Scheduling, Re-Scheduling and Communication in the Multi-Agent Extended Enterprise Environment

Joaquim Reis[1], Nuno Mamede[2]

[1] ISCTE, Dept. Ciências e Tecnologias de Informação, Avenida das Forças Armadas, 1649-026 Lisboa, Portugal
Joaquim.Reis@iscte.pt
[2] IST, Dept. de Engenharia Informática, Avenida Rovisco Pais, 1049-001 Lisboa, Portugal
Nuno.Mamede@acm.org

Abstract. In this article we describe a multi-agent dynamic scheduling environment where autonomous agents represent enterprises and manage the capacity of individual macro-resources in a production-distribution context. The agents are linked by client-supplier relationships and inter-agent communication must take place. The model of the environment, the appropriate agent interaction protocol and a cooperative scheduling approach, emphasizing a temporal scheduling perspective of scheduling problems, are described. The scheduling approach is based on a coordination mechanism supported by the interchange of certain temporal information among pairs of client-supplier agents involved. This information allows the agents to locally perceive hard global temporal constraints and recognize non over-constrained problems and, in this case, rule out non temporally-feasible solutions and establish an initial solution. The same kind of information is then used to guide re-scheduling to repair the initial solution and converge to a final one.

Keywords. Scheduling, Multi-Agent Systems, Supply-Chain Management.

1 Introduction

Scheduling is the allocation of resources over time to perform a collection of tasks, subject to temporal and capacity constraints. In classical/Operations Research (OR) scheduling approaches, a centralized perspective is assumed: all problem data is known by a central entity and scheduling decisions are taken by the same entity, based on a well defined criteria. Sometimes, in more modern Artificial Intelligence (AI), or mixed OR/AI, based approaches, the same kind of centralized perspective is assumed too. For a general introduction to OR approaches to scheduling problems see [9] or [2]; AI based approaches can be found in [5], [25] or [10], for instance. Planning and coordination of logistics activities has been, in the areas of OR/Management Science, the subject of investigation since the sixties [8]. The problem of scheduling in this kind of environments has had, recently, a more dedicated attention; see [6], [1], [11] or [15], for instance. In this article, the specific

logistics context of the supply-chain/Extended Enterprise (EE) [14] is considered, for the short-term activity of scheduling of production-distribution tasks. The EE is usually assumed to be a kind of cooperative Virtual Organization, or Virtual Enterprise, where the set of inter-dependent participant enterprises is relatively stable; for concepts, terminology and classification see [3]; in [4] other approaches to scheduling in this kind of context can be found. A distributed approach is more natural in this case, because scheduling data and decisions are inherently distributed, as resources are managed by individual, geographically decentralized and autonomous entities (enterprises, organizations). So, in our approach, we adopted the AI Multi-Agent Systems paradigm (see [13] or [24]). In the following, we describe ongoing investigation developing from work published on the subject of multi-agent scheduling in production-distribution environments (see [16], [17], [18], [19], [20] and [21]).

The scheduling problems we consider have the following features:

a) Communication is involved - Agents must communicate (at least to exchange product orders with clients/suppliers);
b) Cooperation is involved - Each agent must cooperate so that it won't invalidate feasible scheduling solutions;
c) Scheduling activity is highly dynamic - Problem solution development is an on-going process during which unforeseen events must always be accommodated, and re-scheduling and giving up a scheduling problem are options to be considered.

The following sections present: a brief description of the model of the multi-agent scheduling environment (sec.2), the agent interaction protocol (sec.3), the cooperative approach proposed for multi-agent scheduling problems (sec.4), the initial scheduling step (sec.5) and the re-scheduling (sec.6), both from an individual agent perspective, and finally, future work and conclusions (sec.7). Secs. 5 and 6 are presented with examples based on simulations.

2 The Scheduling Environment Model

In past work (referred above) we have proposed a model of the EE scheduling environment based on an *agent network*, with each agent managing an aggregate scheduling resource, representing a *production*, a *transportation*, or a *store* resource, and linked through client-supplier relationships. A scheduling resource is just an individual node of a *physical network* of resources, and accommodates the representation of the agent tasks scheduled and the available capacity along a certain scheduling temporal horizon. Ordinary network agents are named *capacity*, or *manager*, *agents*, because they are responsible for managing the capacity of a resource, and they can be *production*, *transportation* or *store* class agents; production and transportation class agents are grouped under the *processor* agent class, because the capacity they manage is based on a rate. A special *supervision agent*, with no resource, plays the role of an interface with the outside, and can introduce new scheduling problems into the agent network.

