MULTI-AGENT DYNAMIC SCHEDULING AND RE-SCHEDULING WITH GLOBAL TEMPORAL CONSTRAINTS

Joaquim Reis
Dpt. Ciências e Tecnologias de Informação, ISCTE
Avenida das Forças Armadas, 1600 Lisboa, Portugal
e-mail: Joaquim.Reis@iscte.pt

Nuno Mamede
Dpt. de Engenharia Informática, IST
Avenida Rovisco Pais, 1049-001 Lisboa, Portugal
e-mail: Nuno.Mamede@acm.org

Key words: Scheduling, Multi-Agent Systems, Supply-Chain Management.

Abstract: A co-ordination mechanism is proposed for multi-agent production-distribution co-operative scheduling problems, based on a purely temporal perspective. This mechanism is based on communication among pairs of client-supplier agents involved in the problem, and allows agents to locally perceive hard global temporal constraints. By exchanging limited specific information, the agents are able to recognise non over-constrained problems and, in that case, rule out non temporally-feasible solutions and establish an initial solution. The information is then used to guide re-scheduling and repair the initial solution and converge to a final one.

1 INTRODUCTION

This article describes an ongoing investigation that developed from work being published on the subject of multi-agent scheduling in production-distribution environments (see [Reis 1998], [Reis 1999a], [Reis 1999b], [Reis 1999c] and [Reis 2000]).

The specific logistics context of the supply-chain/Extended Enterprise (EE) [O'Neill 1996] is considered for short-term scheduling. The EE is usually assumed to be a kind of Virtual Organisation, or Virtual Enterprise, where the set of participant agents is relatively stable (for concepts, terminology and classification see [Camarinha-Matos 1999]).

Scheduling is the allocation of resources over time to perform a collection of tasks subject to temporal and capacity constraints. For a general introduction to classical/Operations Research (OR) approaches to scheduling problems see [Blazewicz 1994]. For Artificial Intelligence (AI) based approaches see [Zweben 1994].

Planning and co-ordination of logistics activities has been, in the areas of OR/Management Science, the subject of investigation since the sixties [Graves 1993]. Scheduling in this kind of environments has had, recently, a more dedicated attention (*e.g.*, see [Fox 1993], [Arnold 1997], [Kjenstad 1998] or [Rabelo 1998]).

The scheduling problems we consider are:

a) Highly dynamic - Solution development is an on-going process during which unforeseen events must always be accommodated for;
b) Multi-agent co-operative - Each agent has its own local perspective of the scheduling problem, can optimise its own scheduling preferences, but it must co-operate so that it won't invalidate a feasible scheduling solution.

The following sections present: the multi-agent scheduling environment (sec.2), the general approach proposed (sec.3), the initial scheduling step (sec.4) and the re-scheduling (sec.5), both from an individual agent perspective, and finally, future work and conclusions (sec.6). Secs. 4 and 5 are presented with examples based on simulations done.

2 THE SCHEDULING ENVIRONMENT

As scheduling resources are managed by individual, geographically decentralised and autonomous entities (enterprises, organisations) we adopted the AI Multi-Agent Systems paradigm (see, for instance, [ICMAS 1996] or [O'Hare 1996]). In the past work already referred, we have proposed a model of the EE/production-distribution based on an *agent network*, with each agent managing an aggregated scheduling resource (representing a production, a store, or a transportation resource) and linked through client-supplier relationships. A scheduling resource is just an individual node of a network of resources, termed a *physical network,* and accommodates the representation of the agent tasks scheduled and the available capacity along a certain scheduling temporal horizon. Ordinary network agents are termed *capacity agents*, because they are responsible for managing the capacity of a resource. A special *supervision agent*, with no resource, plays the role of an interface with the outside, and can introduce work (*i.e.*, new scheduling problems) in the agent network.

