ABSTRACT
The paper describes functionality of Magenta Multi-Agent Logistics i-Scheduler Engine presented on AAMAS 2006 conferences and gives examples of its application in business domain.

The i-Scheduler Engine was designed to be scalable without risk of combinatorial explosion, in order to handle large transportation networks as a whole. The multi-agent architecture combined with semantic network allows very granular approach for every business entity of transportation network (client, order, cargo, truck, driver, etc) and balancing of their conflicting interests. The i-Scheduler considers individual constraints and, interestingly, specific preferences of customers, drivers, trucks, cargoes, etc. This results in a unique ability to combine inbound and outbound deliveries, different fleets or private networks, driving more value from finding effective backhauls and consolidations.

The paper covers the history of development, architecture and current functionality of the engine and provides a set of case studies in different transportation networks, which outline the most serious challenges Magenta overcame in each case.

Categories and Subject Descriptors
I.2.11 [Computing Methodologies]: Artificial Intelligence – distributed artificial intelligence

General terms

Keywords
Vehicle routing problem, resource allocation, incremental scheduling, truck scheduling, dynamic routing, crossdocks, large-scale transportation networks, contract net, multi-agent systems, distributed decision making, ontology, semantic networks.

1. INTRODUCTION
Vehicle routing problem (VRP) is amongst the most complex business problems. The reason is exceedingly high variety of possible solutions (large solution space), and uncertainty due to high dynamics and volatility of the operational environment and openness of business networks [1]. All this makes traditional combinatorial search and heuristics-based algorithms applicable only for certain types of VRP problems in trucking industry. Most large logistic providers (e.g. Third Party Logistics providers or 3PLs) have to rely on manual scheduling, because traditional software cannot handle complexity of the domain. Computer information systems are mostly used for supporting data flow, storage and visualization.

In recent years, the dynamics of businesses has increased significantly, and this process is gaining strength. Clients of transportation companies want a wider variety of products to be delivered in shorter terms, with increasing penalties for missing the delivery windows. Also, to raise service level, transportation companies have to consider individual preferences of clients.

The other trend is merging of transportation companies, trying to provide wider variety of services in different geographic regions and cut operational costs. This means increasing pressure on planners in terms of available time and amount of decisions to be made, forcing them to make simpler decisions or even leading to mistakes [2].

At the time, when rising fuel prices cause shrinking margins of transportation companies, effective transportation becomes the crucial factor for survival.
2. MODERN SCHEDULING PRACTICES

There are two common approaches for managing the complexity of modern transportation networks. The first is splitting the network into parts. An example can be scheduling different private networks (a network assigned to a specific customer) separately or separating inbound (supplier-to-DC) and outbound (DC-to-store) networks. Moreover, an inbound network can be then further split between a number of planners, each responsible for his own region or set of clients. Due to time pressure, interaction between planners is limited.

While this approach is effective in parallelizing scheduling process and cutting solution space, it leads to loosing visibility of the network. As different parts are scheduled separately, many backhaul and consolidation opportunities are lost, resulting in decreased fleet utilization and increased mileage. Thus, many companies consider combining their transportation networks as an important step in increasing the efficiency of operations.

Another approach is using fixed routes. Although order flow may change from day to day, the average of distribution may not change so quickly. Therefore, operator builds a long-term schedule, based on typical flow. It can be revised as rarely as once a month. This is done manually or with the help of batch optimizers, where they are applicable. Every day, as actual orders arrive, they are matched to corresponding fixed routes. However, as volumes may differ from day to day, planners have to revise the schedule, creating new journeys, canceling and merging existing ones, or searching for alternative routes with available capacity. In large networks, this is often a very complex and intricate process carried out under high time pressure. Due to limited flexibility of scheduling process, resulting schedule often consists routes, which are not logical regarding actual order flow. Still, this approach is field-proven and widely used.

3. SCHEDULER REQUIREMENTS

The above-described trends and business processes in trucking industry drive the requirements for an effective road transportation scheduler for medium and especially large-size operators.

