COORDINATION OF SUPPLY WEBS BASED ON DISPOSITIVE PROTOCOLS

Tim Stockheim

JWG University Frankfurt
Chair of Economics esp. Information Systems
Mertonstr. 17
D-60054 Frankfurt (Germany)
phone: ++49 (0) 69 798 23318
fax: ++49 (0) 69 798 28585
stockheim@wiwi.uni-frankfurt.de

Michael Schwind

JWG University Frankfurt
Chair of Economics esp. Information Systems
schwind@wiwi.uni-frankfurt.de

Oliver Wendt

JWG University Frankfurt
Chair of Economics esp. Information Systems
wendt@wiwi.uni-frankfurt.de

Sven Grolik

JWG University Frankfurt
Chair of Economics esp. Information Systems
grolik@wiwi.uni-frankfurt.de

ABSTRACT

The focus of this paper is the design of a mechanism to help economic agents - either autonomously or cooperatively planning - to achieve Pareto-optimal allocation of resources via a completely decentralized coordination of a logistics network. Besides giving a classification and a short review of existing scheduling approaches capable for supply chain management, this article specifies and evaluates protocols employing time-depended price-functions. By performing simulations with the implemented protocol using well-known Job Shop Scheduling Problems as a benchmark we show the efficiency and feasibility of the designed mechanism. The approach enables each agent to exploit the external effects caused by resource constraints of its supply chain contractors by adapting its production planning. Additionally the systems capability to reconfigure itself in case of production resources failure is increased. The evaluation of the protocols concludes with a welfare analysis investigating the payoff distribution along the supply chain. Finally we conclude that future research on this topic should turn to learning agent systems to reduce communication costs.

1. SCHEDULING TECHNIQUES IN AGENT-BASED SUPPLY CHAIN MANAGEMENT

Modeling, automation and optimization of a Supply Chain (SC), like most problems in operations management, are characterized by the two dimensions *time* and *location*, as well as a *utility* function which depends on both. The vectors time and location themselves span two problem classes in the operations management domain, which are closely related to Supply Chain Management (SCM): the class of Routing Problems (RP) and the class of Scheduling Problems (SP). Traditionally, RP are rarely considered to be in the scope of SCM. Literature mainly focuses on the scheduling aspects of SCM and assumes the RP to be solved at a later stage of planning. SP mostly turn out to be computationally hard (NP-hard)[1] and SCM makes no exception in this context. Handling this difficulty in SCM has led to various approaches, which could be roughly assigned to the following classes:

Solution method:	**Properties of scheduler:**	**Core domains:**
Combinatorial Auctions	central	Game Theory, Ontologies
Bargaining Processes	distributed	Game Theory, Ontologies
Random Search	central, distributed	Stochastic Optimization
Knowledge Based Systems	central, distributed	Automated Reasoning, Ontologies
Learning Systems	central, distributed	Machine Learning

Despite the appearance of the auctioneer as a central scheduler within the first solution method, the decision for a distributed architecture as a Multi Agent System (MAS) should be clear in every other case for three reasons:
MAS can describe the SCM directly by transforming physical nodes in the Supply Chain topology into agents.
MAS are flexible with respect to the mobility of transported and transformed goods, e.g. software agents can be attached to the regarding good as module.
MAS represent distributed computing power, which is helpful for solving complex problems.

MAS could be considered as communities of small autonomous software programs which should have the following properties: rationality, pro- and interactivity [Wooldridge and Jennings, 1995; Franklin and Graesser, 1996]. The MAS itself behaves different compared to the plain sum of actions resulting from the individual rationality of its members, a phenomenon which is called emergence [Goldstein, 1999].

Bargaining Processes (BP):

CONTRACT NET [Davis and Smith, 1983] provided one of the first attempts to design an effective communication protocol that supports the co-ordination and control of distributed problem solving systems. It assigns the tasks by a distributed bilateral, step-by-step ask bid mechanism between "managers" and "contractors" and tries thereby to avoid redundant communication traffic. Sub protocols for special tasks could be defined between groups of agents to announce, bid and award tasks, using example adjacency, processing capability and current workload as selection criterion. [Sandholm, 1993] creates a formal model for the allocation process which is based on a marginal cost calculus of the task performance. [Rosenschein and Zlotkin, 1994; Durfee and Rosenschein, 1994;

[1] E.g. for the Job Shop Scheduling Problem see [Garey et al., 1976].

Rosenschein and Zlotkin, 1996] and [Sandholm and Lesser, 1995; Sandholm, 1996] introduce aspects of Game Theory into contracting mechanisms by questioning the assumption of collaboration. Depending on the aims of the agents [Rosenschein and Zlotkin, 1994] postulate three frameworks: Task Oriented Domains (TOD), State Oriented Domains (SOD) and Worth Oriented Domains (WOD).

The hiding of information, the pretending of decoy tasks and the lying about true intentions are further variants of applicable complications in agents' behavior. This extension is a consequence of the differing economic interests of the supply chain participants and the reason why traditional enterprise scheduling mechanisms cannot be easily applied to the SCM domain. Despite efforts are made in many bargaining approaches driven by Game Theory, mechanism design does not always provide a satisfying trade off for the conflict Pareto optimality vs. individual utility optimization.

[Sen and Durfee, 1996] enhance the CONTRACT NET protocol with different decision models for distributed scheduling bringing communication costs for the negotiation process into account. [Wooldrige and Jennings, 1999] present a model of cooperative problem solving, which is also based on CONTRACT NET. In this model agents try to recognise a potential of cooperation with respect to their own objectives. The process of coalition formation is described by multi-modal logic which represents the mental states of the agents.

[Sierra et al., 2000] introduce argumentation based negotiation for agents to overcome some shortcomings of Game Theoretic approaches. Mainly the fact that only offer and counter-offer on one issue characterise the negotiation space in CONTRACT NET is criticised. The negotiation space is therefore extended by allowing additional information exchange. This should stimulate a persuasion process.

The aforementioned examples only demonstrate a small scope of the range of existing negotiation mechanisms that can be employed to manage the scheduling of tasks in a SC.

A further crucial point in the SC engineering domain is the semantic design of the communication process. Creating sophisticated protocols is usually done by employing Ontologies.[2] An implementation of an ontology designated for agent communication is Knowledge Query and Manipulation Language (KQML) [Finin et al., 1994]. KQML is both a message format and a message handling protocol to support knowledge sharing among agents. KQML focuses on an extensible set of performatives, which defines permissible "speech acts". Negotiation processes like those utilised in CONTRACT NET have been modelled in KQML.

Knowledge Interchange Formalism (KIF) [Genesereth, 1998] represents a subset of KQML. KIF is based on first order logic and includes both a specification of syntax for the language as well as a specification for semantics. KIF is often used to support translation from one content language to another between two agents.