Among capacity agents communication is *local* in the agent network, which means that only pairs of client-supplier agents can communicate. This can occur by the exchange of messages of product request (client to supplier), request acceptance or rejection (supplier to client), re-scheduling request, re-scheduling acceptance, re-scheduling rejection and cancellation of a previously accepted request (both ways), and satisfaction of an accepted request (supplier to client, sent when the time of due-date comes). Request messages contain product, quantity of product and proposed due-date information. Some of the above types of messages convey also specific information for co-ordination of scheduling activity. The supervision agent can only communicate with agents playing the roles of retail and raw-material agents, *i.e.*, the agents located at the downstream and the upstream end of the agent network (these are pure clients and pure suppliers for the network) respectively.

At supervision level, a scheduling problem is defined by a *global product request from the outside* of the agent network (*i.e.*, a request of a final product of the network), the global due-date `RD`, and the global release date `DD`. These two dates are the limits of the scheduling temporal horizon and are considered the hard global temporal constraints of the problem. The supervision agent introduces a new scheduling problem by communicating a global product request from outside to the appropriate retail agent;[1] later, after the capacity agents have propagated among them, upstream the agent network, the necessary *local product requests*, the supervision agent will collect *global product requests to outside* from raw-material agents. A scheduling problem will cease to exist in the network when the time of the last of the local due-date comes, or if some local requests are rejected, or accepted and then cancelled (the supervision agent knows this as messages of satisfaction, rejection and cancellation will be propagated to retail and raw-material agents and then communicated to it).

In order to satisfy a product request from a client a capacity agent must schedule a task on its resource. A task needs a supply of one (in the case of a store or transportation agent), or one or more products (in the case of a production agent and depending on the components/materials for the task product). For those supplies the agent must send the appropriate product requests to the appropriate suppliers (and there is always a unique supplier for each supply product). The task consumes a non-changing amount of resource capacity during its temporal interval. The duration of the task depends primarily on the product, the quantity of product, the characteristics of the resource, and additional parameters. The details of these latter parameters are omitted here to allow simplicity of explanation, and we will assume a non-changing duration for all tasks.[2]

Although tasks are private to the agents (only the communication messages are perceived by more than one agent), we can view the set of tasks that some agents of the network schedule to satisfy a global product request from outside, as whole, as being a *network job* (see example in Figure 1). This is a just

[1] Because our networks can be multi-product, the set of products deliverable can depend on the retail agent.

[2] In our model, the duration of a task can be externally imposed, in the case of a store agent, and will depend also on the per time unit capacity amount invested, in the case of a production or transportation agent. Capacity constraints are represented by a value of maximum capacity, possibly varying along the temporal horizon. This value corresponds to a quantity of product for store agents, and to a rate (*i.e.*, quantity per time unit) for production and transportation agents.

an analogue of the concept of job used in classical production scheduling problems.

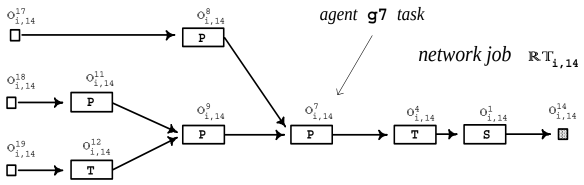

Figure 1- A network job. The task of an agent g_k for the ith global request to retail agent g_r is denoted by $\mathbb{O}^{k}_{i,r}$.

A *solution* for a scheduling problem is a set of product requests agreed by pairs of client-supplier agents and agent tasks necessary to satisfy the global product request given by the problem. A *feasible solution* has nor temporal nor capacity conflicts, *i.e.*, it respects both all temporal and all capacity constraints. For capacity constraints to be respected, no capacity over-allocation must occur with any task of the solution, in any agent, at any moment of the scheduling temporal horizon. For temporal constraints to be respected, all local product requests of the solution must fall within the global release date and global due-date of the problem; also, for each agent, the interval of its task must fall in between the due-date agreed for the client request and the latest of the due-dates agreed for the requests made to suppliers, and the latter due-date must precede the former. Temporal constraints can be summarised in the temporal precedence of specific points in time.