The scheduler must be able to handle large transportation networks in near real time, support complex operations like crossdocking [3], consider fixed routes and fit the schedule into numerous constraints imposed by warehouse working hours, driver work rules, safety regulations and enterprise policies, e.g. on choosing between own fleet and third-party carriers.

In addition, the scheduler is expected to rapidly reschedule orders and transportation resources affected by unexpected events, such as arrival of new orders, cancellations, failures, bad weather conditions, road works and no-show of drivers or loading crews.

Existing solutions cover only part of these requirements. This fact is proven by the research, performed by one of our client transportation companies, described in section 8.3. They examined the market and found no solution, which could support crossdocking operations, combined with near real-time scheduling. As result, they chose one solution – Paragon Fleet Manager – for supporting smaller and more static part of network, which of course was not a satisfactory result.

This forces many companies to rely on manual process, at least for most complex parts of network.

4. HISTORY OF MAGENTA LOGISTICS I-

SCHEDULER DEVELOPMENT

Magenta Technology, launched in 1999, is one of the first industrial companies, who have been focusing on the development and commercial application of multi-agent and semantic web technologies. Today the company employs over 120 people globally, with headquarters in London and software development taking place in Samara, Russia. Magenta has developed systems in the areas of supply chain, logistics, scheduling, resource allocation, business intelligence and many other applications [4].

Road transportation logistics was chosen as one of the most complex business problems, where current scheduling software often fails to deliver satisfactory results. The J2EE development of the first generation of enterprise-ready Logistics i-Scheduler 2.X started in September 2003 and originally was focused on courier companies with real time delivery. Later we realised that multi-agent technology can bring most significant value in complex and large-scale transportation networks, where existing scheduling software usually fails to provide feasible results and manual process becomes too complicated.

The development of the new generation of i-Scheduler, Logistics i-Scheduler 3.0, started in September 2005. It was preceded by a set of experiments in fully distributed decision-making in resource allocation. The focus of the new i-Scheduler was scalability (at least 300 trucks and 2000 orders per day), and open architecture, allowing to increase the complexity and variety of supported operations. Another important requirement was effective support of crossdocking and trailer swap operations, which are widely used in large networks. At the same time, such important features as continuous scheduling, high level of customization and individual approach to every business entity were inherited from i-Scheduler 2.3.

The new requirements suggested adoption of fully decentralized decision making process and fully autonomous agents. The agents extract knowledge about the current situation from semantic network and adjust the network as the result of their activity. The main idea is that for large-scale problems effective schedule can be produced by continuous pro-active interaction between agents in attempt to improve their current status, instead of going into deep combinatorial chains. Decentralized and autonomous agents provide extendable architecture, which allows introducing new business entities and types of activities in scheduling process.

The first version of the new scheduler was finished in December 2005 and was battle-proved in a number of different prototype applications for UK and US truck operators. Three cases of application are described in more details below.

The i-Scheduler is constantly developing and currently it embraces the most important requirements for handling complex transportation networks. At the same time, work continues on enhancement of existing algorithms to further improve scheduling quality and performance. In September 2006 a commercial prototype of full-scale multi-agent Truck Engine for the...
scheduling system for a new UK operator was successfully completed.

5. I-SCHEDULER FUNCTIONALITY IN BRIEF
Currently Magenta Logistics i-Scheduler has the following features.

General functionality
- Incremental continuous scheduling
- Personal constraints and preferences for different business entities of a typical transportation network (orders, trucks, drivers, etc)
- Operator dashboard (see Figure 1): Map, Gantt chart, table views, etc
- Cross-platform

Logistics functionality
- National size UK / US networks
- Different truck types
- LTL/FTL cargoes
- Order delivery windows with stressing capability
- EU Driver regulations
- Crossdocks
- Dynamic route selection via a chain of crossdocks
- Secure/unsecured trailer swap operations
- Driver shift planning
- Events handling: truck broken, truck late, order cancelled, etc
- Economic reports on every journey / truck.

Performance and scalability
- 2000+ orders per day
- 500+ geographic locations
- 500+ trucks
- 20+ crossdocks and trailer swap points
- 1-10 seconds per order depending on scale and complexity of network

The described level of performance was sufficient for all the datasets we worked with. The datasets included large UK and US 3PL providers. A set of experiments conducted earlier [5] showed that processing time per order is approximately linearly dependent on the amount of simultaneously processed orders.

