Task arrival time profile

Each task can be served by a single module, i.e. there is no need to hook two or more modules together to serve a single task. Vice versa, we assume also that a module cannot serve more than one task at a time. All tasks occurring in the system are transportation requests in a railroad network; in the current version of the system, we use an abstracted map of the German railroad network with approximately 250 nodes and 350 links. The net consists of several nodes connected via so-called *location routes*. The numbers on the routes in Figure 3 indicate the distance between two nodes connected via a location route.

Whenever a module serves a transportation task, it computes the path from the origin to the destination node with a shortest path algorithm. The module then rents the intermediate location routes for a certain time window from the *net manager*. The time window for each location route is uniquely determined by the earliest departure time and the latest arrival time of the transportation task. When a location route is allocated by a certain module, the route is blocked for other modules during this time interval. In order to reduce route blocking, however, two or more modules can decide to share a particular location route.

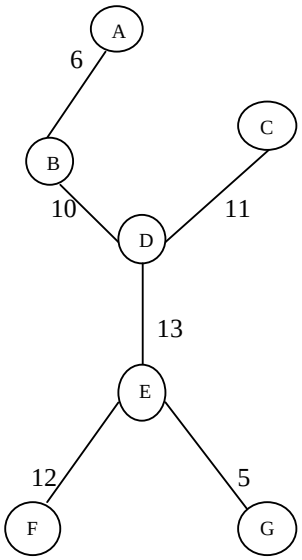

Figure 3: An example railroad network

4) Agent-Oriented Problem Solving

Agent oriented problem solving is a programming paradigm based on autonomous entities – the *agents*. Besides their autonomy, agents are supposed to react to changes in their environment and to be capable of planning their actions in order to achieve their goals. The field of multi-agent systems considers agent based systems with more than one agent. In these systems, an additional agent capability gains importance – the ability to communicate and cooperate with other agents in the system. Multi-agent systems are particularly well-suited for the scenario described in the previous section because they allow a very natural mapping from the entities occurring in the scenario to agents. At first glance, it seems very natural to model the transportation modules introduced in the previous section as agents that pursue their local goals. Doing this, however, results in a conceptual break when it comes to modeling the coupling activities that are necessary for location route sharing. Coupling two or more modules together to form a union requires an election process in order to decide which module should represent the resulting union towards the other unions. Additionally, this approach implies a high degree of intra-union communication whenever the union leader wants to integrate a new module in the existing union. To avoid these problems, we have decided to model the unions as the agents in our system. Applying this scheme results in an important simplification of the system design and the resulting implementation. Merging several unions into a single union does no longer require an election a coordination process among the participating modules as they are straightforwardly integrated in another existing union. The roles of the participating unions (either *master* or *slave*) are determined by the negotiation protocol. Whenever a new task is announced to the system, a new union, consisting only of a single module, is created; we will sometimes refer to unions with only one module as *degenerated* unions. The advantage of applying this scheme is that we do not have to differentiate between modules and unions; every active entity in the system is a union and that's it!

Any agent cooperation within a multi-agent system is based on a negotiation process during which the agents try to figure out a deal that results in mutual benefits for them. The negotiation process amongst several agents is steered by so-called *cooperation protocols*. These protocols tell the individual agents what messages to expect from the peer agents or what messages to send to them, respectively. The two protocols used in our system are the *contract-net* [10] protocol described in Section 4.1 and the *simulated trading* [11] protocol explained in Section 4.2 We have combined these protocols in our scheduling approach to achieve the aforementioned properties (incrementality and anytime) in the following way: an initial solution for the module schedule is obtained by running the contract-net protocol whenever a new task is announced to the system. New tasks are incrementally integrated in the existing scheduling which

guarantees that always a solution for the problem (as far as it is known to the system) exists. However, this solution may be (and usually is) not optimal. In order to improve the quality of the existing solution, the simulated trading protocol is run on the set of tasks (or the respective modules) currently known to the system. Unfortunately, executing the simulated trading protocol is a computationally expensive operation and so it is executed only periodically – either after a fixed number of new tasks has been added to the existing solution or explicitly triggered by a user request. In the following sections, we present the contract-net and simulated trading protocols in detail.