More details on the resources and the physical network are given in [Reis 1998], [Reis 1999b], and [Reis 1999c]; about the agent network and inter-agent communication see [Reis 1999a], also [Reis 1999b], and [Reis 2000].

3 THE PROPOSED APPROACH

Classically, scheduling is considered a difficult problem [Garey 1979]. Generally, solutions for a scheduling problem have to be searched for and the search space can be very large. Additionally, for a multi-agent scheduling problem, a part of the searching effort is invested in co-ordination of the agents involved which, in our case, means sharing information through message exchange. Message exchange is considered costly so, methods of pruning the search space for finding at least a first feasible solution with minimal co-ordination efforts are considered satisfactory.

In our approach we consider two special sets of solutions: the set of *time-feasible solutions*, which respect all temporal constraints, and the set of *resource-feasible solutions*, which respect all capacity constraints. As Figure 2 shows, the set of feasible solutions is the intersection of those two sets, *i.e.*, a feasible solution is one that is both time-feasible and resource-feasible. A problem is termed *temporally over-constrained* if the set of time-feasible solutions is empty, and is termed *resource over-constrained* if the set of resource-feasible solutions is empty. If a problem

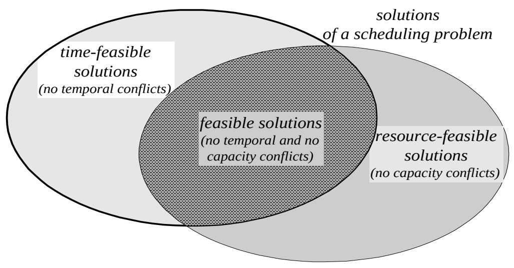

Figure 2- Solutions of a scheduling problem.

has both non-empty time-feasible and resource-feasible solution sets and their intersection is a non-empty set, then the problem has feasible solutions. Now we propose an approach using the following procedure steps for each individual capacity agent:

1. Acceptance and initial solution - Detect if the problem is temporally over-constrained, and if it isn't, establish an initial solution, and proceed in the next step. If it is, terminate the procedure by rejecting the problem, for having no feasible solution;
2. Re-schedule to find a time-feasible solution - If the established solution is time-feasible proceed in the next step. If it isn't, re-schedule to remove all temporal conflicts;

3. Re-schedule to find a feasible solution - For a resource-feasible solution terminate the procedure. In the other case, try to re-schedule to remove all capacity conflicts without creating temporal conflicts, resorting to cancellation, with task un-scheduling, as a last choice, if necessary.

As some approaches in the literature, this procedure starts by establishing one initial, possibly non-feasible, solution which is then repaired in order to remove conflicts (see, for instance, [Minton 1992]). Steps 1 and 2 of the procedure are oriented for a temporal scheduling perspective and concern only to a single scheduling problem of the agent. Step 3 is oriented for a resource scheduling perspective and can involve all scheduling problems of the agent at step 3, as all tasks of the agent compete for the same resource capacity.

resource-feasible. The re-scheduling activity referred can involve re-scheduling the requests and/or the task; as a last choice there can be request cancellation with task un-scheduling (a drastic last choice, but still an effective one).

4 STEP 1: SCHEDULING AN INITIAL SOLUTION

We show, through an example, how an agent can locally recognise a temporally over-constrained problem, and contribute to establish an initial solution in the case this is possible (step 1).