Pairs of client-supplier capacity agents can communicate, basically through the exchange of product request messages (see next section), which contain product, quantity of product and proposed due-date information. The supervision agent communicates with special agents playing the roles of *retail* and *raw-material* agents, located at the downstream and the upstream end of the agent network (which are pure clients and pure suppliers for the network), respectively.

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 DD, and the global release date RD. 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 (networks can be multi-product so, the set of products deliverable can depend on the retail agent); 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 canceled (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.[1] The task consumes a non-changing amount of resource capacity during its temporal interval. The duration of the task depends on the product, the quantity of product and additional parameters related to the characteristics of the resource and, in the case of processor agents, to the amount of capacity dedicated to the task.[2] 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, except for store tasks (with flexible duration, and minimum of 1 time unit).

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 belonging to a *network job*, see example in Fig. 1. This is a just an analogue of the concept of job used in classical production scheduling problems.

---

[1] We assume that there is, for each capacity agent, a unique supplier for each supply product. As a result of this simplifying assumption, the lack of a product supply has, as a final result, the network being unable to satisfy a global product request from the outside. Allowing multiple suppliers for the same supply product opens the door to another issues (like choosing the preferred supplier, possibly with negotiation based on prices, or due-dates), in which we are not interested, for now.

[2] Basically, more capacity invested gives a shorter task duration.

A *solution* for a scheduling problem is a set of product requests agreed by pairs of client-supplier agents and the set of agent tasks, necessary to satisfy the global product request given by the problem, forming the corresponding network job. 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

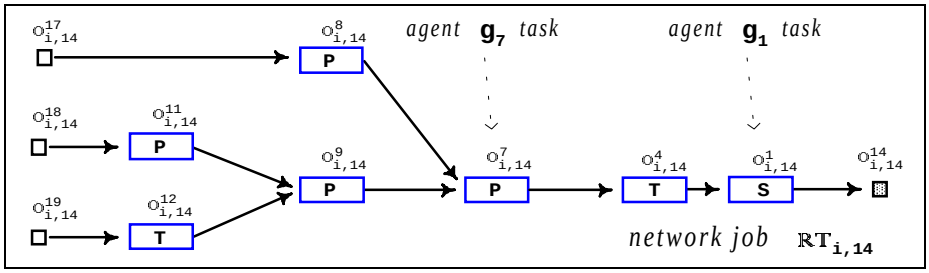

Fig. 1. A network job (job $\mathbf{RT_{i,14}}$). The task of an agent $\mathbf{g_k}$ for the $\mathbf{i^{th}}$ global request to retail agent $\mathbf{g_r}$ is denoted by $\mathbf{O}^{k}_{i,r}$ (and this task belongs to the network job denoted by $\mathbf{RT}_{i,r}$); **P**, **T** and **S** denote production, transportation and store tasks, respectively.

capacity over-allocation must occur with any task of the solution, for 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-date(s) agreed for the request(s) made to the supplier(s), and the latter due-date(s) must precede the former.

More details on the resources and the physical network are given in [16] and [18]; about the agent network and agent architecture see [17], [18], and [19].

3 The Agent Interaction Protocol

In this section we expose the high level inter-agent protocol used for scheduling in the EE network.

The agent interaction activity for scheduling occurs through the interchange of messages, of predetermined types, between pairs of client-supplier agents. The exchange of a message always occurs in the context of a *conversation* between the sender and the receiver agents. A conversation has a *conversation model*, which contains information about the predetermined sequences of the types of messages exchangeable and is defined through a finite state machine. An *interaction protocol* is defined as a set of conversation models. For the interaction of capacity agents we defined the *Manager-Manager* interaction protocol, represented in Fig. 2.

