Actor-Agent Application for Train Driver Rescheduling

Erwin J.W. Abbink[b] David G.A. Mobach[a] Pieter J. Fioole[b] Leo G. Kroon[b,c]
Eddy H.T. van der Heijden[a] Niek J.E. Wijngaards[a]

[a]D-CIS Lab
Thales Research & Technology NL
P.O. Box 90, 2600 AB
Delft

[b]Netherlands Railways
NSR Logistics Innovation
P.O. Box 2025, 3500 HA
Utrecht

[c]Rotterdam School of Management
Erasmus University Rotterdam
P.O. Box 1738, 3000 DR
Rotterdam

{erwin.abbink,pieterjan.fioole,leo.kroon}@ns.nl, {david.mobach,eddy.vanderheijden,niek.wijngaards}@icis.decis.nl

ABSTRACT

This paper describes the design, implementation, visualizations, results and lessons learned of a novel real-world socio-technical research system for the purpose of rescheduling train drivers in the event of disruptions. The research system is structured according to the Actor-Agent paradigm: here agents assist in rescheduling tasks of train drivers. Coordination between agents is based on a team formation process in which possible rescheduling alternatives can be evaluated, based on constraints and preferences of involved human train drivers and dispatchers. The research system is the result of cooperation on decentralised multi-agent crew rescheduling between Netherlands Railways (NS) and the D-CIS Lab. The implementation is realized using the Cougaar framework and includes actual timetable and rolling stock schedule data and driver duty data.

Categories and Subject Descriptors

J.m [**COMPUTER APPLICATIONS**]: Miscellaneous – *transportation*; I.2.1 [**ARTIFICIAL INTELLIGENCE**]: Applications and Expert Systems – *actor-agent systems*; I.2.11 [**ARTIFICIAL INTELLIGENCE**]: Distributed Artificial Intelligence – *multi-agent systems*; I.2.8 [**ARTIFICIAL INTELLIGENCE**]: Problem Solving, Control Methods, and Search – *scheduling*.

General Terms

Performance, Design, Experimentation, Economics.

Keywords

Distributed Systems, Applications, Cross-Cutting, Crew Rescheduling, Railway, Actor-Agent systems.

1. NS TRAIN DRIVER RESCHEDULING

The railway operations of Netherlands Railways (NS) are based on an extensive planning process, consisting of three phases: timetable planning, rolling stock scheduling, and crew scheduling. The crew scheduling process supplies each train with a train driver and with sufficient conductors. In the past years, NS has successfully applied novel Operations Research models to significantly improve the crew scheduling process for which they received the Edelman Award 2008 [2]. The current methods and techniques are very useful for generating the initial daily schedules, yet their calculation time is multiple hours, making them unfit for in-time rescheduling purposes. In this paper we focus on an actor-agent based approach for *rescheduling of train drivers*.

Cite as: Actor-Agent Application for Train Driver Rescheduling, Erwin J. W. Abbink, David G. A. Mobach, Pieter J. Fioole, Leo G. Kroon, Eddy H. T. van der Heijden, Niek J. E. Wijngaards, *Proc. of 8th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2009)*, Decker, Sichman, Sierra and Castelfranchi (eds.), May, 10–15, 2009, Budapest, Hungary, pp. 513–520

After the planning process, the daily plans are carried out in the real-time operations. Preferably, the plans are carried out exactly as scheduled. However, in real-time operations plans have to be updated continuously in order to deal with delays of trains and larger disruptions of the railway system.

A disruption may be due to an incident, or a breakdown of infrastructure or rolling stock. On the Dutch rail network (more than 5,000 daily trains), on average 10 disruptions of a route occur per day. Delays occur more frequently: On average 450 trains experience one or more delays (> 3 minutes) per day. These delays lead to removal of on average 10 train services per day.

NS train drivers operate from 29 crew bases. Each day a driver carries out a number of tasks, which means that he/she operates a train on a trip from a certain start location and start time to a certain end location and end time. The trips of the trains are defined by the timetable. Train drivers can use *positioning* trips to travel to the starting location of driving tasks. In addition, *standby* tasks are defined and assigned to *spare* train drivers: these can be used to resolve rescheduling problems. The tasks of train drivers have been organized in a number of duties, where each *duty* represents the tasks to be carried out by a single driver on a single day. Each duty starts in a crew base, and a hard constraint is that the duty ends at the same crew base within a limited period of time. Also several other constraints must be satisfied by the duties, such as the presence of a meal break at an appropriate time and location, and an average driver working time per crew base of at most 8 hours. Initially in the planning process, duties are anonymous, which means that the allocation of drivers to duties is still needed. The latter is handled by the creation of crew rosters, which describe the sequence of duties that are carried out by the individual drivers on consecutive days.