Suppose g_7 identifies a production agent having the task $o^{7}_{i,14}$ of network job in Figure 1. In Figure 3 a

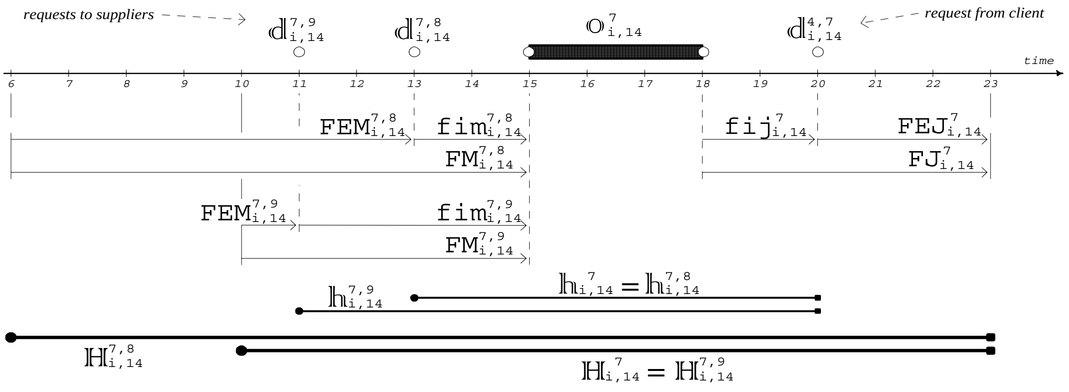

Figure 3- Temporal slacks for agent g_7 task $o^{7}_{i,14}$.

The problem rejection referred in step 1 is accomplished through request rejection and cancellation messages. In step 2, the initial solution, if not temporally-feasible, is repaired. The re-scheduling activity referred can involve re-scheduling the requests and/or the task. The re-scheduling requests are agreements on new client and/or suppliers due-dates for the existing product requests. These are accomplished through re-scheduling request, and corresponding acceptance and rejection types of messages, and must have high priority because scheduling for a time-feasible solution is at stake. This is possible, as the problem was previously recognised as not temporally over-constrained. In step 3, the idea is to repair the present solution by restricting the search for a better one to the set of time-feasible solutions, and search for a solution that is not only time-feasible, but also

possible situation for this task is represented on a timeline. As agent g_7 has two suppliers for the task, two requests to two suppliers are shown, $d^{7,8}_{i,14}$ and $d^{7,9}_{i,14}$, besides the request from the client, $d^{4,7}_{i,14}$. Temporal slacks and derived intervals (denoted by h and H symbols) are also shown in Figure 3. Symbols `fij`, `fim`, `FEJ`, `FEM`, `FJ` and `FM` denote, respectively, the *internal downstream slack*, *internal upstream slacks*, *external downstream slack*, *external upstream slacks*, *downstream slack* and *upstream slacks*. For each kind of upstream slacks there is one per each supplier. Slacks are represented by arrows.

By definition:

$$FJ^{7}_{i,14} = FEJ^{7}_{i,14} + fij^{7}_{i,14}$$

$$\mathtt{fij}^{7}_{i,14}=\mathtt{TIME}(\mathbb{d}^{4,7}_{i,14})-\mathtt{ENDTIME}(\mathbb{O}^{7}_{i,14})$$

$$\mathtt{FM}^{7,j}_{i,14}=\mathtt{FEM}^{7,j}_{i,14}+\mathtt{fim}^{7,j}_{i,14}$$

$$\mathtt{fim}^{7,j}_{i,14}=\mathtt{STARTIME}(\mathbb{O}^{7}_{i,14})-\mathtt{TIME}(\mathbb{d}^{7,j}_{i,14})$$

(the last two for $\mathtt{j=8,9}$)

Internal slacks are inserted locally, and by the initiative of the agent, when scheduling the task and making requests to suppliers; external slacks are imposed by the other agents of the network. We will assume that, in any case, *the agent will maintain positive internal slacks*. Each of the $\mathbb{h}$'s are intervals between one of the supplier due-dates and the client due-date (13 and 20, and 11 and 20, in Figure 3); the $\mathbb{H}$'s are intervals between one of the earliest start times and the latest finish time for the task (6 and 23, and 10 and 23, in Figure 3). The temporal end points of the most restrictive $\mathbb{H}$ interval are hard temporal constraints of the problem (10 and 23 in Figure 3), in particular for the task. Let us denote these by $\mathtt{RD}^{7}_{i,14}$ and $\mathtt{DD}^{7}_{i,14}$, for the upstream and downstream point, respectively.

