AN APPROACH TO MULTI-AGENT COOPERATIVE SCHEDULING IN THE SUPPLY-CHAIN, WITH EXAMPLES

Joaquim Reis
Departamento de Ciências e Tecnologias de Informação, ISCTE, Avenida das Forças Armadas, 1600 Lisboa, Portugal

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

Abstract: The approach to scheduling presented in this article is applicable to multi-agent cooperative supply-chain production-distribution scheduling problems. The approach emphasises a scheduling temporal perspective, it is based on a set of three steps each agent must perform, in which the agents communicate through an interaction protocol, and presupposes the sharing of some specific temporal information (among other) about the scheduling problem, for coordination. It allows the set of agents involved to conclude if a given scheduling problem has, or has not, any feasible solutions. In the first case, agent actions are prescribed to re-schedule, and so repair, a first solution, if it contains constraint violations. The resulting overall agent scheduling behaviour is cooperative. We also include some results of the application of the approach based on simulations.

1 INTRODUCTION

In this article we present an approach to scheduling in cooperative supply-chain production-distribution scheduling environments, including some unpublished details of the same work.

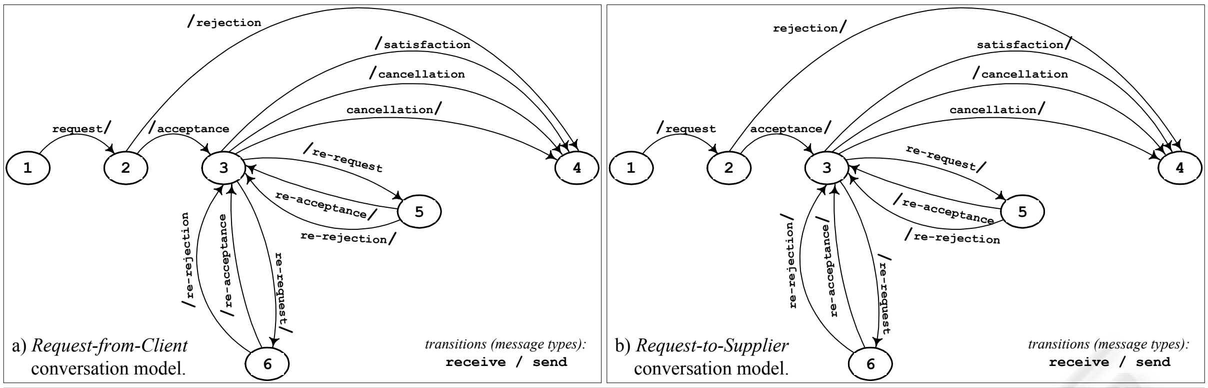
Scheduling is the allocation of resources over time to perform a collection of tasks, subject to temporal and resource capacity constraints (Baker 1974). For classical, Operations Research (OR) based, approaches to scheduling see (Blazewicz 1994); for more modern approaches, Artificial Intelligence (AI) based, see (Zweben 1994), for instance. Planning and coordination of logistics activities (production, distribution) has been the subject of investigation since around 1960, in the areas of OR/Management Science (Graves 1993). More recently, some attention has been paid to scheduling in this kind of environments (*e.g.*, see (Kjenstad 1998) or (Rabelo 1998)).

In our work, the specific logistics context of cooperative supply-chain/Extended Enterprise (EE) (O'Neill 1996) is considered. The EE is usually assumed to be a kind of Virtual Organisation, or Virtual Enterprise, where the set of participant agents (enterprises) is relatively stable (for concepts and terminology see pages 3-14 in (Camarinha-Matos 1999); in this last work, other approaches to scheduling in this kind of context can be found).

The main features of the scheduling problem are: a) decision is decentralised and distributed among multiple autonomous agents, b) problem solution involves communication and cooperation among agents, and c) scheduling can be highly dynamic.

For the modelling of the environment we adopt the AI Multi-Agent Systems paradigm (O'Hare 1996), and consider a network of agents linked through client-supplier relationships and communication channels. *Capacity*, or *manager*, *agents*, manage, each one, the limited capacity of an individual resource, specialised in either *production* or *transportation* or *store tasks*, the last ones with flexible durations. *Producer* and *transporter* agents are both termed *processors*, as their capacity is based on a product rate; *store* agent capacity is based on a product quantity. A *supervision agent* introduces work in the system, and fictitious *retail* and *raw-material agents* define the frontiers of the network with the outside at the downstream and upstream extremes, respectively.