The total number of train drivers is about 3000. Each day, about 1000 duties are carried out. Furthermore, at any moment in time, the number of active duties at that moment is about 300. Due to removals and delays of trains or rescheduling of the rolling stock a number of duties of train drivers may become infeasible. An infeasibility of a duty is due to a time conflict (often caused by delays) and/or a location conflict (often caused by cancelled train services). In both cases, a conflict occurs between two consecutive tasks in the duty. Dispatchers are responsible for rescheduling tasks among train drivers so that all trains are

'manned'. Dispatchers are organised into four regions, where they are responsible for rescheduling train drivers who currently reside in their region. Often dispatchers need to handle task-rescheduling recursion, which they can handle to a certain extent given available time and resources. Typically, ca. five minutes are spent to resolve an inconsistent duty. Frequently, rescheduling problems are left 'open ended' for later resolution by other dispatchers (often in another region). In larger disruptions some trains simply cannot be driven as dispatchers are busy rescheduling train-drivers, causing additional delays for passengers.

The main objectives of the train-driver rescheduling research system described in this result-oriented paper are for D-CIS Lab to explore the effectiveness and suitability of a decentralized, actor-agent based approach to crew rescheduling. The main objective of NS is to determine whether multi-agent technology is sufficiently mature to be used in a real-world decision support system.

2. DESIGN

In this section the main principles underlying the actor-agent based train-driver rescheduling process are introduced. First, the applied design paradigm is introduced. After this, the two main elements of the rescheduling system are described.

2.1 Actor-Agent Paradigm

The actor-agent paradigm [6] explicitly recognizes both human actors and artificial agents as equivalent team members, each fulfilling their respective roles in order to reach the team objectives. The involved agents have quasi cognitive capabilities that are complementary to (as opposed to mimicking) human cognition. Actor-agent teams and communities are hybrid collectives of human experts ("actors") and agents with complementary cognitive capabilities which are focussed on a certain problem and reach a structural and functional complexity that matches the size and nature of the problem as good as possible.

The actor-agent based design process provides the system with several useful global system characteristics. First, the decentralized approach in which agents use local knowledge, world views, and interactions, contributes to an open system design. This openness facilitates easy reconfiguration and/or adaptation to changing system requirements. Second, combining humans and agents within the system design allows for integrating them at their appropriate abstraction levels.

2.2 Train-Driver Rescheduling

The main principle for the train-driver rescheduling application is to model the solution based on 'levels of responsibility' in the dispatch and rescheduling process:

- human dispatchers at the strategic/management level,
- human train drivers at the level of defining and guarding their personal interests, and
- their respective agents at the level of implementing the strategic/management decisions and resolving actual schedule conflicts.

In our approach driver-agents represent driver-actors in the rescheduling process. In the event of a disruption all driver-agents are informed of the disruption details (i.e. removed/delayed train services). The driver-agent(s) directly affected by a disruption (i.e., the disrupted train service is associated with a task in the driver's schedule) assume the role of *team leader*. Each team leader starts a team-configuration process in order to resolve their respective schedule conflicts.

The main principle underlying the actor-agent based rescheduling process is that of *task-exchange*: Each team is extended with additional team members able to take over tasks from agents already participating in the team. A driver-agent may be able to take over tasks without affecting other tasks already present in its schedule, for example replacing a positioning trip with a task (*unconditional* takeover). However, in most cases, a driver-agent will only be able to take over tasks by replacing existing tasks in its schedule (*conditional* takeover). This will then lead to a new set of conflicting tasks to be taken over by another driver-agent. In Figure 1, an example of this process is shown: driver-agent B replaces task U-T with task X-Y from driver-agent A. Driver-agent C now takes over task U-T from driver-agent B unconditionally.

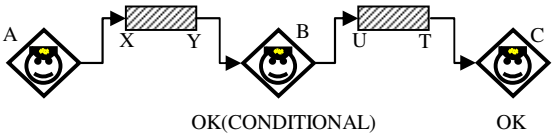

Figure 1 : The task takeover process

In case a driver-agent has determined that tasks can be taken over, a cost-function is applied to determine the costs associated with the takeover: costs are assigned to various aspects such as the amount of overtime introduced, replacement of meal breaks, etc. Subsequently the set of new conflicting tasks of this agent (if any) is announced to other driver-agents. This leads to a recursive addition of layers of team members to the team, resulting in a team consisting of multiple task-exchange configurations, originating at the team-leader.

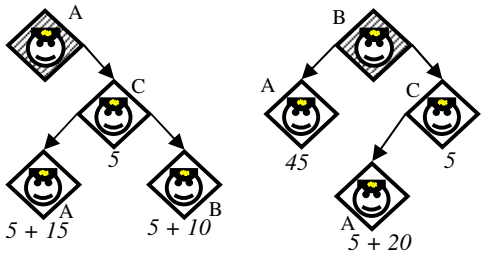

Figure 2: Two example task exchange teams consisting of three agents.