It is easy to see that, in order for a solution to be time-feasible, the interval of the task must be contained in the most restricted $\mathbb{h}$ interval, and each $\mathbb{h}$ interval must be contained in the corresponding (same supplier) $\mathbb{H}$ interval (and this must also hold for every capacity agent involved in the network job). For this to hold, no temporal slack can be negative. Also, if the duration of the most restrictive $\mathbb{H}$ interval is less than the task duration, the problem is temporally over-constrained, and the agent can reject it.

We propose that product request messages from the client, additionally to product request information, *carry the value of the* FEJ *slack*; also, request acceptance messages from suppliers will *carry the value of the respective* FEM *slack*.[3] In our example, agent $\mathtt{g}_7$ would then calculate the $\mathtt{DD}^{7}_{i,14}$ and $\mathtt{RD}^{7}_{i,14}$ values by:

$$\mathtt{DD}^{7}_{i,14}=\mathtt{TIME}(\mathbb{d}^{4,7}_{i,14})+\mathtt{FEJ}^{7}_{i,14}$$ and

$$\mathtt{RD}^{7}_{i,14}=\underset{\mathtt{j=8,9}}{\mathtt{MAX}}\,(\mathtt{RD}^{7,j}_{i,14})$$

where:

$$\mathtt{RD}^{7,j}_{i,14}=\mathtt{TIME}(\mathbb{d}^{7,j}_{i,14})-\mathtt{FEM}^{7,j}_{i,14}\quad(\mathtt{j=8,9})$$

When the agent receives request $\mathbb{d}^{4,7}_{i,14}$ from the client, it will, guaranteeing non-negative fij and fim values, schedule the task and make requests $\mathbb{d}^{7,8}_{i,14}$ and $\mathbb{d}^{7,9}_{i,14}$ to suppliers, passing them also the (supplier FEJ) value:

$$\mathtt{FJ}^{7}_{i,14}+\mathtt{fim}^{7,j}_{i,14}\qquad(\mathtt{j=8,9})$$

When the agent receives all the request acceptances from the suppliers, verifies first if the problem is temporally over-constrained, by testing if:

$$\mathtt{DD}^{7}_{i,14}-\mathtt{RD}^{7}_{i,14} < \mathtt{DURATION}(\mathbb{O}^{7}_{i,14})$$

If this is true the problem must be rejected. Otherwise, the agent will send the acceptance message to the client, passing it also the (client FEM) value:

$$\mathtt{FM}^{7}_{i,14}+\mathtt{fij}^{7}_{i,14}$$

where:

$$\mathtt{FM}^{7}_{i,14}=\mathtt{STARTIME}(\mathbb{O}^{7}_{i,14})-\mathtt{RD}^{7}_{i,14}.$$

If the step 1 concludes with a non temporally over-constrained problem, agent $\mathtt{g}_7$ will internally keep the tuple:

$$< \mathtt{DD}^{7}_{i,14},\ \left\{\mathtt{RD}^{7,8}_{i,14},\mathtt{RD}^{7,9}_{i,14}\right\},\ \mathbb{d}^{4,7}_{i,14},\ \left\{\mathbb{d}^{7,8}_{i,14},\mathbb{d}^{7,9}_{i,14}\right\},\ \mathbb{O}^{7}_{i,14} >$$

as its own local perspective of the network scheduling problem and its initial solution.

5 STEP 2: RE-SCHEDULING FOR A TIME-FEASIBLE SOLUTION

We now show how, starting from an initial solution with temporal conflicts, can an agent locally contribute to obtain a time-feasible solution (step 2), following the same example.