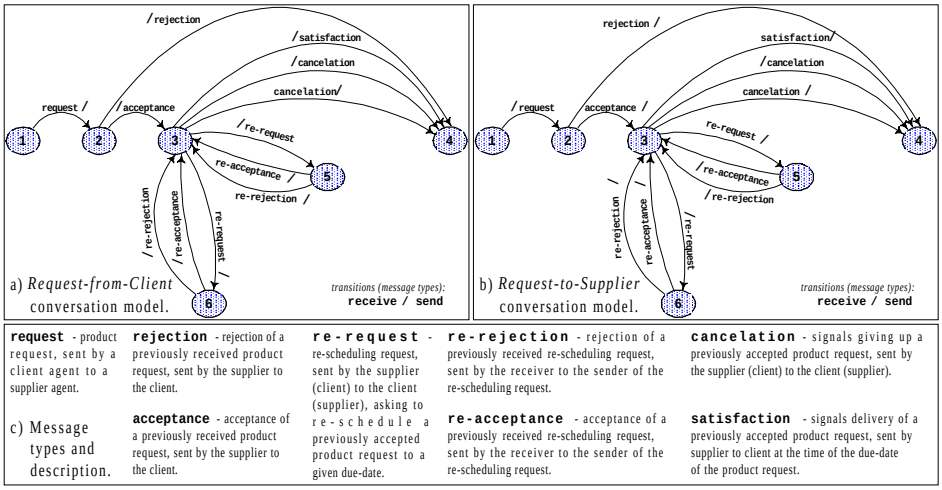

Fig. 2. Conversation models and messages for the *Manager-Manager* interaction protocol.

The protocol has the associated conversation models *Request-from-Client* and *Request-to-Supplier*, described as finite state machine diagrams in Fig. 2-a and Fig. 2-b, and to be used by an agent when playing the roles of a supplier and a client agent, respectively. Fig. 2-c describes the types of messages exchangeable.

4 A Cooperative Multi-Agent Scheduling Approach

Classically, scheduling is considered a difficult problem [7]. In general, 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 effort is invested in coordination 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 coordination efforts are considered satisfactory.

The approach we propose in this article, for the cooperative individual (agent) scheduling behavior, is a *minimal approach*, that is, an agent viewing a scheduling problem solution as feasible won't do anything respecting to the scheduling problem. A first version of this approach appeared in [20]; in [21] we presented a refined approach only for processor, *i.e.*, production or transportation, agents; in the present article we cover also store agents and, additionally, include the respective *minimal* re-scheduling actions for processor and store agent cases.

Consider two sets of solutions for a scheduling problem: the set of *time-feasible solutions*, which respect all temporal constraints, and the set of *resource-feasible solutions*, which respect all capacity constraints. A feasible solution is one that is

both time-feasible and resource-feasible so, the set of feasible solutions is the intersection of those two sets. A problem is *temporally over-constrained* if the set of time-feasible solutions is empty, and is *resource over-constrained* if the set of resource-feasible solutions is empty. If a problem has both non-empty time-feasible and resource-feasible solution sets, and their intersection is non-empty, then the problem has feasible solutions. We propose an approach using the following three step procedure, for each individual capacity agent:

Step 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, because it has no feasible solution;

Step 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;

Step 3. Re-schedule to find a feasible solution - For a resource-feasible solution, terminate the procedure; otherwise, 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, [12]. 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 agent resource capacity.

5 Step 1: Scheduling an Initial Solution

We now show, through examples for processor and store class agents, how an agent can locally recognize a non temporally over-constrained problem and, in that case, contribute to establish an initial solution (step 1).

For a processor agent, suppose an agent g_7 has a scheduling problem with the processor task $O^{7}_{i,14}$ of network job in Fig. 1. In Fig. 3-a, a possible situation for this task (which also represents a feasible solution for the problem) is represented on a timeline. As 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, $d^{4,7}_{i,14}$. Intervals (denoted by h and H symbols) and temporal slacks are shown in Fig. 3-a. 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:

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

$$FM^{7,j}_{i,14}=FEM^{7,j}_{i,14}+fim^{7,j}_{i,14},\quad fim^{7,j}_{i,14}=STARTTIME(\mathbb{O}^{7}_{i,14})-TIME(\mathbb{d}^{7,j}_{i,14})\quad (j=8,9)$$