In Figure 2, two team configurations are shown, consisting of three driver-agents (A, B and C): In the left team, driver-agent A is team leader, and starts the task-exchange process. The same driver-agent participates as a team member in its own team, as well as (in two different configurations) in the team led by driver-agent B: This feature allows for any driver-agent to participate in possible task-exchange configurations and thus to facilitate the solution process to find the best task-exchange configurations for each team. Driver-agents can withdraw themselves from teams and team-configurations based on the commitment levels in the

task-exchange protocol, ensuring local and global consistency when final team-configurations are determined. Details of this protocol are beyond the scope of this paper.

The team extension process is considered complete when a configuration of task-exchanges is determined in which all conflicts are resolved, or any remaining conflicts are sufficiently shifted forward in time to be resolved at a later point in time (re-introduced as new conflicts later). At this point, the recursive team formation process is reversed: Each layer within a team selects the task-exchange associated with the lowest cost, starting at the lowest layer. Once all team leaders have determined a final team configuration, the entire solution is presented to the dispatcher.

During the team formation process, a number of heuristics are used, aimed at limiting team extension to promising additional team members. These heuristics currently include:

- If a task takeover leads to a new conflict in the duty of the agent taking over the conflict, this conflict must be positioned later in time. This ensures that conflicts are always shifted forward in time.
- A scoreboard mechanism is used to inform agents about the best solution found in a team. Agents use this to determine if team participation is useful (see section 3.1.2).
- Detecting and aborting similar task-exchange configurations: many team extensions lead to driver-agents ending up with similar conflicts which need to be resolved. When a driver-agent detects that a conflict resulting from a tak-over is similar to another conflict this agent is already resolving elsewhere in the same team, the agent does not pursue this new option.

2.3 Route Calculation

An important element in the rescheduling process described above is determining whether a driver-agent is able to take part in task-exchanges. To this end, a model of the timetable is included in the system design, containing all available train services. This model is updated with disruption information during run-time to ensure that it reflects actual available train services. Using this model, driver-agents can analyze whether a task-exchange is possible by determining the answers to the following questions:

1. Is the driver able to get to the starting location of the specified tasks on time, from its current location?
2. Is the driver able to return to his/her own schedule after completing the specified tasks?

If it is determined that the tasks specified in the task-exchange can be taken over, the impact of this exchange will have to be determined: The route calculation process is aimed at preserving as much of the original train driver's schedule as possible, limiting the number of originally scheduled tasks that will become infeasible due to the exchange.

3. IMPLEMENTATION

The research system is constructed in the real-world domain of disruptions on the Dutch railroad network as a reliable and scalable operational system which can run together with the current systems in place. The work on this research system is partly subsidized; therefore it is not constructed as a full production system. As performance is required in the order of minutes (i.e. comparable to or better than human dispatcher performance), the following implementation decisions were made:

- The actor-agent based solution process is defined first, after which an accompanying architecture is designed, on the basis of which suitable technologies are selected.
- To support the explorative nature of the project, it is necessary to (always) have a running research system for testing, debugging and assessment purposes.
- The research system is to be as decentralized as possible.

The research system focuses on rescheduling train driver duties in real-time over the course of a single day. It is assumed that any necessary timetable and rolling stock plan modifications to cope with disruptions have been implemented, and a new rolling stock plan is in place, to which the driver schedules are to be adapted. Whenever a solution to disruptions has been accepted, this 'new' plan is the basis for the rescheduling in response to new disruptions (or a change in duration/consequences of an 'old' disruption).

For the implementation of the research system, the Cougaar agent framework [1] is used. Cougaar has initially been developed for logistics operations and provides a useful Java-based agent platform with emphasis on stability. Cougaar provides an agent model based on *plugins*, allowing for clean separation of functionality within agents. Furthermore, Cougaar allows for implementing *services*, which can be made available to agents running on a Cougaar *node*. Cougaar agents use a distributed *blackboard* for storing information and communication purposes.

To start the system a bootstrap-agent dynamically instantiates the driver-agent population and other main agents. This allows for a flexible startup process, enabling for example dynamic instantiation of the agents across a multiple-node configuration.

In order to run realistic scenarios, a dataset containing train activities and driver tasks for a full day has been provided by NS. The dataset is distilled from multiple data sources, is stored in a MySQL database, and is made available to agents through a *SQL-Service*. The database is to be connected to real-time disruption information for more continuous experimentation of the research system.

3.1 Approach

The implementation of the rescheduling system is realized using a cyclic (re)design, implementation, and evaluation process, comparable to the well-known rapid prototyping type of approach. The three main implementation cycles of the research system are discussed below.

3.1.1 Cycle 1: Demonstrator

A small-scale version of the research system is implemented, consisting of less than 10 driver-agents and with coarse grained route calculation capabilities. This version focuses mainly on the implementation of the driver-agents and the task-exchange protocol: All driver-agents are instantiated with a driver schedule and are able to communicate with each other using the aforementioned blackboard in order to from teams and resolve conflicts due to disruptions. In this version, all driver-agents are

designed to act in a de-centralized way; that is, all agents have access to an internal model of the rail network, in order to determine possible routes required in the task exchange process.