6. I-SCHEDULER ARCHITECTURE
The Logistics i-Scheduler is based on J2EE platform. The core part of i-Scheduler is the scheduling engine, which can be divided into two layers: data layer, which holds representation of the current network, and business logic layer, represented by agent swarm. Data layer is called scene. It is based upon concepts and relations, described in ontology in a form of semantic network, storing domain knowledge (see Figure 2).

The scene holds information about the current situation in transportation network: geography, position of vehicles, schedules, orders, etc. Each instance in scene has its version, so that agents can recognize changes in the region they are interested in. Instances are linked with each other by relations of different types, allowing agents to navigate scene, quickly extracting required information. Some entities in the scene (trucks, orders and journeys) are represented by agents. In other words, each agent has a corresponding instance in the scene, but the opposite is not always true.

The agents, corresponding to business entities, form the agent swarm. There can be thousands of agents of different types working simultaneously (on certain datasets, there were more than 10000 agents). Agent swarm is controlled by dispatcher, which
distributes processor time between agents. In current implementation it sequentially passes control to agents in an endless cycle. If no agents want to act, dispatcher causes the swarm thread to sleep for certain time, saving processor time.

Each type of agent has its own decision making logic and goals. A goal is defined as the superposition of a set of criteria with different weights. Individual agent can have its own weights assigned for different criteria. This allows individual approach to different entities. On the other hand, real-life applications showed that this model is overcomplicated in tuning and analysis. In most cases it is sufficient to have prioritized set of criteria instead of using superposition.

Each agent has “memory” – a snapshot of part of scene, which affects its state. For example, for journey agent this is journey schedule, the assigned truck and allocated orders. Like the scene, the memory is also represented by a semantic network. Agent uses its memory to compare his knowledge against current situation in reality scene and detect changes. Changes detection is based on comparing scene instances version. Memory is also used to build plans, which can later be applied to reality scene. A plan is a set of changes, agent wants to apply to the network. Plans are built in agent memory to make sure that common scene is not corrupted in case of unpredicted agent behavior. This approach also allows agents to work in parallel computation threads, though multithread computations are not supported now.

When a new agent is created, it downloads information from the scene. Using this information, the agent compares its current state to its goals and performs actions to improve it. Every time situation is changed, the agent revises current state and performs corresponding actions if necessary. Agent can also act proactively, not waiting for changes in the scene.

After analysis of scene, agent searches for candidates to conduct negotiations with (this is called pre-matching), according to its parameters. At this stage various heuristics can be applied to cut decision making space, e.g. search for journeys only within given radius and time interval from order windows. These heuristics can depend on agent’s state – the more satisfied agents are more passive, narrowing their search only to most promising candidates.

Although agents are not limited to any behavioural pattern, in i-Scheduler they usually act as a resource or a demand. The principal protocol used for negotiations is Iterated Contract Net Protocol [6]. Negotiations can be started by both demand and resource agents. After an agent finds a suitable decision, it may apply this decision or send confirmation for application to the plan owner. After that the changes are applied to reality scene and other agents may see them and act on them.

Options search via negotiations is a continuous process. Agent can be involved in negotiations with several partners simultaneously. Agent uses coordinator role to control negotiation threads. Each thread is represented by state-machine-like contractor role, which stores current state of negotiations and received messages. Negotiation threads may be assigned different priorities depending on their importance and state. If the situation changes and some negotiation threads become irrelevant, they may be cancelled at any moment. Vice versa, new negotiations may be started while other are still active. If partner does not respond for a certain period of time, agent automatically terminates negotiations by timeout. If agent finds an option which improves its current state, it immediately applies it. Otherwise, the option is stored in memory.

Stored options are used for coordinated decision making between agents and quick reallocation if situation changes (e.g. if transportation instruction agent is dropped from its journey, it may immediately accept the most profitable option from those stored in its memory). Agent constantly monitors and re-negotiates outdated options.