With the Coordination Language (COOL) [Barbuceanu and Fox, 1995; Barbuceanu and Fox, 1996] present a subset of KQML designed for the description of co-ordination processes. The models of co-ordination involving speech acts are described with a Finite State Machine (FSM). Including a set of rules that is matched to the system states, the MAS communicating in COOL is enabled to meet decisions about resource allocation. The application, which is especially engineered for SCs, performs well in contracting and scheduling for larger networks, even for sophisticated rule sets.

Combinatorial Auctions (CA):

One way to resolve the strong complementarities constructing a SC is to auction the resources offered and requested by the participants, employing mechanisms which take account of them. Such auctions are called Combinatorial Auctions. In CA, agents place all or nothing bids for bundles of goods. The

[2] See [Grüninger and Uschold, 1996] for a comprehensive introduction.

auction computes high quality allocation of bundles, ensuring that the agents do not receive undesirable partial bundles. Although the general problem of winner determination is NP-hard [Rothkopf et al., 1998] the problem could be solved relatively quickly for larger allocation problems with mixed-integer-linear programming [Anderson et al., 2000].

[Wellman, 1993] creates Market Oriented Programming (MOP) as a base to solve the Task Allocation Problem (TAP) associated with distributed problem solving. The tasks are scheduled by an auction algorithm called WALRAS [Cheng and Wellman, 1995], which computes an equilibrium price for the bids (offers) of the consumers (producers) coupled to a utility- (production-) function. Wellman applies the MOP to a transportation network to calculate the cheapest maximum flow solution. [Wellman and Walsh, 1998; Wellman et al., 2001] try to develop auction protocols, which can handle CAs in the MOP-framework. They inquire Simple Ascending Auction Protocols (SAAP) and a Generalized Vickery Auction Protocols (GVAP) recognizing that none of both can guarantee an optimal solution for decentralized scheduling.

Based on TODs introduced in [Rosenschein and Zlotkin, 1994], [Walsh et al., 1998; Walsh and Wellman, 1999] present the concept of Task Dependency Networks (TDN) to facilitate the modeling of interdependent tasks in a SC. One of the crucial points for the design of auction mechanisms is, as already discussed for BPs, the question whether the agents act strategically or not. So far auction models relied on truth-telling agents. In [Walsh et al., 2000] examine a one-shot CA-mechanism with strategic bidding policies, computed by Monte-Carlo simulations. The protocol avoids some difficulties of the coordination mechanism by allowing de-commitment. While non-strategic bidding leads always to the construction of a stable SC-network with almost optimal allocations, strategic-bidding can augment the gains of the suppliers, including the imperfection of preventing a stable network for bidding margins near zero. A comprehensive and detailed explanation of the material presented above can be found in [Walsh, 2001].

Randomized Search Strategies (RSS):

Randomized Search Strategies (RSS) are one way to circumvent the computational consequences of NP-completeness sacrificing some solution quality. They use random choice to guide themselves through the problem search space. The search path, however, is not a directionless random walk, moreover, knowledge gained from previous results is used and recombined with some randomizing features. Popular representatives of this techniques are Genetic Algorithms (GA) proposed by [Holland, 1975; Goldberg, 1989] and Simulated Annealing (SA) first described by [Kirkpatrick and Gelatt, 1983]. Combination of both methodologies presented by [Goldberg and Mahfoud, 1993; Wendt, 1994] outperform the plain approach in some problem domains, e.g. for the Traveling Salesman Problem (TSP). The first applications of GAs to scheduling problems were introduced by [Davis, 1985]. The main problem for constructing efficient heuristics lies in finding the right representation of the tasks and resources on the alleles. In the mid 90s this problem was solved by reusing encoding patterns employed for sequencing problems, mainly the TSP [Fang et al., 1993; Husbands, 1994; Singh and Youssef, 1996; Yamada and Nakano, 1997; Lin et al., 1997a]. Parallel GA applications for the Job-Shop Scheduling Problem (JSSP) where examined by [Lin et al., 1997b] and obtained a more-than-linear acceleration compared to the single GA approach, as well as a better solution quality.

[Aarts et al., 1988] examined the applicability of SA to scheduling [Sadeh and Nakakuki, 1996; Sadeh et al., 1997]. A hybrid approach called Genetic Simulated Annealing is used by [Shroff et al., 1996]. It combines classical crossover operators with Simulated Annealing's acceptance probabilities, thus allowing for "longer jumps" within the search space than Cooperative Simulated Annealing [König and Wendt, 2000], which often gets trapped in local optima of the "rugged" fitness landscapes spanned by purely local search operators. In [Yamada and Nakano, 1996] the role of the "escape" operator is successfully played by a repair operator based on the Shifting Bottleneck Algorithm, which is used whenever a new local neighbor solution gets rejected by the SA.

With **A**gent **N**etwork for **T**ask **S**cheduler (ANTS) [Sauter and Parunak, 1999] present an **A**gent **B**ased **M**arket **A**rchitecture (ABMA) using a modified CONTRACT NET protocol for communication. Task scheduling is handled by unit-process- and part-brokers employing a function called **D**ensity-based **E**mergent **S**cheduling **K**ernel (DESK) which represents the ratio between the minimal resource commitment time and the time span available in the overall production schedule for every part. Minimizing the aggregate workload for all resources is objective of shifting tasks within available time span. If a conflict is not resolvable a renegotiation of a part of the schedule is necessary.

Within the ABMA MAGNET (**M**ulti **Ag**ent **Ne**gotiation **T**estbed) [Collins et al., 1997; Collins et al., 1999; Collins et al., 2000] create a hybrid method between CA and SA which allows multi period negotiations of customer and supplier agents within the framework of a market place. The customer agent builds feasible plans for the production process based on his operational requirements with the Critical Path Method (CPM) and places these offers. Supplier agents respond with bids for the composite tasks, or parts of them, if they are willing to give a discount for breaking the structure. The customer agent selects the best bid bundle offered by the suppliers by means of an SA evaluation algorithm. Re-negotiation is allowed, and if a suitable solution is found the market manager confirms the commitment and supervises the execution.

In the MASCOT (**M**ulti **A**gent **S**upply Chain **Co**ordination **T**ool) environment [Sadeh and Fox, 1998; Kjenstadt, 1998; Sadeh et al., 1999] use coordination agents to meet optimal planning decisions for the suppliers and customers in a production process. The decisions on the low level tier of the architecture are made by employing a stochastic optimization tool named MICRO-BOSS (**Micro Bo**ttleneck **S**cheduling **S**ystem) [Sadeh, 1991; Sadeh, 1994]. The micro-opportunistic scheduler propagates the resource constraints and apparent cost for a proposed schedule to all agents and applies a conflict resolution technique in case of collisions. Critical activities are determined for the resources and then jobs are assigned by a least cost approach. The process is iterated until a feasible solution emerges.

Knowledge Based Systems (KBS):

Knowledge Based Systems (KBS) for scheduling tasks can be considered as hybrid systems. For one thing they make use of a Blackboard Architecture (BA) [Hayes-Roth, 1985], for another they rely on RSS.

A blackboard is an information processing structure composed of several cooperating Knowledge Sources (KS): each containing any kind of algorithm, rules, data, and so forth, a separate Control Element (CE): determining the order in which the KS are executed, and the blackboard itself: the locus of communication and global memory.