Cycle 1 has resulted in a demonstrator system at the end of 2007. Based on this demonstrator, a decision was made to continue research and development in 2008.

3.1.2 Cycle 2: Scaling up

In the second cycle the number of driver-agents and the route calculation functionality are scaled up in several steps, resulting in a version supporting a sizable driver-agent population with full route calculation functionality.

Evaluation of the demonstrator showed that incorporating a model of the real-time rail network in every driver-agent introduces a large amount of computational overhead. This resulted in the separation of this functionality into a *network-agent*. On the one hand, this results in a large amount of communication between driver-agents and the network-agent. On the other hand, by keeping a centralized view on the rail network in the system, the network-agent is able to calculate routes more efficiently (e.g. request caching), resulting in less (overall) memory usage and reduced calculation times.

A drawback of this approach is that due to the large amount of route calculation requests initiated by the driver-agents, a substantial queue of requests is formed at the introduced network-agent, impeding the rescheduling process by delaying answers returned to the driver-agents. Analysis of the requests shows that ca. 50 % of the requests require detailed rail network calculation. The remainder can be answered without calculation. For example, for a request where 200 kilometres have to be covered within 60 minutes it is unnecessary to perform a complicated network-calculation when it is known that trains cannot travel faster than 130 kilometres per hour on the Dutch rail network.

To take advantage of this property, a *route-analyzer-agent* is introduced to interact with driver-agents. The route-analyzer-agent is supported by several network-agents, and pre-evaluates all requests to determine if a request can be answered without involving detailed network calculations. The route-analyzer-agent is able to do this very efficiently by applying global knowledge of the network and by using historical knowledge from previous requests for the same disruption.

Additionally, the network-agents can be distributed to process requests in parallel, with the route-analyzer-agent acting as the central point of contact for the driver-agents. The route-analyzer-agent maintains a priority queue on requests, to further optimize the usage of the network-agents. The effect is that each network-agent now has to process less requests, (i.e. only those where detailed information of the network is required).

In addition to the changes regarding the representation of the rail network and the calculation of routes, efforts are made to apply heuristics to make the rescheduling process more efficient by regulating the size of the task-exchange teams. To this end, a *scoreboard* mechanism is introduced: Before joining a driver-agent team, each driver-agent compares the costs of the task-exchange configuration it is part of (i.e. the task exchanges leading up to the task-exchange this driver-agent is evaluating) against the current 'best' (i.e. lowest) costs published on the scoreboard. The scoreboard mechanism ensures that only solution alternatives are evaluated which may improve the currently found solution(s), due to the strictly increasing property of the cost-function.

At the end of cycle 2, a version of the research system was realized capable of finding solutions for relatively large disruptions with a driver-agent population consisting of up to 200 agents within reasonable time.

3.1.3 Cycle 3: Full-scale prototype

In the third cycle a full-scale prototype is developed, ultimately resulting in an evaluation phase during which the system runs in parallel with existing rescheduling systems. Important issues confronted here are the dispatcher-interface to the rescheduling system and the connection to real-time disruption data streams.

With respect to the dispatcher-interface, again a centralisation versus decentralisation issue is tackled (yet on smaller scale with less impact on the overall performance than in cycle 2). The first dispatcher-agent (as only one was developed in the first phase) is initially endowed with functionality to monitor and control the task-exchange team configuration process of the driver-agents. During subsequent developments, the dispatcher-agent grew in complexity (including the GUI it presents to a dispatcher) and additional dispatcher agents were designed. The monitoring and control functionality was not replicated in each dispatcher agent; a new agent was developed and charged with this responsibility: the *process-manager-agent*.

Currently, cycle 3 is ongoing. A full-driver agent population is supported. This cycle is expected to be concluded in the first quarter of 2009.

3.1.4 (De)centralisation

Our pragmatic approach allowed for specific reflection moments to assess our current progress, including the (emergent) behaviour of the overall system. The above described evolution of the architecture shows how an initial, fully decentralised approach was deliberately changed into an architecture in which two distributed subsystems can be distinguished:

- The driver-agent rescheduling subsystem, including dispatcher-agents, process-manager-agent (PMA), and the driver-agents.
- The route-analyzer subsystem, including the route-analyzer-agent (RAA) and network-agents (NA).

Figure 3 shows the agents and their relations as implemented in the current research system:

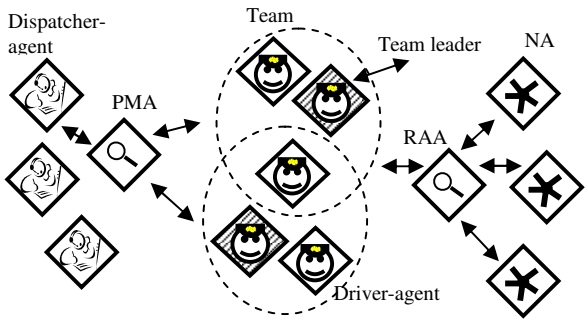

Figure 3: Current research system architecture.