The decision making is based on microeconomics. Each agent evaluates options, based on real-life costs, associated with it. The model advances the savings model, introduced in Clarke and Wright Savings Algorithm [7]. Additionally, we use virtual penalties for poor consolidated journeys to make local attractors less stable. These penalties are not associated with any real costs, but help Transportation Instruction agents abandon poorly consolidated journeys, thus provoking reconstruction of poor schedule parts. It should be mentioned that this process is self-accelerating – the more agents abandon poor journey, the higher become penalties for remaining agents. Experiments showed that the approach gives several percent reduction in overall mileage.

7. SCALABILITY LIMITATIONS

Although it was stated previously that current performance was sufficient to handle all datasets we encountered, there are two fundamental scalability limitations:

- In-memory scheduling
- Single-thread scheduling

We perform scheduling, holding all information in memory. On largest datasets, this results in consuming about 1.5 Gigabytes of RAM. We are now investigating possibility of temporarily serializing passive agents, thus controlling the amount of in-memory instances. The other possible limitation is single thread computing, which prevents from using the power of parallel computations. We see the solution for this in using parallel swarms, coordinating decisions via a message queue, but this is still subject for investigation.

8. VIRTUAL MARKET FRAMEWORK

As result of Logistics i-Scheduler development, its core components were extracted into a separate framework called Virtual Market. This library accelerates development of distributed agent swarm applications. The key features of the framework are:

- Ontology as data model and storage
- Agent swarm
- Changes tracking and memory update mechanism
- Negotiation protocols framework
- Swarm coordination

Virtual Market serves as a basis for several Magenta projects in different domains.

9. CASE STUDIES

To receive proof that the software works, from the start of development a set of experiments with real life datasets was performed. The datasets included:
Currently, all datasets can be handled by the same engine by simply tuning decision logic parameters and supplying network data. This proves that the scheduling engine is a universal solution for most truck logistics companies.

9.1 Case Study 1: Medium Complexity Private Networks

At the first stage of development, we took a medium complexity transportation network as a playground for the new engine. We made a detailed analysis of the results produced for this network by the previous generation of i-Scheduler and we had feedback from client saying that the schedule was feasible and of sufficient quality. Consequently, it was a good benchmark for the new i-Scheduler.

The key objective for this experiment was the combination of primary and secondary deliveries and scheduling a network of medium complexity in batch and real time.

Network parameters:

- 200 orders per day
- 7 distribution centers
- Most orders are more than half truck load
- Shared fleet for primary and secondary deliveries
- 40 own fleet trucks
- Standard and double-decker trailers and rigid vans
- 12 third party carriers
- Dedicated primary moves
- 3 day planning cycle

The main requirements were effective backhauls and consolidations, vehicle capacity support, constraints stressing, driver breaks support and continuous scheduling. This experiment was described in [5]. Unlike batch optimizers, we imitated sequential arrival of orders with even distribution over time.

The results proved that distributed decision making as a new approach to scheduling worked efficiently. The results were very close to those delivered by previous generation of i-Scheduler. Creating the schedule took 4 minutes.

The first issue caused by distributed nature of multi-agent swarm was balancing agent activity. If a large number of agents decide to start negotiations simultaneously (each agent can conduct negotiations with hundreds of partners), the amount of messages to be processed per dispatcher cycle becomes too large. This had two consequences:

- Too large time between periods of activity for a certain agent. As a result a lot of messages are cancelled by timeout and system’s response time increases
- Combinatorial growth of amount of considered options. If agent tries to consider all options before making decision (negotiate with all candidates simultaneously), the “calculated options/decisions made” ratio becomes too large, which greatly slows down the system

Another issue was that while the system was busy improving schedule, processing time for new events grew significantly. To overcome the above-described shortcomings, a simple but effective adaptive balancing mechanism was introduced.

We introduced World Agent for managing the activities pool. In our terms, “activity” is a set of actions, performed by an agent during the process of negotiations with one partner. When an agent starts new negotiations, he captures the activity, increasing the counter of current activities. When negotiations are finished, activity is released.