Clearly, for the local situation represented Figure 3, all slacks are positive: all slack arrows point to the future side of the time line. So, the solution is seen as

[3] For a retail agent, the FEJ value can be pre-specified (*e.g.*, 0). For a raw-material agent the FEM value will be the difference between the due-date of the agent global request to the outside and the global RD date.

temporally-feasible by agent g_7, and g_7 will do nothing, unless it receives any re-scheduling request, which it could accept provided non-negative internal slacks can be maintained (this must also always hold in the following).

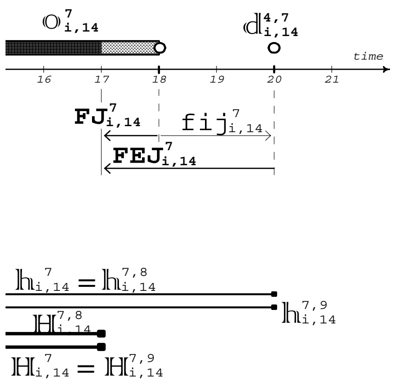
Figure 4- Re-scheduling case 1.

We now pose four kinds of different scenarios were, at least agent g_7 must take the initiative of some re-scheduling actions. These are described by the re-scheduling cases 1, 2, 3 and 4, in Figure 4, Figure 5, Figure 6 and Figure 7, respectively. Cases 1 and 2 *must be tested for first*.

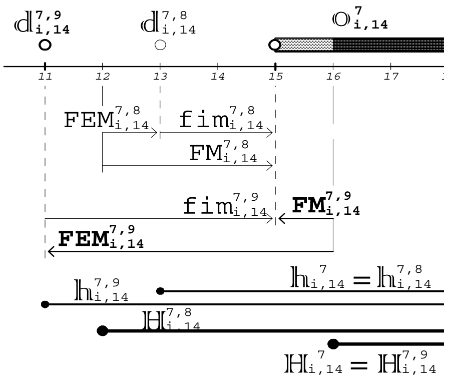
Figure 5- Re-scheduling case 2.

Case 1 occurs if $\mathtt{FJ}_{i,14}^{7} < 0$ *and* $\mathtt{FEJ}_{i,14}^{7} < 0$ (the task and the client request violate the hard temporal constraint downstream, 17 in Figure 4). The detection of case 1 must be followed by the appropriate task re-scheduling and client request re-scheduling to earlier times. The re-scheduling of some requests to suppliers can then be necessary to maintain non-negative internal (fim) slacks.

Case 2 occurs if $\mathtt{FM}_{i,14}^{7,j} < 0$ *and* $\mathtt{FEM}_{i,14}^{7,j} < 0$, for some $j=8,9$ (the task and some requests to suppliers violate hard temporal constraints upstream, 16 in Figure 5). The detection of case 2 must be followed by the appropriate task re-scheduling and the re-scheduling of the offending requests to suppliers to later times. The re-scheduling of the client request can then be necessary to maintain a non-negative internal (fij) slack.

After handling cases 1 and 2, cases 3 and 4 are handled.

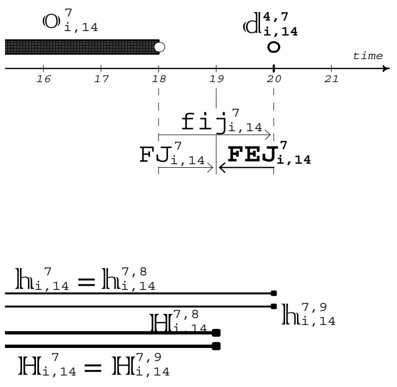
Figure 6- Re-scheduling case 3.

Case 3 occurs if $\mathtt{FJ}_{i,14}^{7} < 0$. (the client request violates the hard temporal constraint downstream, 19 in Figure 6). The detection of case 3 must be followed by the appropriate client request re-scheduling to an earlier time.