[Hildum, 1994; Neimann et al., 1994] outlines a blackboard hierarchy with a scheduler blackboard containing schedule-, resource- and goal-KSs on the one hand, and a simulator blackboard embedding a world-, events-, application data-blackboard on the other hand. Triggered through events and determined by teamwork of the remaining KS using implicit rules, the blackboard system applies different RSS implemented in the scheduler's board (based on MICRO-BOSS) until a feasible plan for resource allocation emerges.

With Integrating Process Planning and Production Scheduling (IP3S) Architecture [Sadeh et al., 1996; Sadeh et al., 1998] present a blackboard-based shell which supports, concurrent development and dynamic revision of integrated process planning and production scheduling solutions, mixed initiative functionalities (which allows users the creation and testing of “what-if scenarios”), workflow management functionalities, and supply chain coordination functionalities.

[Grüninger and Fox, 1997] propose an ontology for enterprise integration in a cooperating MAS for integrated supply chain management. Activities of the intelligent agents are represented in first-order logic axioms, which are embedded in a micro-theory. By reasoning with these axioms, a satisfying propagation of scheduling constraints is achieved.

Learning Systems (LS):

One way to augment system performance of distributed scheduling systems in SCs is to introduce learning in the agents' behaviour. In the domain of Machine Learning (ML), Reinforcement Learning (RL) is an encouraging approach [Sutton and Barto, 1998]. RL is derived as a hybrid between Dynamic Programming and Monte-Carlo simulation. The advantage of this optimization method lies in the fact that the search for optimal decision sweeps only over regions in the state space, which could be considered as relevant. Mostly the RL system is implemented employing Artificial Neural Networks (ANNs), Principle Component Analysis (PCA) and GA Selection (GAS) for further state space compression.

[Zhang, 1996; Zhang and Dietterich, 1995] apply RL to learn domain-specific heuristics for job shop scheduling. A repair based scheduler starts with a critical-path schedule and incrementally repairs constraint violations with the goal of finding a short conflict-free schedule. The RL-algorithm is allied to train a NN to learn a heuristic evaluation function over states. This evaluation function is used by a one–step lookahead search to find good solutions to new scheduling problems. The performance of the system is better than the best known non-learning search procedure based on SA [Zhang and Dietterich, 2000].

[Miagkikh and Punch, 1999] propose a distributed system of RL-agents to solve combinatorial optimization problems. The system performs local search encoding information by RL and global search through the emergent action of the agent population.

[Arai et al., 2000] use a population of RL-agents employing a modified RL method called Profit Sharing (PS). PS uses trial and error experiences and reinforces effective rules instead of estimating values using the sequential values in state space. Applied to a scheduling task PS outpaces the traditional RL strategy.

[Zeng and Sycara, 1996; Zeng and Sycara, 1997] show an alternative approach to learning agents in the negotiation domain. They present a sequential decision making model of negotiation called Baazar. The model learns by a Bayesian belief update process. The model addresses complicated issues such as multi-issue and multi-criteria negotiation.

COORDINATION OF SUPPLY WEBS BY SOFTWARE AGENTS

The area of research described here clearly funds in the randomized search strategies domain. The following section contrasts different processes/protocols for collaborative rescheduling in supply webs. Along with the description of the three different protocols it compares their characteristics with respect to efficiency of coordination and communication overhead.

Figure 1 illustrates a rescheduling protocol for an agent not having access to any information except for time and price information agreed upon in the current contracts with its customers and suppliers. Each agent successively tries to improve its schedule by randomly selecting new start times for its tasks and then requesting up-to-date price information from both predecessors and successors in the supply web. Whenever this step turns out to be profitable (compared to the status quo), both contracts get updated.

Task rescheduling based on price information for a single point in time

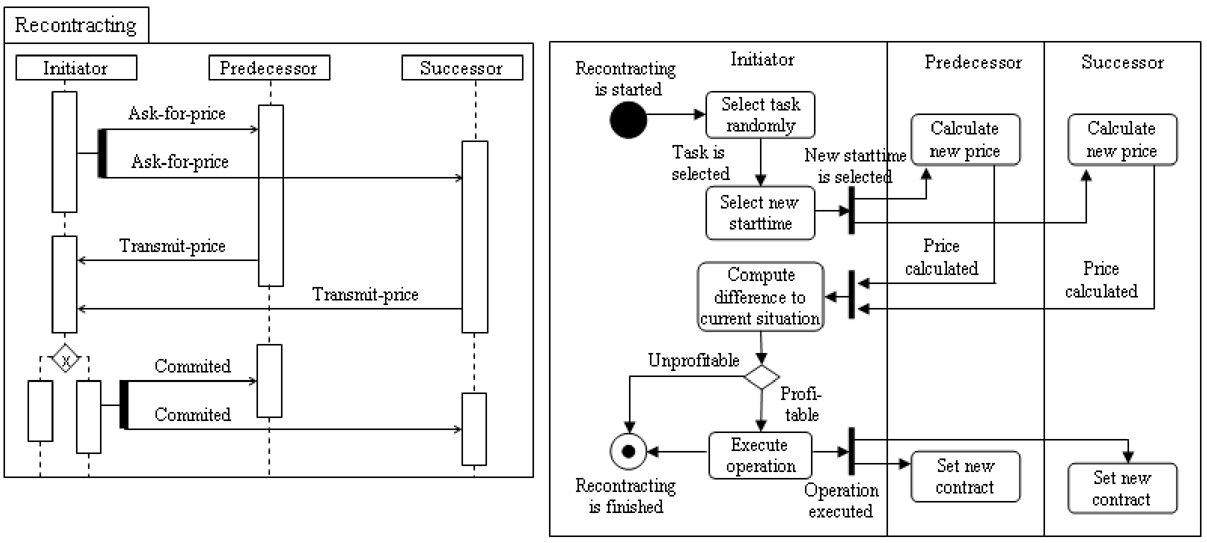

Fig. 1 Simple rescheduling protocol based on single-point price information (AUML[3])