At every dispatch cycle, World Agent calculates standard cycle duration based on the following characteristics:

- Computer speed tests result
- Amount of agents in swarm
- Optimal time for handling single agent on standard computer

If current dispatch cycle took longer than standard time, the amount of allowed simultaneous activities is increased, and vice versa. If the current activities count is below allowed activities level, it is allowed to start new activity.

For each start of negotiations, agent has to book an activity in the activities pool, otherwise he is forbidden to start negotiations. Unallocated agents (e.g. orders that have not found a suitable journey for themselves) have higher priority in occupying pool, than allocated ones.

Each agent also has its individual limit of activities, meaning that an agent can’t start negotiations with more than X partners simultaneously. To satisfy these limits, agents have to prioritize their negotiations and start from most prospective ones.

Experiment results showed significant – up to 10 times - decrease of processing time per order when this balancing approach was used, without loosing in schedule quality.

Apart from activity balancing, an anticipated result, natural for local and distributed decision-making, was dispersion of the results. With equal initial parameters and sequence of orders, the dispersion of schedule mileage between two launches was up to 2%. This is explained by non-determinism and unpredictability of agent activities sequence. Activities pool size depends on cycle time, which, consequently, depends on processor time, available for the scheduling thread. Obviously, this parameter may vary, thus affecting the list of agents who can successfully book activity on a certain cycle.

On one hand, this shows that i-Scheduler does not effectively cope with all local attractors. On the other hand, while the value of the dispersion is significant, it is not critical. Further works showed, that as algorithms become more tuned and powerful, the dispersion ratio shrinks. Currently, it does not exceed 1%.

9.2 Case Study 2: Large-Scale Complex Shared Network

The next dataset was a large-scale shared transportation network. This kind of network is considered the main target for the i-Scheduler, where its competitive advantages can bring most
The key objective of experiment was real-time planning in a highly complex network with crossdocks and dynamical routing.

Network characteristics:
- 4300 orders for three days
- Complex order profile: a lot of small-volume orders
- Few orders can be given away to TPC
- Majority of orders require complex planning – the price of a mistake is high
- 650 locations
- 3 cross docks
- 7 trailer swap locations
- 150 own fleet trucks, various types
- 25 third party carriers
- Carrier availability time
- Different pricing schemes

As for all other datasets, we imitated incremental arrival of orders. The required functionality included location availability windows support, effective backhauls and consolidation, various vehicle capacities, constraints stressing, continuous planning, crossdocking, dynamic routing and driver shifts planning.

Unfortunately, it was not possible to compare results with client schedule, as it contained a set of special requirements, which i-Scheduler did not support at that time. Only a rough comparison could be performed.

The first challenge in this case was the scale of the dataset. After splitting orders into primitive movements between waypoints (using crossdocks as intermediate points for consolidation), we received ~7200 transportation instructions (TI). Scheduling time exceeded 12 hours, with overall mileage 140 000 miles, which was a poor result. To improve the results, a dynamic clustering algorithm for transportation instructions was introduced. Similar instructions were merged into a single one. This step resulted in 2200 instructions, 4.5 hours scheduling time and 125 000 miles.

This showed that clustering (which performs a more global and less precise job) can be a very effective combination with multiagent optimization. Although these results were much better, they were still worse than anticipated.

We implemented the following dynamic routing algorithm. When a new order arrives, order agent selects a set of reasonable routes (in most cases two or three) through the chain of crossdocks between pick up and delivery locations. These routes are predefined manually and are stored as transportation patterns. For each route, agent considers its mileage and consolidation probability, and chooses the most appropriate route regarding current network state. After that, the order splits into transportation instructions, each represented by a separate TI Agent. Each TI has its own pick up and delivery windows.

Obviously, a leg of a journey cannot be started before the previous one is finished. This simple rule causes necessity to coordinate decisions between TI agents – they must select the pair of options, which give the most effective transportation for all concerned TIs. Chains of dependent TIs can theoretically be very long but normally there would not be more than three TIs per order.

The first approach – widely used in existing scheduling software – was artificially cutting TI time windows so that they did not intersect, but this had a severe impact on schedule quality, as a lot of effective consolidations were lost due to narrow TI windows.