3.2 Debugging Tools

The complexity of the train-driver rescheduling process can be expressed in the large number of involved agents and the difficulty in assessing the quality of a solution found. During the implementation of the research system a number of debugging tools were needed, as the currently available debugging tools were inadequate.

One important approach was to devise specific tests (e.g., small disruption scenarios with only three involved driver-agents who should find an optimal solution in a not so complex way) for each new increment of the research system.

Another tool was needed to assess the quality of solutions found by the research system. For this purpose, the dispatcher-agent provides a GUI which represents train driver duties in the same manner as is currently used by dispatchers[1]. In addition, the human dispatcher can specify a disruption scenario via this GUI, and configure the solution process parameters (e.g. weights of the cost function elements, scoreboard settings). The results of the team-configuration process are also presented to the dispatcher using the GUI (see Figure 4).

Most agent frameworks do not provide sophisticated debugging tools at the level of *multiple* agents. Although communication logs are of interest to understand one agent's functionality, the more emergent effects of multiple interacting (and even cooperating) agents need to be analyzed at a higher level of abstraction.

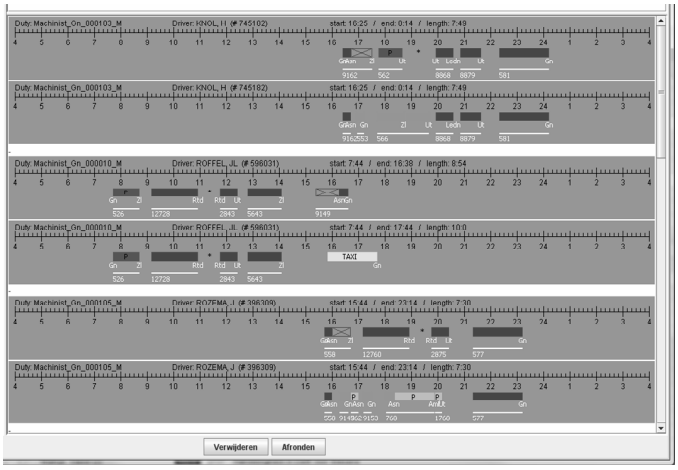

Figure 4: Dispatcher GUI displaying train driver duties.

For this purpose, a visualization tool is developed to gain insight in the dynamics of the team-formation process. Based on event logs of the driver-agents a graphical view of the interactions between the driver-agents over time is visualized[2]. Figure 5 shows a snapshot of a replay of the team-formation process: Each cluster of nodes represents a team. Within these teams, the nodes represent driver-agents taking part in the task exchange process: Driver-agents acting as team leaders are depicted using a rectangle, team members using circles. The team-formation visualization tool has proven its value in debugging of the team-formation process and quality assessment of proposed solutions. In addition, this visualization tool facilitates explaining our approach to train driver rescheduling to a wider audience.

4. INITIAL RESULTS

This section provides an overview of our current[3] findings. The current version (October 2008) of the system is able to find solutions for relatively large disruptions, using sizeable driver-agent populations.

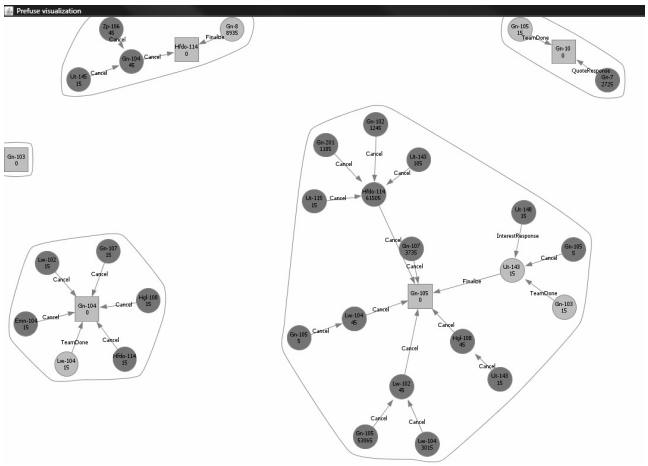

Figure 5: Team-formation visualization tool.

To illustrate this, results of two relatively complex example scenarios are presented. The first scenario consists of a complete blockage between stations Groningen and Zwolle from 16:00 to 17:00. The second scenario consists of a complete blockage at station *Vught*. Figure 6 provides a simplified overview of the Dutch railway network, in which the locations of the disruption scenarios are indicated. Solid lines denote the network responsibilities of NS, dotted lines are handled by other operators. The results described below are compared to the research system itself instead of compared to results of other systems: work is in progress to compare results of actual disruption scenarios with real world dispatcher responses. The acceptance and use of the research system by human dispatchers will be studied at the end of development cycle 3.