Task rescheduling based on price functions

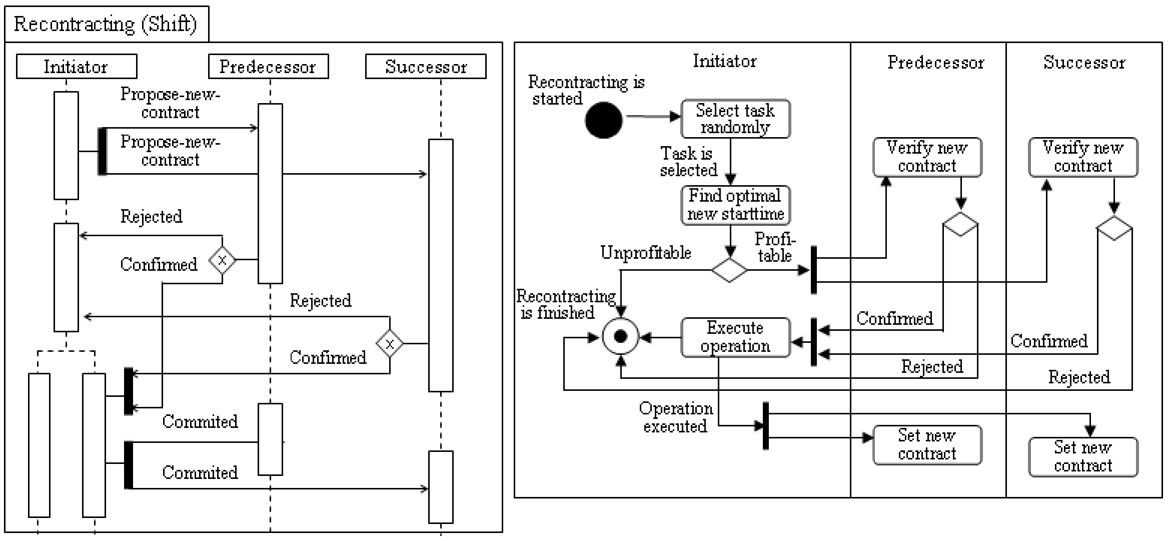

Fig. 2 Rescheduling based on price vector information

In order to reduce communication efforts and the number of rejected and therefore wasted rescheduling steps, agents may proactively propagate price vectors for all relevant points in time the service might presumably be contracted for. Figure 2 illustrates the rescheduling activity: Since every agent may use the stored price vectors received from his supplier or customers to find a tentative answer to the question whether the partner would agree to a specific rescheduling proposal, there is no

[3] To better illustrate the process we choose the Agent Unified Modelling Language (AUML). To get further information we suggest reading the following papers [Odell et al., 2001; Bauer, 1999] or refer to the website http://www.auml.org.

need to communicate a proposal until a point in time has been found that makes one's partners better off, too. However (due to the asynchronous planning of all partners in the supply web) the available "copy" of the partners' price vector might turn out to be outdated, whenever the actual rescheduling is proposed, thus making it necessary to reconfirm before the actual rescheduling occurs (which is done by simply transmitting a copy of the regarding contract time and price).

Propagation of price vectors

Figure 3 shows a randomly generated schedule needing 57 time units to completion, which is still far above the minimal schedule that could be generated if (and only if) all information were available to a central planning agent. In this example every agent plans the execution of four different tasks, each of them being part of four different supply chains (indicated by different textures). With the exception of the first agent of each supply chain, each agent has contracted the delivery of his product with his successor in the respective supply chain[4]. Each contract is defined by a pair (t; p) agreed upon by the two parties. Both the supplier and the customer may propose a new pair of delivery time and price which may be accepted or rejected by the other party. Whenever it gets accepted, the requesting agent makes the final decision whether or not the old contract gets replaced by the new one[5].

The first number shown at the tasks represents the point in time when the product will be supplied by the predecessor, the second number represents the planned start time of task execution[6].

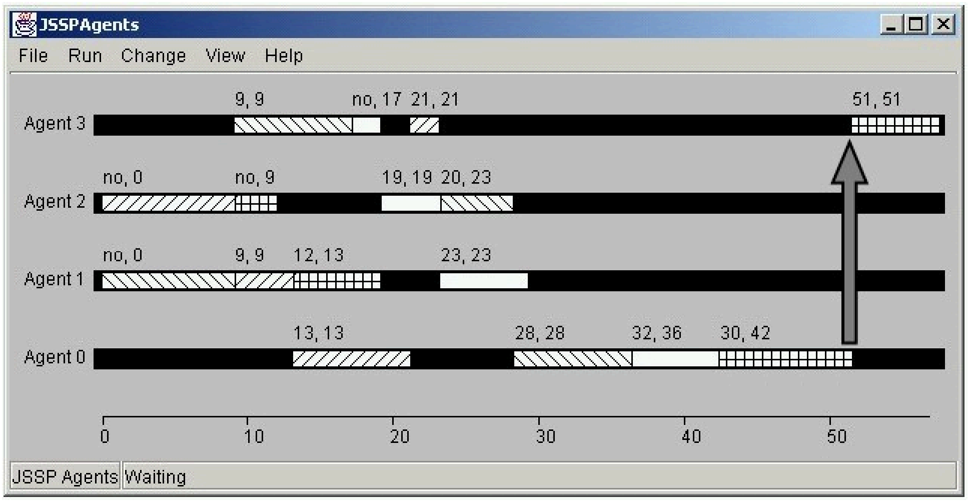

Fig. 3 Initial Schedule with current contract times and planned production times

Agent 0 and agent 3 did contract a delivery (for the "checkered" supply chain) at time 51 (refer to the arrow). At the start of our simulation, all contracts are arbitrarily[7] fixed at a price of 1000 monetary units (MU). Relating to this contract, agent 0 now communicates a (time-based) price function to agent 3, indicating his willingness to *charge* more or less than the agreed price for an earlier or later delivery. Since an earlier delivery is not possible, no values are communicated (i.e. the price is

---

[4] The last agent contracts with a "consumer" (not shown in the figure), who needs all four products and whose utility is determined by the arrival of the latest delivery.

[5] This allows for asynchronously requesting multiple binding offers and then substituting all relevant contracts in one single rescheduling step.

[6] The "no" at the first task of a supply chain indicates, that there is no predecessor.

[7] Since for the evaluation of any replanning activities, only the agent's relative profit or loss are relevant, this assumption does not pose any restrictions to generality.

"infinitely high"), for a delayed delivery agent 0 is not willing to offer a discount in this stage of the negotiation process, i.e. his prices are the same as for current contract time 51. All suppliers communicate their time-based price vectors to their customers while the customers communicate their respective willingness to *pay* more or less for early or late delivery in the analogue way. The upper diagram of figure 4 shows the resulting time-based offers (prices of the supplier agent 3 relating to the "checkered" supply chain). The vertical line relates to the agreed delivery time specified in the corresponding contract.

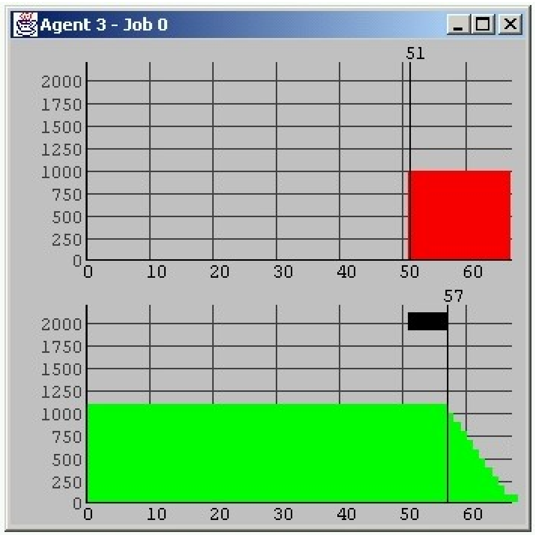

Fig. 4 Time-based supply and demand prices communicated to agent 3 (for the "checkered" process)

The lower part of figure 4 shows the demand side, communicated to agent 3 by the final "consumer", being the terminal to all four supply chains. The vertical line indicates the agreed delivery at time 57. For each time unit of delay will only be accepted by the consumer for a rebate of 100 MU. A speedup of one time unit, on the other hand, pays off by an increase of the customers' willingness to pay[8] to a total of 1100 MU.