A. Contract-Net Protocol

In the context of the contract-net protocol (Smith, 1980) depicted in Figure 4**,** there are two types of participants: one *manager* and a group of *bidders*. The protocol is initiated by the manager which sends a description of the task under consideration to the bidders. Note, that "task" is a not transportation task mentioned earlier but rather some abstract description of a problem to be solved. We will present the instantiation of the general protocol to our scenario later. After the bidders have received the task description, each of them computes a bid that informs the manager about costs that will be charged if the task is assigned to that particular bidder. After all bidders have submitted their bids to the manager, the manager selects the bid that minimizes his cost and assigns the task to the respective bidder (+) and rejects the offers of the other bidders (-). In our system, this protocol is adopted by creating a new (degenerated) union when a new task is announced to the system. The module in the union plans its path and time constraints for the task and then the parent union initiates the contract-net protocol as the manager and offers the modules plan to the other currently active unions. These unions check if they contain one or more modules that are potential sharing peers and if this is the case, they offer a sharing commitment

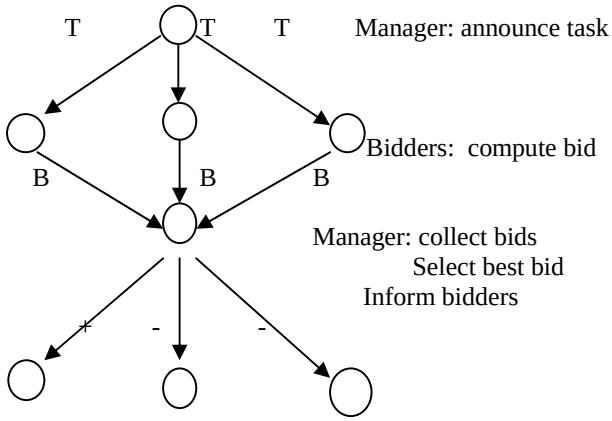

Figure 4: ContractNet Protocol

To the new union. The new union collects these offers and selects the one that has the largest cost saving potential. It then transfers the module to the winning union and ceases to exist because it does not contain other modules. If no union offers a sharing commitment, the new union remains active as degenerated union

B.Simulated Trading

The simulated trading protocol is an algorithm designed to improve existing solutions, not to construct new solutions from scratch. In our case, the input and the output of the protocol are valid schedules where the cost of the output are always less or equal to the cost of the input. This is trivially true since the output can always be the input if no cheaper schedule exists. However, this property is nonetheless important because it guarantees that the protocol can be aborted at any time and still yield a valid solution. Furthermore, if the protocol is given enough computation time, it is guaranteed to find the optimal solution. Now, how does this work the protocol is initiated by a special agent, the

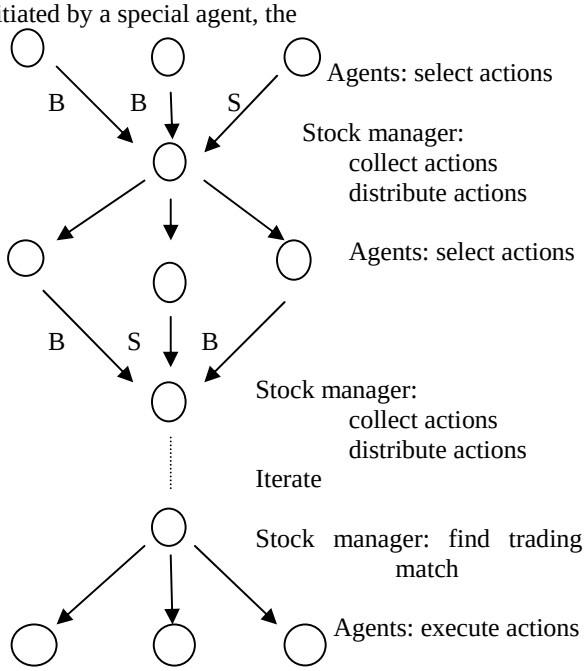

Figure 5: Simulated Trading

Stock manager. In the course of the protocol execution, the agents (here called *traders*) perform several rounds of hypothetical trading, i.e. the traders either choose to sell some of their goods or to buy something from others. In our context a sell operation corresponds to removing a module from a union and a buy operation corresponds to integrating a module in a union. Thus, the unions try to optimize their cost by exchanging unprofitable modules with better ones. The decision which module to sell depends on a probability distribution induced by the potential cost reduction if the module was sold. Vice versa, the decision to buy a module offered by another union depends on the potential cost reduction if the