Internal slacks are inserted locally, 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. It is assumed that, in any case, *the agent will maintain non negative internal slacks*. Each of the $\mathbb{h}$'s is an interval between one of the supplier due-dates and the client due-date (13 and 19, and 12 and 19, in Fig. 3-a); each of the $\mathbb{H}$'s is an interval between one of the earliest start times and the latest finish time for the task (10 and 21, and 11 and 21, in Fig. 3-a). Each of the latter pairs of temporal

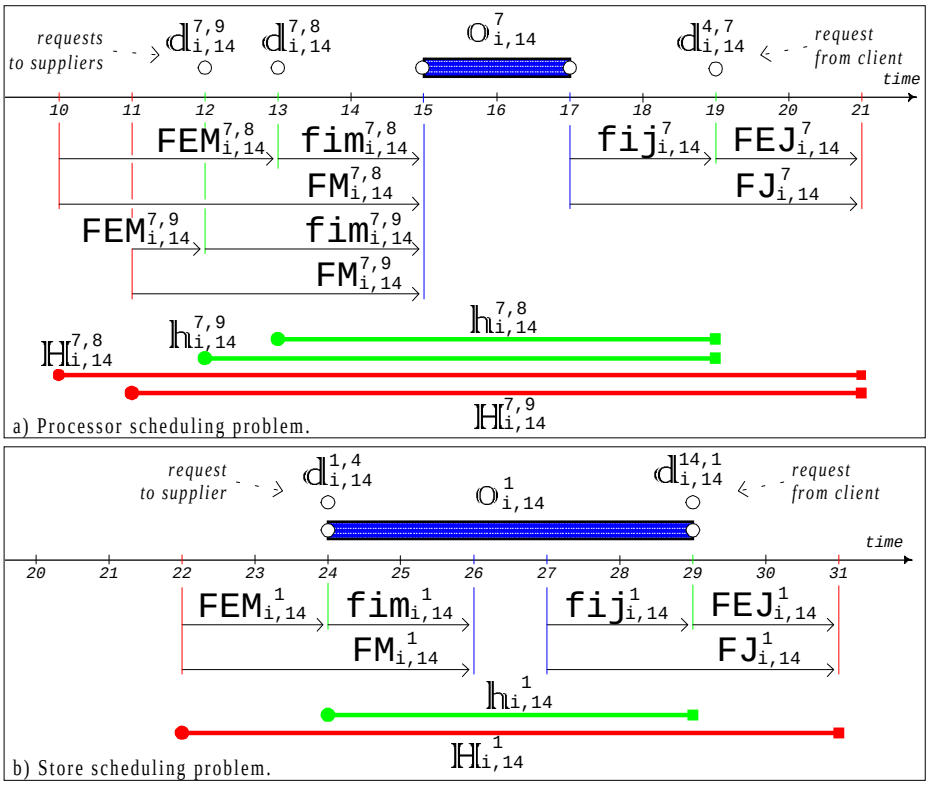

Fig. 3. Scheduling problem parameters: a) for processor agent $\mathbf{g_7}$, and b) for store agent $\mathbf{g_1}$ (for the values in the timelines of these two cases no relationship is intended).

points are hard temporal constraints for one of the former pairs of due-dates. Also, the temporal end points of the most restrictive $\mathbb{H}$ interval (11 and 21 in Fig. 3-a) are hard temporal constraints for the task; let us denote these by $RD^{7}_{i,14}$ and $DD^{7}_{i,14}$, 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. 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*. In our example, agent g_7 would then calculate the $DD^{7}_{i,14}$ and $RD^{7}_{i,14}$ values by:

$$DD^{7}_{i,14}=TIME(\mathbb{d}^{4,7}_{i,14})+FEJ^{7}_{i,14} \quad \text{and} \quad RD^{7}_{i,14}=\underset{j=8,9}{MAX}(RD^{7,j}_{i,14})$$