Since we abstract from storage cost in this first model, all time-based price functions are monotonous, since no customer disagrees to an earlier and no supplier to a later delivery.

Agent 3's current schedule permits him to shift the checkered task to an earlier point of time (up to 28 time units), but for an earlier delivery of his input from agent 0 he would only be willing to pay more if he receives (at least) the same amount form the consumer for an earlier contract.

This means that every agent combines the time-based demand function received from his customer(s) with information concerning his current planning situation to derive his own time-based demand function he communicates to his supplier(s). This way the demand price functions propagate from the end of the supply web (the consumer) to the beginning, while, the time-based price functions for the supply propagate vice versa, i.e. from the manufacturer of the first raw materials to the consumer. Of course, the rationale that an agent uses to "blend" his own local information on rescheduling opportunities into this propagation process ultimately determines his profit (and the profit of other agents).

[8] The consumer agent's willingness to pay is exogenous to the model and thus will not change during the renegotiation process.

Simultaneous Rescheduling of two tasks by a single agent

As we pointed out above, propagating time-based price functions should help to reduce communication overhead for unprofitable offers. On the other hand, running a complete local optimization of the internal rescheduling opportunities and then trying to implement this schedule by simultaneously renegotiating all contracts did not yield acceptable results either.

As a "compromise" we therefore implemented a Shift&Swap protocol, that randomly (with a given probability) decides either to try shifting a single task (as outlined in figure 2) or, alternatively, to swap the order of two adjacent tasks.

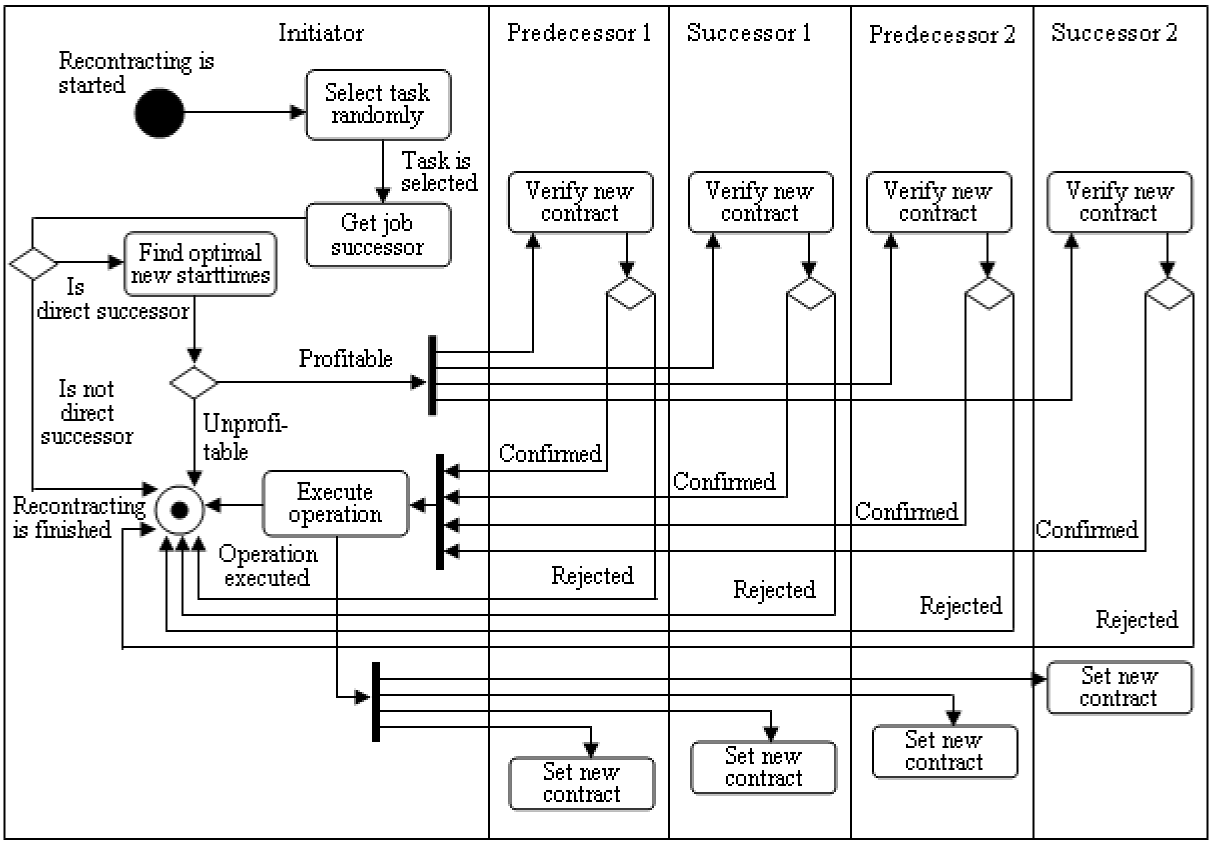

Fig. 5 Shift&Swap protocol

Assume that agent 0 randomly selects the white task (refer to figure 3) for an operation. Unfortunately, this task is completely stuck in between its predecessor and successor, leaving no room for any shift. Alternatively, the agent can try to swap the task, i.e. he tries to reverse the order of two tasks' execution. The immediate interal successor task to the white is the checkered one, and the profitable effect of their swap may be derived from the current time-based price functions displayed in figure 6.

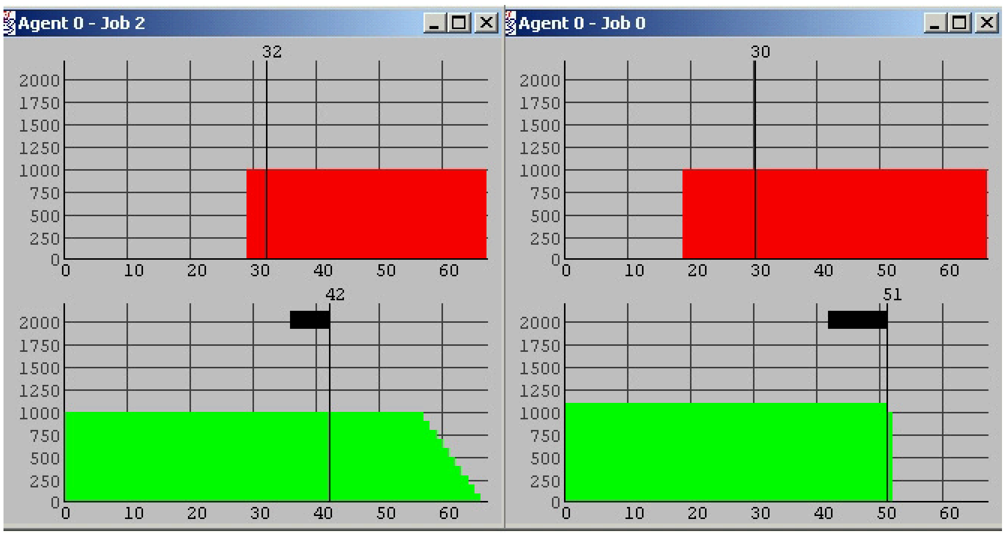

Fig. 6 Time-based supply and demand price functions communicated to agent 0 for the white task (job2, left) and the checkered task (job 0, right)