module would be integrated in the union. After the stock manager has collected the hypothetical sell and buy actions, it must find a valid *trading match* in the set of actions. There a several validity requirements for a trading match e. g. there must be no two buy actions on the same sell operation, etc. Finding a trading match is a nontrivial task and accounts for the computational complexity of the simulated trading protocol If a trading match is found, the stock manager informs the traders which actions must be executed, i.e. which modules must be exchanged. In this section, we have outlined some basic ideas of agent-oriented problem solving. In the next section, we will present the local planning algorithm of the unions. The plan integration operator developed in there enables a union to find a module schedule with a maximum number of location route sharing.

.

CONCLUSION:

Software agents and their applications in railway transportation systems have been proposed for over one decade to solve he dynamic scheduling problems. A number of agent-based systems have already been reported in the literature. In this paper we discussed, different agent based systems for railway transportation and then in the end we explored different agent problem solving strategies for dynamic scheduling. Finally we conclude that the research results obviously show the probable usage of agent technology to improve the performance of railway transportation systems.

ACKNOWLEDGEMENTS

We like to express our gratitude to all those who gave us the possibility to carry out the paper. We would like to thank Mr.K.Satyanarayana, chancellor of K.L.University, Dr.K.Raja Sekhara Rao, Dean, and K.L.University for stimulating suggestions and encouragement. We have further more to thank Prof.S.Venkateswarlu, Dr.K.Subrahmanyam, who encouraged us to go ahead with this paper

REFERENCES

[1] Agent Technology: Enabling Next Generation Computing: A Roadmap for Agent-Based Computing by Michael Luck, Peter McBurney and Chris Priest January 2003, Version 1.0, ISBN 0854 327886

[2] Bo Chen, *Member, IEEE*, and Harry H. Cheng, *Senior Member, IEEE,"* A Review of the Applications of Agent Technology in Traffic and Transportation Systems" IEEE TRANSACTIONS ON INTELLIGENT TRANSPORTATION SYSTEMS, VOL. 11, NO. 2, JUNE 2010.

[3] Agent Technology: Computing as Interaction: A Roadmap for Agent Based Computing by Michael Luck, Peter McBurney, Onn Shehory, Steve Willmott and the AgentLink Community, © September 2005, AgentLink III ISBN 085432 845 9

[4] A. Cuppari, P. L. Guida, M. Martelli, V. Mascardi, and F. Zini, "Prototyping freight trains traffic management using multi-agent systems," in *Proc. Int. Conf. Inf. Intell. Syst.*, Los Alamitos, CA, 1999, pp. 646–653

[5] K. Fischer, N. Kuhn, H. J. Muller, J. P. Muller, and M. Pischel, "Sophisticated and distributed: The transportation domain," in *Proc. 9th Conf. Artif. Intell. Appl.*, Los Alamitos, CA, 1993, p. 454

[6] J. Lind, K. Fischer. Transportation scheduling and simulation in a railroad scenario: A Multi-Agent Approach. December 1998

[7] Krummheuer, E. (1997). Rendezvous und schnelle Sprinter. Verkehrsforum. http://www.verkehrsforum.de/magazin/archiv/1 97/1 97 2.html.

[8] Windhoff AG (1996). CargoSprinter. http://www.windhoff.de http://www.fortunecity.de/anlagen/atlantik/23/sprinter.htm.

[9] B¨urckert, H.-J., Fischer, K., and Vierke, G. (1998). Transportation scheduling with HolonicMAS, the TeleTruck approach. In Proc. PAAM98

[10] Smith, R. (1980). The contract net protocol: High-level communication and control in a distributed problemsolver. IEEE Trans. on Computers.

[11] Bachem, A., Hochst¨attler, W., and Malich, M. (1992). Simulated Trading: A New Approach For Solving Vehicle Routing Problems. Technical Report 92.125, Mathematisches Institut der Universit ¨at zu K¨oln.