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

When the agent receives request $\mathbb{d}^{4,7}_{i,14}$ from the client, it will, guaranteeing non-negative values of fij and fim for the task to be scheduled, make requests $\mathbb{d}^{7,8}_{i,14}$ and $\mathbb{d}^{7,9}_{i,14}$ to suppliers, passing them also the (supplier FEJ) value $FJ^{7}_{i,14}+fim^{7,j}_{i,14}$ (for 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 $DD^{7}_{i,14}-RD^{7}_{i,14} < DURATION(\mathbb{O}^{7}_{i,14})$. 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 $FM^{7}_{i,14}+fij^{7}_{i,14}$, where $FM^{7}_{i,14}=STARTIME(\mathbb{O}^{7}_{i,14})-RD^{7}_{i,14}$. If step 1 concludes with a non temporally over-constrained problem, agent g_7 will internally keep the tuple $<DD^{7}_{i,14},\{RD^{7,8}_{i,14},RD^{7,9}_{i,14}\},\mathbb{d}^{4,7}_{i,14},\{\mathbb{d}^{7,8}_{i,14},\mathbb{d}^{7,9}_{i,14}\},\mathbb{O}^{7}_{i,14}>$, which represents the agent local perspective of the scheduling problem, and also includes (a part of) the initial solution.

For a store agent, suppose an agent g_1 has a scheduling problem with the store task $\mathbb{O}^{1}_{i,14}$ of network job in Fig. 1. In Fig. 3-b, a possible situation for this task (which also represents a feasible solution for the problem) is represented on a timeline. The case is similar to the one for agent g_7, with the exceptions described in the following. There is only one request to a supplier, as g_1 is a store agent. The internal slacks are defined differently: the task interval is equal to the (unique) $\mathbb{h}$ interval, and part of the task duration is considered as internal slack (if its duration is greater than 1, which is the minimum assuming the task must exist), with the relationship $fij^{1}_{i,14}+fim^{1}_{i,14}=DURATION(\mathbb{O}^{1}_{i,14})-1$ always holding. Fig. 3-b suggests *symmetrical* definitions for fij and fim slacks, with the minimum duration interval "centered" in the $\mathbb{h}$ interval but, for purposes of temporal constraint

violation identification (see next section), the following definitions must be used. For the downstream side violations (cases 1 and 3, in the next section), fij and fim are defined by (the minimum duration interval is shifted to the left):

$fij^{1}_{i,14}$=DURATION($\mathbb{O}^{1}_{i,14}$)-1, and $fim^{1}_{i,14}$=0

For the upstream side violations (cases 2 and 4, in the next section), fij and fim are defined by (the minimum duration interval is shifted to the right):

$fij^{1}_{i,14}$=0, and $fim^{1}_{i,14}$=DURATION($\mathbb{O}^{1}_{i,14}$)-1

The problem is temporally over-constrained if $DD^{7}_{i,14}$-$RD^{7}_{i,14}$ < 1. The values of $FEJ^{1}_{i,14}$+DURATION($\mathbb{O}^{1}_{i,14}$)-1 and $FEM^{1}_{i,14}$+DURATION($\mathbb{O}^{1}_{i,14}$)-1 must be passed to the supplier and to the client (as the supplier FEJ value, and the client FEM value), respectively. Finally, if step 1 concludes with a non temporally over-constrained problem, agent g_1 will keep the tuple <$DD^{1}_{i,14}$,{$RD^{1}_{i,14}$},$\mathbb{d}^{14,1}_{i,14}$, {$\mathbb{d}^{1,4}_{i,14}$},$\mathbb{O}^{1}_{i,14}$>.

6 Step 2: Re-Scheduling for a Time-Feasible Solution

In this section we show how, starting from an initial solution with temporal conflicts, agents g_7 and g_1 can locally contribute to obtain a time-feasible solution (step 2).

For the local situations represented in Fig. 3-a and Fig. 3-b, all slacks are positive so, the solution is seen as temporally-feasible (by agent g_7, and agent g_1, respectively). In these cases, an agent will do nothing, unless it receives any re-scheduling request, which it could accept provided non-negative internal slacks can be maintained, for a processor agent, or a task with duration greater than 0 is possible, in the case of a store agent. Otherwise, four kinds of possible local situations can occur where the agent itself must take the initiative of some re-scheduling actions.

The situations referred are described by the re-scheduling cases 1, 2, 3 and 4, for which we show examples in Fig. 4 for processor agents (for agent g_7), and in Fig. 5 for store agents (for agent g_1). Each figure represents, for each case: a) the situation detected, and b) the situation after a minimal re-scheduling action. No relationship is intended among timeline values of the processor and the store agent cases. In the text following, upper indexes are omitted in slack symbols in order to cover both, processor and store, agent cases. Cases 1 and 2 *must be considered first*, by the agent.