The results discussed in this section are qualified by the number of driver-agents involved in the final solution, as well as the number of spare drivers and additional overtime introduced. The aim of the result analysis in the two scenarios is to assess scalability and behavior of the research system. Interestingly enough, not only the number of driver-agents is of importance to assess scalability, but also whether spare driver-agents are involved. The cost function adds extra cost for the use of spare driver-agents; the intent is to avoid making spare driver capacity too 'cheap'. In addition, the cost function heavily depends on

[1] These representations are e.g. used in the CREWS application (www.siscog.pt) and in the Resource Manager system by FUNKWERK.

[2] Based on the *Prefuse Information Visualization Toolkit* (http://prefuse.org/)

[3] The findings presented in this section are based on the early Q4 2008 version of the research system. The current version (Q1 2009) contains a number of improvements in performance and solution quality.

overtime by involved driver-agents, which is shown for the final solutions found.

The number of messages exchanged between agents provides an indication of the 'performance' of the research system. The route-analyzer-agent together with the network-agents comprise the computationally most expensive part of the research system; the number of routes calculated provides an indication of the effort spent. In addition, calculation times are provided: these times are indicative only (although they do indicate that the research system already outperforms a single human dispatcher), as the research system has not been (re-)engineered as a real-time operational system.

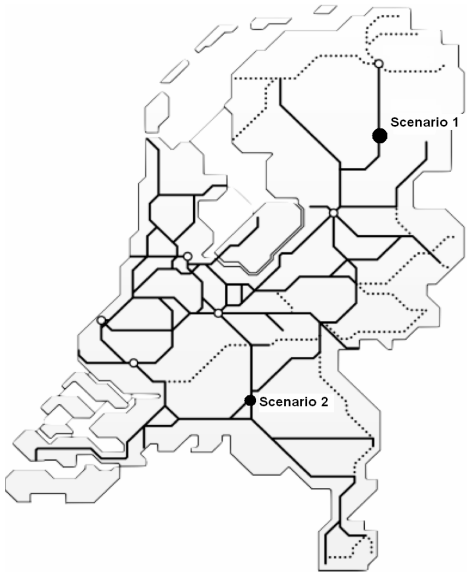

Figure 6: Simplified Dutch railway network and location of disruption scenarios.

4.1 Scenario 1: Blockage Groningen-Zwolle

For scenario 1 the number of cancelled train services due to the blockage is 11, which leads to 11 driver-agents to act as team-leaders. Table 1 shows the results for the first scenario of various runs with different driver-agent populations.

In run 1 all 59 selected driver-agents are located near the disruption at the time of the disruption: in this case a solution can be found. This solution could be improved by including a spare driver, as is shown in run 2. In run 3 there are 28 additional driver-agents included that are located near the disruption, but after the actual disruption itself is already solved. This also improves the solution found, as driver-agents now have the possibility to exchange tasks forward in time until later in the evening, when more free capacity is available.

In this case adding extra spare drivers does not contribute to a better solution. It does, however, help to find the *same* solution more quickly, as is shown in run 4. This can be explained by the scoreboard mechanism: Even though the spare driver is not chosen in the final solution, these driver-agents publish a value on the scoreboard early on in the process, aiding in the elimination of many more costly solutions at an early stage.

In run 5, 48 driver-agents are added which are located relatively far from the disruption at the time of the disruption. None of these added agents contribute to an actual solution. This is detected quickly by the system: The final solution remains the same and the time needed to find the solution increases only marginally.

Table 1: Results scenario 1.

Run	# driver-agents (spare drivers)	Total # final team members	Overtime (minutes)	# spare drivers in final solution	Average calculation time per team-leader (seconds)[4]
1	59 no spare	21	237	0	0:32
2	59 +1 spare	14	0	1	0:06
3	87 no spare	14	0	0	0:08
4	87 +4 spare	14	0	0	0:07
5	135 +4 spare	14	0	0	0:10
6	183 +4 spare	14	0	0	0:18

In run 6, 49 driver-agents are added that are located relatively close to the disruption. These agents do not contribute to an improvement of the final solution but did participate in the team-formation process. Therefore the calculation time for this run increased substantially as opposed to the previous runs.

Table 2: Communication statistics for scenario 1.

Run	# messages route-analyzer-agent	# routes calculated by network-agent	# of messages sent
1	41.314	13.920	398.000
2	4.826	1.944	52.250
3	6.943	1.952	73.500
4	4.893	2.033	61.500
5	8.465	2.390	108.750
6	14.182	4.171	167.000

As shown in Table 2, a large amount of communication between the agents was needed to find the final solution for run 1. Because the final solution is not a very good solution in terms of overtime and number of agents participating, a large number of alternatives have to be considered during the negotiating process in order to conclude that a better solution does not exist. In run 2 where the final solution has no overtime, a large number of alternatives which were explored in run 1 can now be eliminated beforehand due to the score-board mechanism. This leads to much less communication. In runs 3, 4 and 5 there are more agents in the system, which leads to more communication; in these cases most of the extra requests for routes from the driver-agents can be

[4] Calculation times indicative only; System configuration: Intel Pentium D 3.4 Ghz. 2.0 GB RAM

handled by the route-analyzer-agent without sending them to the network-agents. Because of this the calculation time increases only marginally. Only in run 6 the network-agent has to process almost twice as many requests, which leads to the increase in calculation time.