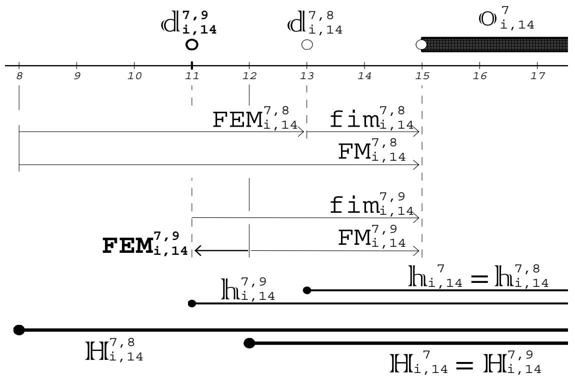
Figure 7- Re-scheduling case 4.

Case 4 occurs if $\mathtt{FEM}_{i,14}^{7,j} < 0$, for some $j=8,9$ (some requests to suppliers violate hard temporal constraints upstream, 12 in Figure 7). The detection of case 4 must be followed by the appropriate re-scheduling of the offending requests to suppliers later times.

6 FUTURE WORK AND CONCLUSION

We described an approach for multi-agent co-operative scheduling based on a three step procedure for individual agents. Step 1 allows agents to detect locally if the problem is temporally over-constrained and, in the case it isn't, schedule an initial, possibly non time-feasible solution. By locally exchanging specific temporal slack values, agents are able to locally perceive the hard global temporal constraints of a problem, and rule out non time-feasible solutions in the subsequent steps. Each of the slack values exchanged in step 1 corresponds, for a particular agent, to a sum of slacks, downstream and upstream the agent network, and so, they cannot be considered private information of any agent in particular. In step 2, if necessary, agents repair the initial solution to obtain a time-feasible one.

The procedure is very general with respect to its step 3. This step can be refined to accommodate additional improved co-ordination mechanisms for implementing certain search strategies, based on capacity/resource constrainedness (*e.g.*, see [Sycara 1991] or [Sadeh 1994]), leading the agents on a fast convergence to specific feasible solutions. For instance, feasible solutions satisfying some scheduling preferences or optimising some criteria, either from an individual agent perspective, or from a global one, or both. This is a subject of our future work.

7 REFERENCES

Arnold 1997 Arnold, Jörg, et al, 1997, *Production Planning and Control within Supply Chains*, ESPRIT project 20544, X-CITTIC.

Blazewicz 1994 Blazewicz, J.;Ecker, K.H.;Schmidt, G.;Weglarz, J., 1994, *Scheduling in Computer and Manufacturing Systems*, Springer Verlag, 1994

Camarinha-Matos 1999 Camarinha-Matos, Luís M.; Afsarmanesh, Hamideh, 1999, *The Virtual Enterprise Concept*, in Infrastructures for Virtual Enterprises, Camarinha-Matos, L.M. and Afsarmanesh, H. (eds.), Kluwer Academic Publishers, Dordrecht, The Netherlands, 1999, 3-14.

Fox 1993 Fox, Mark S., et al, 1993, *The Integrated Supply Chain Management System*, Department of Industrial Engineering, University of Toronto, Canada.

Garey 1979 Garey, M.R.; Johnson, D.S., 1979, *Computers and Intractability: A Guide to the Theory of NP-Completeness*, W.H. Freeman and Co., New York.

Graves 1993 Graves, S.C.; Kan, A.H.G. Rinnooy; Zipkin, P.H., (Eds.), *Logistics of Production and Inventory*, Handbooks in Operations Research and Management Science, Volume 4, North-Holland, Amsterdam, 1993.

ICMAS 1996 ICMAS 1996, 1996, *Proceedings of the Second International Conference on Multi-Agent Systems (Contents)*, ICMAS-96, AAAI Press, Menlo Park, California, 1996.