Case 1 occurs if $FJ_{i,14}$<0 *and* $FEJ_{i,14}$<0 (the task and the client request violate the hard temporal constraint downstream, 17 in Fig. 4-1-a, and 20 in Fig. 5-1-a). The detection of case 1 must be followed by the appropriate task re-scheduling and client

request re-scheduling to earlier times (resulting in the situation shown in Fig. 4-1-b, and Fig. 5-1-b). Re-scheduling of some requests to suppliers can (or cannot) then be necessary to maintain non-negative `fim` slacks, at the upstream side.

Case 2 occurs if, for some supplier, $\mathtt{FM}_{i,14}<0$ *and* $\mathtt{FEM}_{i,14}<0$, (the task and

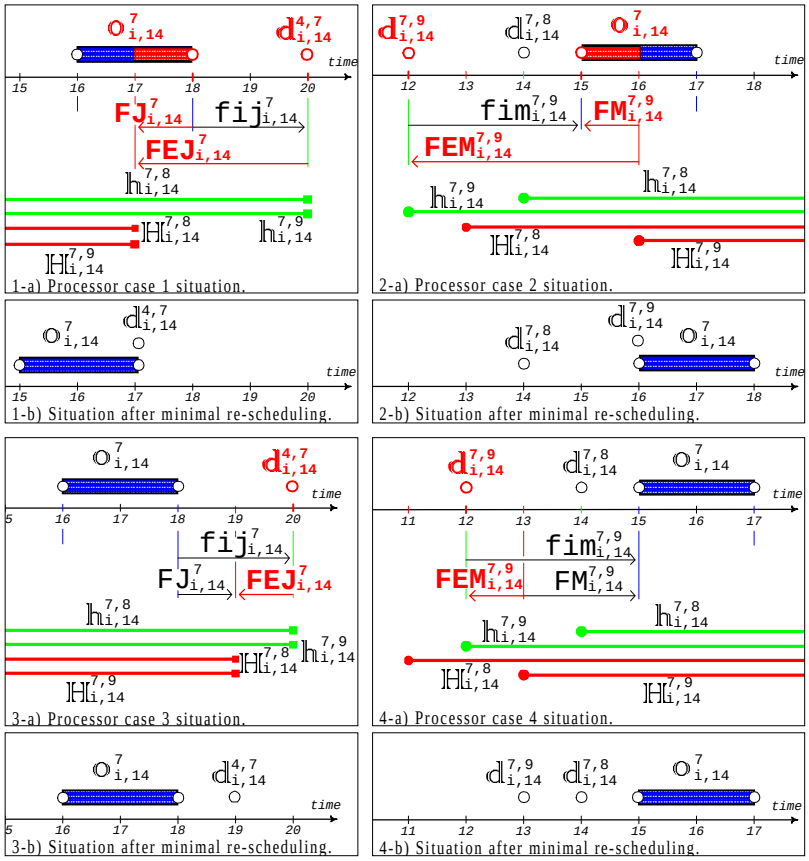

Fig. 4. Examples of re-scheduling cases 1, 2, 3 and 4, for a processor agent, with situations before, and after, minimal re-scheduling.

some requests to suppliers violate hard temporal constraints upstream, 16 in Fig. 4-2-a, and 29 in Fig. 5-2-a). 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 (resulting in the situation shown in Fig. 4-2-b, and Fig. 5-2-b). Re-scheduling of the client request can (or cannot) then be necessary to maintain a non-negative `fij` slack, at the downstream side.

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

Case 3 occurs if $\mathtt{FJ}_{i,14}<0$ (the client request violates the hard temporal constraint downstream, 19 in Fig. 4-3-a, and 26 in Fig. 5-3-a). The detection of case