4.2 Scenario 2: Blockage Vught station

For scenario 2 the number of cancelled train services due to the blockage is 14. Table 3 shows the results for the first scenario of various runs with different driver-agent populations.

Table 3: Results for scenario 2.

Run	# driver-agents (spare drivers)	Total # team members	Overtime (minutes)	# spare-drivers in final solution	Average calculation time per team-leader (seconds)
1	33 no spare	20	501	0	0:19
2	33 +2 spare	19	386	2	0:15
3	106 +2 spare	28	121	2	0:39
4	106 +10 spare	26	0	4	0:37
5	157 +10 spare	26	87	3	0:44

In the first run only the driver-agents who are located close to the disruption at the time of the disruption were included. These agents were able to find a solution for the disruption, but they needed to introduce a lot of overtime. After including two spare drivers in run 2 the system was able to find a better solution. In run 3, more agents that are located relatively close to the disruption were included. This resulted in a more acceptable solution, but also resulted in more than twice the calculation time. Including all available spare drivers in the neighborhood of the disruption improved this solution further. In this case 4 spare drivers were actually used in the final solution, the others helped to find the solution more quickly.

In run 5, 51 driver-agents are added that are located further away from the disruption at the time of the disruption. In this case there are 87 minutes of overtime in the final solution where there was no overtime in the solution of run 4. This can be explained by the fact that the solution found in run 5 contains one less spare driver-agent in the final solution.

Table 4 shows that when a relatively good solution can be found early in the negotiation process, the score-board mechanism is able to reduce the total amount of communication needed. In run 4, the total number of requests from driver-agents to the route analyzer-agents decreased with almost 30% and the total number of messages decreased by more then 20% compared to run 3.

This can be explained by the elimination of a large number of alternatives by the score-board mechanism early in the negotiation process. The number of requests processed by the network-agent increased slightly, because additionally the routes for the spare drivers had to be calculated. In run 5, there was more communication needed than in the fourth run, but still less than in the third run.

As shown in the results discussed above, the score-board mechanism helps to find good solutions quickly, in cases where a good solution exists. In cases where only solutions with high costs exists (i.e. complex solutions with many task-exchanges) the scoreboard is not seeded with relatively low-cost values leading to the evaluation of almost all alternatives in the search-space, before concluding that the best solution has very high costs. A possible approach currently explored to solve this problem is to allow *partial solutions* of a conflict to be published on the scoreboard. A partial solution means that a conflict is sufficiently moved forward in time to be solved at a later point in time.

Table 4: Communication statistics for scenario 2.

Run	# messages route-analyzer-agent	# routes calculated by network-agent	# of messages sent
1	20.190	9.246	190.841
2	20.056	8.260	187.033
3	46.012	10.136	431.528
4	33.741	11.421	341.316
5	37.551	12.561	388.892

5. DISCUSSION

The train-driver rescheduling research system is a real-world application of actor-agent systems. Since summer 2007, about nine man years of research & development have been successfully invested. The performance of the current system (ca. 200 driver-agents) is expected to grow unto the full 1000+ driver-agents in Q1 2009. The expected performance in the context of (usually) 300 active train drivers looks promising given the current developments. The current calculation times indicate that the research system easily outperforms a human dispatcher in the quantity and thoroughness of the solutions found. One possible use of the research system is to study the combination of rescheduling properties of initial daily crew schedules given specific disruptions and rescheduling parameters. This may lead to new insights in the role and use of spare train-drivers as well as the amount of 'free time' required in the train driver duties.

5.1 Related work

Traditionally, crew scheduling problems are approached using Operations Research techniques. Real-time crew rescheduling however is a relatively new area of research. To our knowledge, no research has been published on agent-based crew rescheduling applications in the railway domain. Shibghatullah et al. [4] propose an agent-based framework for bus crew scheduling including crew-reassignment. The paper provides an overview of the potential advantages of agent-based approaches (e.g., modelling individual preferences, more suited for partial, on-demand rescheduling), but lacks further details of the proposed framework. In [5], Tranvouez et al. present a multi-agent based approach to disruption management in the supply chain domain. A distinction is made between partial and complete rescheduling, as well as periodic and event driver rescheduling. It is argued that partial, event driven rescheduling appears to increase schedule

stability. Our rescheduling approach can also be classified as partial, event-driven rescheduling. Schedule stability is also an important and desirable characteristic in the railway crew rescheduling domain, given the fact that initial crew schedules are already optimized for efficiency, and changes should be minimized. Tranvouez et al. further describe a BDI architecture, for its potential to design complex decision making processes and to represent rich domain expert knowledge.