The checkered task (job 0) will now be terminated at time 45, yielding 1100 MU which is an excess profit of 100 MU. Of course, the white task (Job 2) will now be executed from time 45 to time 51 and thus its delivery to the customer gets delayed by nine time units from 42 to 51 but as we see from figure 6, delivery of the white task does no seem to be a bottleneck for the consumer since he is still willing to pay the full 1000 MU, leaving agent 0 with a total profit of 100 MU from this swap.

Note that for the checkered task agent 0 receives his input by time 30 and thus there is no need to renegotiate with the supply side for this swap. The same is true for the white task, since the delayed start of task execution could never lead to supply side problems (in the absence of storage cost). But in general, side effects for a maximum of four agents have to be considered to evaluate the profitability of a swap, two agents on the supply and two on the demand side. For a shift, there is only one agent involved on the demand an one on the supply side. Of course, this number increases, once we do introduce multiple sourcing.

WELFARE ANALYSIS FOR THE SUPPLY WEB

Applying the negotiation protocols introduced above to a multi-agent system (six production agents and one consumption agent) results in the average payoff distribution depicted in figure 10 (right) given the decentralized version of the well-known Fisher-and-Thompson 6x6 problem [Fisher and Thompson, 1963] and 5,000 runs of 25.000 transitions each.

The respective agents receive disproportionate payoff shares resulting from renegotiation. While agents 4 and 6 get most of the profit the payoff for agents 1 and 2 remains significantly below average.

How can the welfare distribution be explained? One approach is identifying the agents' average position within the supply web. Comparing the agents' orders to their payoffs within the course of the Fisher-and-Thompson 6x6 problem reveals that especially those agents with low net earnings do not supply any task directly to the consumer.

To support this hypothesis individual payoffs for all tasks were determined using numerical simulations. The simulation pattern is as follows: First, all positions of tasks within the process chain are assigned increasing ordinal numbers. Thus, all tasks at the source are numbered "1" and the tasks directly delivered to the consumer agent are numbered "6". Now, the payoffs of all positions (1 to 6) (for every agent after a simulation run consisting of 25,000 steps) are cumulated. These payoffs are determined as the difference between supply and delivery price for the respective task. These results in the histogram illustrated by figure 7 (left) showing aggregate return for every process step. Noticeably, significant parts of overall profits are produced by direct sales to the consumer while considerably smaller amounts are used to pay agents with an "upstream position" in the supply chain.

To anticipate individual profits we use a simple exponential estimator for the distribution. Here, determining an estimation results in $p^{2,873}$ with p denoting process position[9].

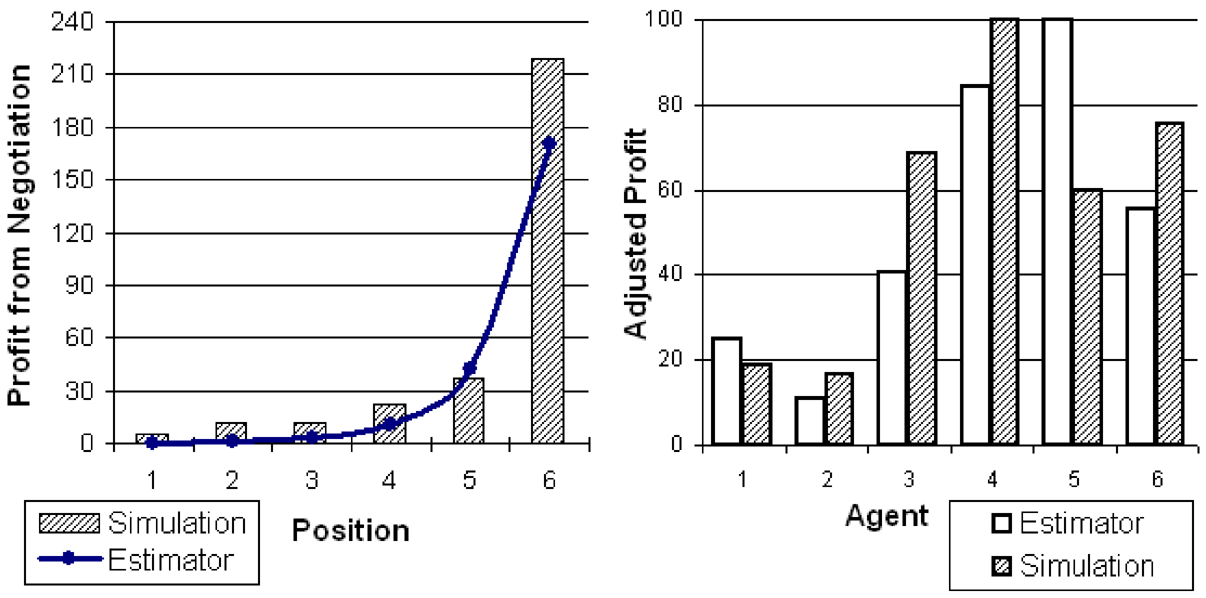

Fig. 7: Aggregated payoffs and estimator for each job step (left), Aggregated payoffs and estimator for each agent (right)

Figure 7 (right) depicts the expected profit distribution when the estimator is applied: Adding the estimates according to the respective sub processes for each agent results in an overall estimation concerning the profit situation of an agent with regard to all his production steps. The maximum profit (made by agent 4) was normalized to 100. The assumption that higher payoffs indicate key positions in the supply chain implies that the dominating players are positioned "downstream".

CONCLUSION

Most scheduling systems for supply chain optimization rely on Randomized Search Strategies (RSS) employing price information valid only for a single point in time. The introduction of time dependency via dispositive protocols in our RSS driven approach enables agents to exploit the external effects caused by resource constraints of its supply chain contractors by adapting its production planning. The assumption of individual rationality holds for the system, because the agents act only for their own behalf. This property gives us the chance to analyze the payoff distribution along the supply chain, indicating an exponential growth "downstream". Besides higher system flexibility if rescheduling is required in case of production-failure, time dependant price functions seem to be a suitable medium for the integration of detailed information in the agents' communication process. Related work on the scheduling domain shows that involving learning in agent's behavior is a promising approach to improve system performance and should therefore be in the focus of our future work.

[9] The confidence interval for the exponent (95 percent) is (2.85365; 2.89344).