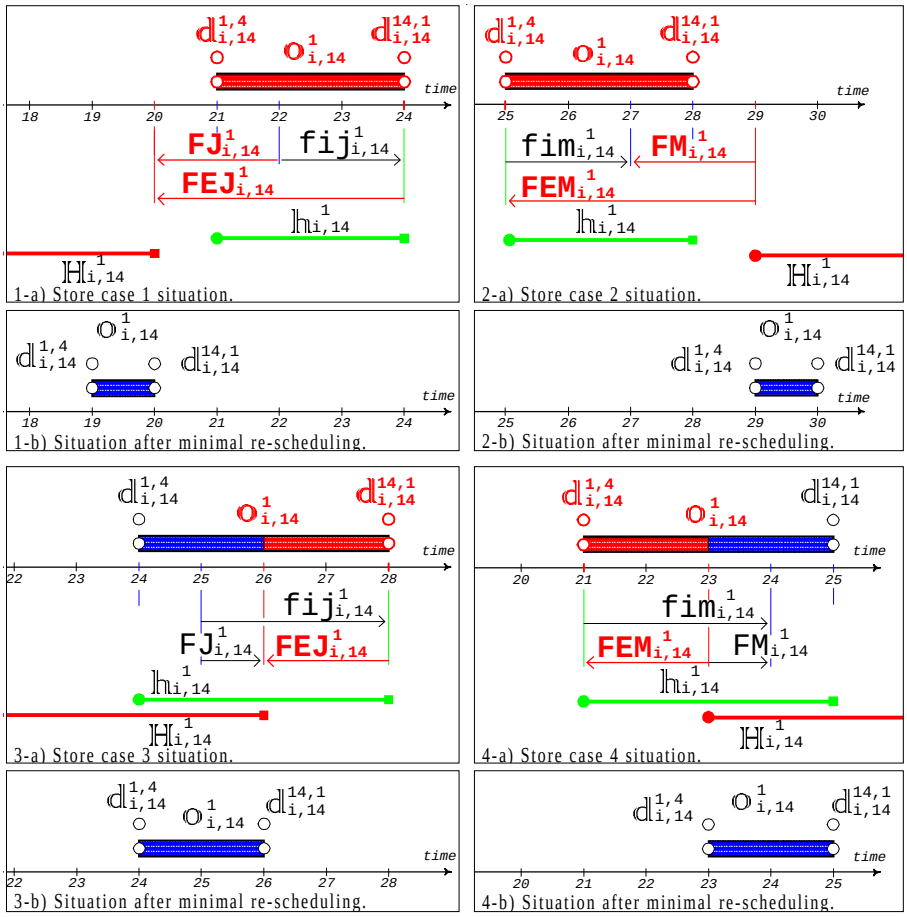

Fig. 5. Examples of re-scheduling cases 1, 2, 3 and 4, for a store agent, with situations before, and after, minimal re-scheduling.

3 must be followed by the appropriate client request re-scheduling to an earlier time (resulting in the situation shown in Fig. 4-3-b, and Fig. 5-3-b).

Case 4 occurs if, for some supplier, $FEM_{i,14} < 0$ (some requests to suppliers violate hard temporal constraints upstream, 13 in Fig. 4-4-a, and 23 in Fig. 5-4-a). The detection of case 4 must be followed by the appropriate re-scheduling of the offending requests to suppliers to later times (resulting in the situation shown in Fig. 4-4-b, and Fig. 5-4-b).

7 Conclusion and Future Work

We described a multi-agent dynamic scheduling environment involving communication and cooperation, and an approach for multi-agent cooperative 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 coordination mechanisms for implementing certain search strategies, based on capacity/resource constrainedness (*e.g.*, see [23] or [22]), leading the agents on a fast convergence to specific feasible solutions. For instance, feasible solutions satisfying some scheduling preferences or optimizing some criteria, either from an individual agent perspective, or from a global one, or both. This is a subject for our future work.

References

1. Arnold, Jörg, et al, *Production Planning and Control within Supply Chains*, ESPRIT project 20544, X-CITTIC, 1997.
2. Blazewicz, J.; Ecker, K.H.; Schmidt, G.; Weglarz, J., *Scheduling in Computer and Manufacturing Systems*, Springer Verlag, 1994
3. Camarinha-Matos, Luís M.; Afsarmanesh, Hamideh, *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.
4. Camarinha-Matos, Luís M.; Afsarmanesh, Hamideh, *Infrastructures for Virtual Enterprises, Networking Industrial Enterprises*, Luís M. Camarinha-Matos, Hamideh Afsarmanesh (eds.), Kluwer Academic Publishers, Dordrecht, The Netherlands, 1999.
5. Dorn, J.; Froeschel, K., *Scheduling of Production Processes*, Dorn, J.; Froeschel, K. (eds.), Elis Horwood, Ld., 1993.
6. Fox, Mark S., et al, *The Integrated Supply Chain Management System*, Department of Industrial Engineering, University of Toronto, Canada, 1993.
7. Garey, M.R.; Johnson, D.S., *Computers and Intractability: A Guide to the Theory of NP-Completeness*, W.H. Freeman and Co., New York, 1979.
8. 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.
9. Kan, A.H.G. Rinnooy, *Machine Scheduling Problems*, Martinus Nijhoff, The Hague, 1976.