Mao et al. [3] recognize the need for short-term operational planning and scheduling methods in the domain of airport resource scheduling, and present an agent-based approach based on two coordination mechanisms: decommitment penalties and a Vickrey auction mechanism. The coordination approach used in this paper is based on a combination of similar mechanisms: The driver-agent interaction protocol has auction-like properties (agents report costs (i.e. bid) for taking over tasks), and decommitment penalties are determined based on increasing commitment levels. In literature, coordination approaches based on negotiation concepts are often divided in cooperative and non-cooperative (self-interested) approaches. Although driver-agents in our model can in some respects be considered as self-interested agents (driver preferences are included in the agent's cost function), the agents cooperate to achieve the global goal of resolving disruptions, and agents do not engage in direct competition.

5.2 Lessons Learned

A number of lessons learned can be distilled at this point in the project:

- *Co-development is crucial.* The close cooperation between NS and D-CIS Lab enabled the transfer of domain knowledge as well as actor-agent knowledge to the mutual benefit of both parties and the overall success of the research system.
- *Real-world data is difficult.* More effort than expected was invested in understanding the problem domain and distilling useful parts of real-world data (incl. incompleteness, inaccuracy and sometimes unavailability).
- *Re-design is not evil.* During the development of the research system progressive insight led to re-design of important parts of the driver-agents and the route-analyzer-agent. Although this effort seemed distracting, the overall research system's quality improved substantially.
- *Decentralization is not a silver bullet.* A careful balance has to be made between decentralizing task-exchange functionality from centralizing route analysis and calculation to boost performance without violating our actor-agent principles.
- *Multi-agent debugging tools are a necessity.* Message analyzers are not enough to debug a full fledged multi-agent system; an additional visualization tool was created to debug the (emergent) team-formation process.

The results of this project encourage us to continue our work on agent-based rescheduling in real-world domains. Future work entails using the research system as a platform for additional performance analysis and testing of techniques, cost-functions as well as agent platforms.

6. ACKNOWLEDGMENTS

The authors express their gratitude to NS and Cor Baars (DNV / Cibit) for starting this project. The following D-CIS Lab colleagues provided valuable contributions: Hilbrandt van Boven, Pascal Hoetmer, Michel Oey, Reinier Timmer, Louis Oudhuis, Martijn Broos, Sorin Iacob, Thomas Hood, and Kees Nieuwenhuis. The research reported here is part of the Interactive Collaborative Information Systems (ICIS) project (www.icis.decis.nl), supported by the Dutch Ministry of Economic Affairs, grant nr: BSIK03024. The ICIS project is hosted by the D-CIS Lab (www.decis.nl), the open research partnership of Thales Nederland, the Delft University of Technology, the University of Amsterdam and the Netherlands Organisation for Applied Scientific Research (TNO).

7. REFERENCES

[1] Helsinger, A., Thome, M., and Wright, T. (2004), Cougaar: A Scalable, Distributed Multi-agent Architecture. In: *Proc. of the Int. Conf. on Systems, Man and Cybernetics*, The Netherlands.

[2] Kroon, L., Huisman, D., Abbink, E., Fioole, P.J., Fischetti, M., Maróti, G., Schrijver, L., Steenbeek, A., Ybema, R. (2008). The New Dutch Timetable: The OR Revolution. In: *Interfaces* (to appear).

[3] Mao, X., ter Mors, A., Roos, and N., Witteveen, C. (2007), Coordinating Competitive Agents in Dynamic Airport Resource Scheduling. In P. Petta, J. P. Mueller, M. Klusch, M. Georgeff (Eds.). *Proc. of the 5th German Conf. on Multiagent System Technologies*, LNAI, Springer Verlag, vol. **4687**, pp. 133-144.

[4] Shibghatullah, A.S., Eldabi, T., Rzevski, G. (2006), A Framework for Crew Scheduling Management System Using Multi-Agents System. In: *28th Int. Conf. on Information Technology Interfaces (ITI 2006)*, Cavtat, Croatia.

[5] Tranvouez, E., Ferrarini, A. (2006), MultiAgent Modelling of Cooperative Disruption Management in Supply Chains. In: *Int. Conf. on Service Systems and Service Management* (2006), Troyes, pp. 853-858.

[6] Wijngaards, N., Kempen, M., Smit, A., and Nieuwenhuis, K. (2006), Towards Sustained Team Effectiveness. In: Lindemann, G., et al. (Eds.), Selected revised papers from the workshops on *Norms and Institutions for Regulated Multi-Agent Systems* (ANIREM) and *Organizations and Organization Oriented Programming* at AAMAS'05, LNCS, Springer Verlag, vol. **3913**, pp. 33-45.