REFERENCES

Aarts, E.; Vaessens, R. and Lenstra J. (1988). Job-Shop Scheduling by Simulated Annealing, In: Operations Research 40, pp. 113-125.

Andersson, A.; Tenhunen, M. and Ygge, F. (2000). Integer Programming for Combinatorial Auction Winner Determination, In: ICMAS'00, Uppsala, Sweden.

Arai, S.; Sycara, K. and Payne T. (2000). Experience-based Reinforcement Learning to Acquire Effective Behavior in a Multi-agent Domain, In: Proceedings of the 6th Pacific Rim International Conference on Artificial Intelligence.

Barbuceanu, M. and Fox, M. (1995). COOL: A Language for Describing Coordination in Multi Agent Systems, In: First International Conference on Multi-Agents Systems, San Francisco, California.

Barbuceanu, M. and Fox, M. (1996). Coordinating Multiple Agents in the Supply Chain, In: Proceedings of the Fifth Workshops on Enabling Technology for Collaborative Enterprises, WET ICE'96, IEEE Computer Society Press, pp. 134-141.

Bauer, B. (1999). Extending UML for the Specification of Agent Interaction Protocols, Submitted for the 6th Call for Proposal of FIPA.

Cheng, J. and Wellman, M. (1995). The WALRAS Algorithm: A Convergent Distributed Implementation of General Equilibrium Outcomes, In: Computational Economics.

Collins, J.; Jamison, S.; Mobasher, B. and Gini, M. (1997). A Market Architecture for Multi-Agent Contracting, In: Proc. of the 2d Int'l Conf. on Autonomous Agents, pp. 285-292.

Collins, J.; Sundareswara, R.; Tsvetovat, M.; Gini, M. and Mobasher, B. (1999). Search Strategies for Bid Selection in Multi-Agent Contracting, In: Proc. of IJCAI-99 Workshop on Agent-mediated Electronic Commerce (AmEC-99),

Collins, J.; Sundareswara, R.; Tsvetovat, M.; Gini, M. and Mobasher, B. (2000). Multi Agent Contracting for Supply Chain Management, Technical Report 00-010, University of Minnesota.

Davis, L. (1985). Job-shop scheduling with genetic algorithms. In: Grefenstette, J.(Ed.), Proc. Intl. Conference on GAs, Lawrence Erlbaum, pp. 136-140.

Davis, R. and Smith, R. (1983). Negotiation as a Metaphor for Distributed Problem Solving, In: Artificial Intelligence 20, pp. 63-109.

Durfee, E. and Rosenschein, J. (1994). Distributed Problem Solving and Multi-Agent Systems: Comparisons and Examples, The Thirteenth International Distributed Artificial Intelligence, pp. 94-104.

Fang, H.; Ross, P. and Corne, D. (1993). A Promising Genetic Algorithm Approach to Job-Shop Scheduling, Rescheduling, and Open Shop Scheduling Problems, In: S. Forrest (ed.), Proc. of the 5th International Conference on GA, pp. 375-382. Morgan Kaufmann.

Finin, T.; Fritzson, R.; McKay, D. and Entire R. (1994). KQML as an Agent Communication Language", In: Proc. of the 3rd Intern. Conf. on Information and Knowledge Management.

Fisher, H. and Thompson, G. (1963). Probabilistic Learning Combinations of Local Job-Shop Scheduling Rules, In: Industrial Scheduling, Muth, J. and Thompson, G. (Eds.), Englewood Cliffs, pp. 225-251, Prentice Hall.

Franklin S. and Graesser A. (1996). Is it an Agent, or just a Program? A Taxonomy for Autonomous Agents, In: 3rd Intern. Workshop on Agent Theories, Architectures, and Languages.

Garey, M.; Johnson, D. and Sethi, R. (1976). The Complexity of Flow Shop and Job Shop Scheduling, Math. Operations Research (1), pp. 117-129.

Genesereth, M. (1998). Knowledge Interchange Format: Draft proposed American National Standard, In: NCITS.T2/98-004.

Goldberg, D. (1989). Genetic Algorithms in Search, Optimization and Machine Learning, Addison Wesley.

Goldberg, D. and Mahfoud, S. (1993). Parallel Recombinative Simulated Annealing: A Genetic Algorithm.

Goldstein, J. (1999). Emergence as a Construct: History and Issues, In: Emergence (1).

Grüninger, M. and Fox, M. (1997). On Ontologies and Enterprise Modeling, In: International Conference on Enterprise Integration Modelling Technology 97, Springer.

Grüninger, M. and Uschold, M. (1996). Ontologies: Principles, Methods and Applications, In: Knowledge Engineering Review (11:2), pp. 93-136.

Hayes-Roth, B. (1985). A Blackboard Architecture for Control, In: Artificial Intelligence 26, pp. 251-321.

Hildum, D. (1994). Flexibility in a knowledge-based system for solving dynamic resource-constrained scheduling problems, Ph.D. Thesis, Massachusetts.

Holland, J. (1975). Adaptation in Natural and Artificial System: An Introductory Analysis with Applications to Biology, Control and AI, In: Univ. of Michigan Press, Ann Arbor MI.

Husbands, P. (1994): Genetic Algorithms for Scheduling, In: AISB Quarterly, No.89.

Kirkpatrick, S. and Gelatt, M. (1983). Optimization by Simulated Annealing, In: Science 220, pp. 671-680.

Kjenstadt D. (1998). Coordinated Supply Chain Scheduling, In: Ph.D. Thesis.

König, W. and Wendt, O. (2000). Kooperative Lokale Suche. Wann lohnt der Heterogenitätsverlust?, In: Jahnke, B. and Wall, F. (Eds.): IT-gestützte betriebswirtschaftliche Entscheidungsprozesse, Wiesbaden, pp. 63-86.

Lin, S.; Goodman, E. and Punch W. (1997). A Genetic Algorithm Approach to Dynamic Job-Shop Scheduling Problems, In: Proc. 7th International Conf. on Genetic Algorithms, Morgan Kaufmann Publishers, San Francisco, pp. 481-488.

Lin, S.; Goodman, E. and Punch W. (1997). Investigating Parallel Genetic Algorithms on Job Shop Scheduling Problems, In: Angeline P.; Reynolds R. and Eberhart R. (Eds.), 6th Internat. Conference on Evolutionary Programming, pp. 383-393.

Miagkikh, V. and Punch, W. (1999). An Approach to Solving Combinatorial Optimization Problems Using a Population of Reinforcement Learning Agents, In: Genetic and Evolutionary Computation Conf. (GECCO-99).

Neiman, D.; Hildum, D.; Lesser, V. and Sandhom, T. (1994). Exploiting Meta-Level Information in a Distributed Scheduling System, In: Proc. 12th Nat'l Conf. Artificial Intelligence, Seattle.

Odell, J.; Parunak, v. D. and Bauer, B. (2001). Representing Agent Interaction Protocols in UML, In: Agent-Oriented Software Engineering, Ciancarini, P. and Wooldridge, M. (Eds.), pp. 121-140, Springer, Berlin.