10. Kerr, Roger; Szelke, Elizabeth (eds.), *Artificial Intelligence in Reactive Scheduling*, Chapman & Hall, 1995.
11. Kjenstad, Dag, *Coordinated Supply Chain Scheduling*, PhD. Thesis, Norwegian University of Science and Technology, Trondheim, Norway, 1998.
12. Minton, Steven, et al, *Minimizing Conflicts: a Heuristic Repair Method for Constraint Satisfaction and Scheduling Problems*, Artificial Intelligence 58, 1992, 161-205.
13. O'Hare, G.M.P.; Jennings, N.R., *Foundations of Distributed Artificial Intelligence*, John Wiley & Sons, Inc., 1996, New York, USA.
14. O'Neill, H.; Sackett, P., *The Extended Enterprise Reference Framework*, Balanced Automation Systems II, Camarinha-Matos, L.M. and Afsarmanesh, H. (eds.), 1996, Chapman & Hall, London, UK, 401-412
15. Rabelo, Ricardo J.; Camarinha-Matos, Luis M.; Afsarmanesh, Hamideh, *Multiagent Perspectives to Agile Scheduling*, Basys'98 International Conference on Balanced Automation Systems, Prague, Czech Republic, 1998.
16. Reis, J.; Mamede, N.; O'Neill, H., *Ontologia para um Modelo de Planeamento e Controlo na Empresa Estendida*, Proceedings of the IBERAMIA'98, Lisbon, Portugal, 1998, Helder Coelho (ed.), Edições Colibri, Lisbon, Portugal, 43-54 (in portuguese).
17. Reis, J.; Mamede, N.; O'Neill, H., *Agent Communication for Scheduling in the Extended Enterprise*, Proceedings of the IFIP TC5 WG5.3/PRODNET Conference on Infrastructures for Virtual Enterprises, Porto, Portugal, 1999, Camarinha-Matos, L.M., Afsarmanesh H. (eds.), Kluwer Academic Publishers, Dordrecht, The Netherlands, 1999, 353-364.
18. Reis, J.; Mamede, N., *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, 1999, Filipe, J. and Cordeiro, J. (eds.), 285-291.
19. Reis, J.; Mamede, N., *An Agent Architecture for Multi Agent Dynamic Scheduling*, Proceedings of the ICEIS'2000, 2nd. International Conference on Enterprise Information Systems, Stafford, UK, 2000, Sharp, B., Cordeiro, J. and Filipe, J. (eds.), 203-208.
20. Reis, J.; Mamede, N.; O'Neill, H., *Locally Perceiving Hard Global Constraints in Multi-Agent Scheduling*, Journal of Intelligent Manufacturing, Vol. 12, No. 2, 2001, 227-240.
21. Reis, J.; Mamede, N., *Multi-Agent Dynamic Scheduling and Re-Scheduling with Global Temporal Constraints*, Proceedings of the ICEIS'2001, 3rd International Conference on Enterprise Information Systems, Setúbal, Portugal, 2001, Miranda, P., Sharp, B., Pakstas, A. and Filipe, J (eds.), Vol. I, 315 321.
22. Sadeh, Norman, *Micro-Oportunistic Scheduling: The Micro-Boss Factory Scheduler*, Intelligent Scheduling, Morgan Kaufman, 1994, Chapter 4.
23. Sycara, Katia P.; Roth, Steven F.; Sadeh, Norman; Fox, Mark S., *Resource Allocation in Distributed Factory Scheduling*, IEEE Expert, February, 1991, 29-40.
24. Weiss, Gerhard (ed.), *Multiagent Systems, A Modern Approach to Distributed Artificial Intelligence*, The MIT Press, 1999.
25. Zweben, Monte; Fox, Mark S., *Intelligent Scheduling*, Morgan Kaufmann Publishers, Inc., San Francisco, California, 1994.