Kjenstad 1998 Kjenstad, Dag, 1998, *Coordinated Supply Chain Scheduling*, PhD. Thesis, Norwegian University of Science and Technology, Trondheim, Norway.

Minton 1992 Minton, Steven, et al, 1992, *Minimizing Conflicts: a Heuristic Repair Method for Constraint Satisfaction and Scheduling Problems*, Artificial Intelligence 58, 1992, 161-205.

O'Hare 1996 O'Hare, G.M.P.; Jennings, N.R., 1996, *Foundations of Distributed Artificial Intelligence*, John Wiley & Sons, Inc., 1996, New York, USA.

O'Neill 1996 O'Neill, H.; Sackett, P., 1996, *The Extended Enterprise Reference Framework*, Balanced Automation Systems II, Camarinha-Matos, L.M. and Afsarmanesh, H. (Eds.), 1996, Chapman & Hall, London, UK, pp 401-412

Rabelo 1998 Rabelo, Ricardo J.; Camarinha-Matos, Luis M.; Afsarmanesh, Hamideh, 1998, *Multiagent Perspectives to Agile Scheduling*, Basys'98 International Conference on Balanced Automation Systems, Prague, Czech Republic, 26-28/08/98.

Reis 1998 Reis, J.; Mamede, N.; O'Neill, H., 1998, *Ontologia para um Modelo de Planeamento e Controlo na Empresa Estendida*, Proceedings of the IBERAMIA'98, Lisbon, Portugal, October 5-9, 1998, Helder Coelho (Ed.), Edições Colibri, Lisbon, Portugal, pp. 43-54 (in portugese).

Reis 1999a Reis, J.; Mamede, N.; O'Neill, H., 1999, *Agent Communication for Scheduling in the Extended Enterprise*, Proceedings of the IFIP TC5 WG5.3/PRODNET Conference on Infrastructures for Virtual Enterprises, Porto, Portugal, October 27-28, 1999, Camarinha-Matos, L.M., Afsarmanesh H. (eds.), Kluwer Academic Publishers, Dordrecht, The Netherlands, 1999, pp 353-364.

Reis 1999b Reis, J.; Mamede, N., 1999, *What's in a Node, Nodes and Agents in Logistic Networks*, Proceedings of the ICEIS'99, 1st International Conference on Enterprise Information Systems, Setúbal, Portugal, Mars 27-30, 1999, Filipe, J. and Cordeiro, J. (Eds.), pp. 285-291.

Reis 1999c Reis, J.; Mamede, N., 1999, *Task Modeling in a Multi-Agent Logistic Domain*, Proceedings of the ICEIS'99, 1st International Conference on Enterprise Information Systems, Setúbal, Portugal, Mars 27-30, 1999, Filipe, J. and Cordeiro, J. (Eds.), pp. 769.

Reis 2000 Reis, J.; Mamede, N., 2000, *An Agent Architecture for Multi Agent Dynamic Scheduling*, Proceedings of the ICEIS'2000, 2st International Conference on Enterprise Information Systems, Stafford, UK, 4-7 July, 2000, Sharp, B., Cordeiro, J. and Filipe, J. (Eds.), pp. 203-208.

Sadeh 1994 Sadeh, Norman, 1994, *Micro-Oportunistic Scheduling: The Micro-Boss Factory Scheduler*, Intelligent Scheduling, Morgan Kaufman, 1994, Chapter 4.

Sycara 1991 Sycara, Katia P.;Roth, Steven F.;Sadeh, Norman;Fox, Mark S., 1991, *Resource Allocation in Distributed Factory Scheduling*, IEEE Expert, February, 1991, 29-40.

Zweben 1994 Zweben, Monte; Fox, Mark S., 1994, *Intelligent Scheduling*, Morgan Kaufmann Publishers, Inc., San Francisco, California, 1994.