Rosenschein, J. and Zlotkin, G. (1994). Rules of Encounter, The MIT Press, Cambridge, Massachusetts.

Rosenschein, J. and Zlotkin, G. (1996). Mechanism for Automated Negotiation in State Oriented Domains, Journal of Artificial Intelligence Research, 5: pp. 163-238.

Rothkopf, M.; Pekec, A. and Harstad, R. (1998). Computationally Manageable Combinatorial Auctions, In: Management Science, 44(8): pp. 1131-1147.

Sadeh, N.; Hildum, D.; Laliberty, T.; Smith, S.; McA'Nulty, J. and Kjenstad, D. (1996). An Integrated Process-Planning and Production-Scheduling Shell For Agile Manufacturing, In: Proceedings of the IFIP WG5.7 Working Conference.

Sadeh, N.; Nakakuki, Y. and Thangiah, S. (1997). Learning to Recognize. (Un)promising Focussed Simulated Annealing Runs: Efficient Search Procedures for Job-Shop Scheduling and Vehicle Routing, In: Annals of Operations Research (75), pp. 189-208.

Sadeh, N.; Hildum, D.; Laliberty, T.; McA'Nulty, J.; Kjenstad, D. and Tseng, A. (1998). Blackboard Architecture for Integrating Process Planning and Production Scheduling, In: Concurrent Engineering: Research & Appl. (CERA), 6(2), pp. 88-100.

Sadeh, N.; Hildum, D.; Kjenstadt, D. and Tseng A. (1999). MASCOT: An Agent-Based Architecture for Coordinated Mixed-Initiative Supply Chain Planning and Scheduling, In: Proc. 3rd Intern. Conf. on Autonomous Ag.(Agents'99), Seattle.

Sadeh, N. (1991). Look-Ahead Techniques for Micro Opportunistic Job-Shop Scheduling, Ph.D. Thesis, Carnegie Mellon University.

Sadeh, N. (1994). Micro Opportunistic Job-Shop Scheduling: The Micro Boss Factory Scheduler. In: Zweben, M. and Fox M. (eds.) Intelligent Scheduling, Morgan Kaufmann, pp. 99-136.

Sadeh, N. and Fox, M. (1998). Variable and Value Ordering Heuristics for the Job Shop Scheduling Constraint Satisfaction Problem: A State-of-the-art Review of Job-Shop Scheduling Techniques, Artificial Intelligence 86(1), pp. 1-41.

Sadeh, N. and Nakakuki, Y. (1996). Focused Simulated Annealing Search – An Application to Job-Shop Scheduling, In: Annals of Operations Research (63), pp. 77-103.

Sandholm, T. (1993). An Implementation of the Contract Net Protocol Based on Marginal Cost Calculations, In: Proceedings of the Eleventh National Conference on Artificial Intelligence, pp. 256-262, Washington.

— First — Previous — Next — Last — Contents —

Sandholm, T.: Negotiation among Self-Interested Computationally Limited Agents, Dissertation, Amherst, MA, 1996

Sandholm, T. and Lesser, V. (1995). On Automated Contracting in Multi-enterprise Manufacturing, In: Proc. Improving Manufacturing Performance in a Distributed Enterprise: Advanced Systems and Tools, Edinburgh.

Sauter, J. and Parunak, v. D. (1999). ANTS in the Supply Chain, In: Workshop on Agent Based Decision Support for Managing the Internet-Enabled Supply Chain, Seattle.

Shroff, P.; Watson, D.; Flann, R. and Freund, R. (1996). Genetic Simulated Annealing for Scheduling Data-dependent Tasks in Heterogeneous Environments, In: 5th IEEE Heterogeneous Computing Workshop (HCW96).

Sierra, C.; Lomuscio, A.; Wooldrige, M.; Faratin, P. and Jennings, N. (2000). Automated Negotiation: Prospects, Methods and Challenges, In: Journal of Group Decision and Negotiation.

Singh, H. and Youssef, A. (1996). Mapping and Scheduling Heterogeneous Task Graphs Using Genetic Algorithms, In: 5th IEEE Heterogeneous Computing Workshop (HCW 96).

Sutton, R. and Barto, A (1998). Reinforcement Learning: an Introduction, MIT Press, Cambridge.

Walsh, W.; Wellman, M.; Wurman, P. and MacKie-Mason, J. (1998). Some Economics of Market-Based Distributed Scheduling, In: Eighteenth International Conference of Distributed Computing Systems, Michigan.

Walsh, W.; Wellman, M. and Ygge F. (2000). Combinatorial Auctions for Supply Chain Formation, In: ACM Conference on Electronic Commerce, Minneapolis.

Walsh, W. (2001). Market Protocols for Decentralized Supply Chain Formation, In: Ph.D. Thesis.

Walsh, W. and Wellman, M. (1999). Modeling Supply Chain Formation in Multiagent Systems, In: Agent Mediated Electronic Commerce (IJCAI Workshop), Michigan.

Wellman, M.; Walsh, W.; Wurman, P. and MacKie-Mason, J. (2001). Auction Protocols for Decentralized Scheduling, In: Games and Economic Behavior.

Wellman, M. (1993). A Market-Oriented Programming Environment and its Application to Distributed Multicommodity Flow Problems, In: J. of Artificial Intelligence Research (1).

Wellman, M. and Walsh, W. (1998). A Market Protocol for Decentralized Task Allocation, In: Revised and extended version of a paper presented at the Third International Conference on Multiagent Systems.

Wendt, O. (1994). Naturanaloge Verfahren zur approximativen Lösung Kombinatorischer Optimierungsprobleme, In: Ph.D. Thesis, Frankfurt.

Wooldrige, M. and Jennings, N. (1995). Intelligent Agents: Theory and Practice, The Knowledge Engineering Review.

Wooldrige, M. and Jennings, N. (1999). The Cooperative Problem Solving Process, In: Journal of Logic and Computation 9 (4) pp. 563-592.

Yamada, T. and Nakano, R. (1996). Job-Shop Scheduling by Simulated Annealing Combined with Deterministic Local Search, In: Kluwer Academic Publishers, pp. 237–248, MA, USA.

Zeng, D. and Sycara, K. (1996). Bayesian Learning in Negotiation, In: Working Notes for the AAAI Symposium on Adaptation, Co-evolution and Learning in Multiagent Systems, Stanford University, CA.

Zeng, D. and Sycara, K. (1997). Benefits of Learning in Negotiation, In: Proceedings of AAAI-97.

Zhang, W. (1996). Reinforcement Learning for Job-Shop Scheduling, In: Ph.D. Thesis, Oregon State University.

Zhang, W. and Dietterich, T. (1995). A Reinforcement Learning Approach to Job-shop Scheduling, In: Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence, San Francisco, USA.

Zhang, W. and Dietterich, T. (2000). Solving Combinatorial Optimization Tasks by Reinforcement Learning: A General Methodology Applied to Resource-Constrained Scheduling, In: Journal of Artificial Intelligence Research 1 (2000) 1-1.