Therefore, a more effective approach was applied. As stated above in the i-Scheduler Architecture section, each TI agent searches for options and stores them in its memory independently. When it finds a prospective option, it determines whether it contradicts with positions of any of its neighboring TIs. If this is the case, he asks the ‘neighbors’ for the cost of reallocation to free required timeslot. They select the best appropriate option from their memory and return difference between costs, compared to current state. If the overall cost of the new option is still better than its current position, TI agent applies this option and his neighbors reallocate to corresponding positions.

The approach only slightly increases computational complexity of the solution, as it uses pre-calculated options. At the same time, it enables finding effective balance between transportation options for different legs.

As a result scheduling time increased up to 6 hours, but overall mileage reached 115 000. The average truck utilization was 52%, which was already close to anticipated results, as trucks also have to carry back used equipment (package), which was not included in the dataset.

### 9.3 Case Study 3: Pilot Project With a 3PL Provider

Recently Magenta was working for 3 months with one of the biggest UK 3PL providers on a pilot project to prove that Magenta i-Scheduler is capable to handle their complex primary network.

The network serves one of major national UK retailers, who has two types of deliveries. Primary deliveries run from suppliers – mainly manufacturers of foods and groceries – to distribution centres – depots, where product is either prepared for store delivery, or is consolidated and sent to other depots. Secondary deliveries run from depots to retail stores. Deliveries to stores are made in two cycles – morning and evening. Deliveries into depots therefore also have to be done in two cycles which differ by product range, delivery dynamics and some other factors.

The client was interested in handling primary network movements and crossdocking with multi-agent technology. Secondary deliveries are mostly fixed and do not require crossdocking, that is why the operator was less interested in handling secondary network. Below are the parameters of primary network:

- 200+ supplier locations (pick up locations)
- 9 RDC (drop locations)
- 7 consolidation points
- 20 trailer swap locations
- ~1800 orders per day
- 24x7 operations
- Driver shifts comply with general EU regulations
The variety of operations including trailer swaps, direct deliveries, en route collections and trunking through the network

Several types of equipment

Third party carriers and secondary network vehicles are widely used for effective consolidation and backhauls

Different types of vehicles are used in the network to support delivery of product with various constraints

Both RDC and consolidation points can be used for crossdocking. At the moment, scheduling is performed manually. Previously the company tried to implement software packages to support primary deliveries, but these projects faced two major issues:

- Extremely short time to deliver for goods, moved in the network, exceeded software capabilities
- Software was not able to support routing via crossdocks and trailer swap points

For half of orders there are only approximately 4 hours for creating transportation plan, for the other half this time usually does not exceed 8 hours. Such short timeline allows keeping stock levels at distribution centres at minimum. Crossdocking implies complex dependencies between schedule parts and great variety of transportation options.

The logistics operator performed internal analysis of existing business processes coming to two main conclusions:

- Manual process is close to transactional. Under time pressure, human planners do not consider all possible options to find more optimal transportation plan but tend to use the simplest decisions – which often means running additional route. Consequently, transportation plans are not always cost-efficient. Time pressure is also often the reason behind mistakes in plans.
- The ability to handle network is close to its limit. Network complexity becomes a limiting factor for business growth.

To manage the network, operator uses a fixed schedule, which is revised once a month. In everyday operations, a team of 5 planners matches real orders into this network, adjusting it to order flow fluctuations. About 80% of the orders match the network rather seamlessly, while the remaining 20% may present difficulties for the planners.

The fixed schedule revealed a very interesting opportunity for multiagent approach. Fixed schedule provides an excellent starting point for multi-agent optimization, allowing agents to use the power of local optimization taking into consideration all the complexity of network, and at the same time start from basic solution, which is close to optimal schedule for larger part of the order flow. The results of the pilot project proved effectiveness of this approach. Detailed analysis showed that given correct data, the scheduler can achieve results, which outperform those, achieved by human planners.

The main challenge of the pilot project was inconsistency of provided data, as a result of manual scheduling process. We also revealed some logical differences related to order handling, and minor algorithm weaknesses, which affected the result. Still, the prototype was considered successful, although the impact of the limiting factors prevented the software from operating at an optimal level. The main success factor is that Magenta i-Scheduler was able to support key considerations of cross docking and trailer swaps, reflecting real world attributes of the network.

As result, the operator agreed to go forward with Magenta and start next step of development of the production system.

10. OTHER MULTIAGENT TRUCK SCHEDULING SYSTEMS

The most advanced multiagent scheduling system at the moment is Whitestein LS/ATN solution [8]. The system has already proven benefits in production system for ABX Logistics GmbH. Unfortunately, we have only limited information about the system, so it is impossible to make detailed comparison. The system copes with large transportation network and supports incremental scheduling. Yet the system doesn’t support cross docking operations. Whitestein uses a combination of centralized and distributed approaches, while Magenta uses fully distributed decision making approach. Further investigation of LS/ATN solution is on the list of our next steps.

11. RELATED WORKS

Multi-agent systems in logistics is a new challenging paradigm of solving optimization problems in real time. Still currently, many of multi-agent researches are focused on ideas of classical constraint programming [9], cooperative scheduling [10], different variants of combinatorial auctions [11] and other more traditional approaches rather than on exploring new features and advantages of multi-agent systems based on swarm intelligence principles [12].

One of most important new waves of researches is based on the idea of micro-economics of agents’ decision making [13,14]. Virtual money helps agents to limit the number of considered options and reduce time of computations. Another very important idea is that modern companies need to have chance to re-negotiate previously made decisions, constantly adapting to new events. For example in [15] it was analytically shown that for competing logistics companies the usage of recommitment of current contract for another superior offer can help agents reach higher utility levels. It is clear that under conditions of uncertainty, the negotiations with ability to pay penalties help to cover lack of information about future opportunities.

These important principles were integrated in our first patent [16], which described virtual market of demand and resource agents. As result we replaced time-consuming combinatorial search by goal-driven agents which work as autonomous “business units” or “holons” [17]. Instead of fixed penalties, we use flexible compensations calculated in real time, which depend on market situation and help to restrict chains of negotiations. Today in our swarm-based approach agents can represent not only trucks and orders but also clients, journeys, fleet, cross-docks, drivers, etc. This gives a chance to cover complexity of modern businesses, introducing new types of agents, which represent interests of not only people, but also of all other physical or mental entities whose interests (not only constraints!) must be taken in consideration to provide high quality of scheduling results [18, 4].

This means significant step in developing complex adaptive systems [19,20] with full set of new phenomena of not-stable
equilibriums, catastrophes, bifurcations, oscillations and many other which can bring a lot of value, but require further investigations.

12. CONCLUSIONS

The described case studies show that the Logistics i-Scheduler is capable to handle a wide range of transportation networks, from small to large scale and with different types of operations. The architecture is flexible enough to continue increasing the functionality of scheduling engine.

New versions of i-Scheduler become more and more intelligent not because of specific pre-build “blocks” of reasoning but because of interaction of thousands agents with conflicting interests which solve problems by conflict-driven self-accelerating negotiations, implementing so called “emergent intelligence”.

It gives i-Scheduler a number of new features, which radically differ our approach from traditional software providing high non-linearity of system behaviour, pro-activity and non-determinism, sensitivity to history of events, evolutionary irreversibility, flexibility and reliability, high efficiency and self-restoring after failures. These new qualities of our software are the topic of separate paper.

The main focus of Magenta now is implementing full scale production multiagent scheduling system. Besides, we shall continue improving existing scheduling algorithms. Areas of further investigations include:

- Microeconomics model and overcoming local attractors
- Integration with other algorithms, like clustering and stochastic algorithms
- Multithread / multi-server (multi-swarm) scheduling
- Support of additional logistics requirements (e.g. location availability schedule)
- Integration with other parts of supply chain, e.g. primary and secondary networks, warehouses.

Multiagent scheduler can bring most valuable benefits for large-scale complex network. For such class of networks, Magenta Logistics i-Scheduler seems to be the only solution, capable to handle all the complexity and variety of operations.

13. REFERENCES