Reis J. (2006).
AN APPROACH TO MULTI-AGENT COOPERATIVE SCHEDULING IN THE SUPPLY-CHAIN, WITH EXAMPLES.
In *Proceedings of the First International Conference on Software and Data Technologies*, pages 325-332
DOI: 10.5220/0001310703250332



request - product request, sent by a client agent to a supplier agent.

rejection - rejection of a previously received product request, sent by the supplier to the client.

acceptance - acceptance of a previously received product request, sent by the supplier to the client.

re-request - re-scheduling request, sent by the supplier (client) to the client (supplier), asking to re-schedule a previously accepted product request to a given due-date.

re-rejection - rejection of a previously received re-scheduling request, sent by the receiver to the sender of the re-scheduling request.

re-acceptance - acceptance of a previously received re-scheduling request, sent by the receiver to the sender of the re-scheduling request.

cancellation - signals giving up a previously accepted product request, sent by the supplier (client) to the client (supplier).

satisfaction - signals delivery of a previously accepted product request, sent by supplier to client at the time of the due-date of the product request.

c) Message types and description.

Figure 1: Conversation model state diagrams, in a) and b), and message types for the agent interaction protocol, in c).

Our temporal scheduling approach is described in (Reis 2001a) for processor agents, (Reis 2001b)] includes a three-step procedure and processor agent re-scheduling cases, and (Reis 2001c) includes store agent re-scheduling cases (our earlier work is referred in these articles). Here we include also some demonstration examples, taken from (Reis 2002), which exposes our whole model and approach. The following sections present: the high level agent interaction protocol used, basic concepts underlying the approach, the steps of the approach, some demonstration examples, and a conclusion.

2 INTERACTION PROTOCOL

In Figure 1 we present the high level agent interaction protocol used by capacity agents, defined through a pair of symmetrical *conversation models* (*Request-from-Client* and *Request-to-Supplier*, shown as state diagrams) in the context of which certain types of messages (also shown and described) can be exchanged.

In Figure 2 we show an example of a network job built by agents of a hypothetical agent network for a scheduling problem. The precedence relationships among the tasks (the arrows forming a tree) reflect the client-supplier relationships among the agents.

A scheduling problem is introduced by the network *supervision agent* g_0, through a global interval $\mathbb{H}$=<RD,DD> (where DD and RD are the

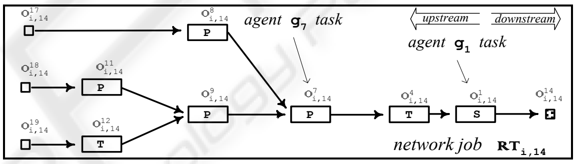


Figure 2: Example of a network job: job $\mathbb{RT}_{i,14}$. The task of a capacity agent g_k for the i_{th} global request to retail agent g_r is denoted by $\mathbb{O}^{k}_{i,r}$ (this task belongs to a network job denoted by $\mathbb{RT}_{i,r}$); P, T and S denote production, transportation and store tasks, respectively; the remaining tasks are fictitious, and belong to retail and raw-material agents (g_{14}, g_{17}, and g_{18}, and g_{19}, which define the limits of the agent network at the downstream and upstream extremes).

global hard temporal limits), and a *global request from outside* $\mathbb{d}$, containing *retail agent* identification, product, quantity and date for satisfaction (the request due-date, dd=TIME($\mathbb{d}$)). In forming the job depicted in Figure 2, retail agent g_{14} first receives from g_0 values of DD and $\mathbb{d}$, then sends a `request` type message to capacity agent g_1, essentially containing $\mathbb{d}$. Starting from g_1, agents in the client-supplier tree then perform a set of communicative actions (sending `request` messages containing local requests to one or more suppliers to ask for task supplies). This upstream propagation of local requests ends with the raw-material agents g_{17}, g_{18} and g_{19} passing to g_0 the local requests of capacity agents g_8, g_{11} and g_{12}, as *global requests to outside*. Subsequently, these agents receive from g_0 the value of RD, and then, `acceptance` messages are propagated downstream, starting from the raw-material agents. For each of the capacity agents, an `acceptance` message confirms its task, which is then scheduled. According to the approach we propose (see ahead), if any agent in the client-supplier tree detects that the problem has no feasible solution, or receives a `rejection` message from a supplier, it sends a `rejection` message to its client and cancels the accepted requests of its suppliers. This would lead to failure in establishing the job, with rejection of the scheduling problem by the agent network as a whole.

After the establishment of a job for a scheduling problem, `re-request`, `re-acceptance` and `re-rejection` messages can be used by the agents to ask for, accept or reject re-scheduling requests to, or from, its client or suppliers, to repair the initial solution schedule, in the case they locally detect temporal or capacity constraint violations. As a last choice, agents can resort to cancellation messages, if a feasible solution cannot be found. This can happen because, as the environment is dynamic, new scheduling problems appear and the individual agent resource capacities are limited.

3 COOPERATIVE APPROACH

The approach we propose is similar to some others that operate through *scheduling by repair* (a first, possibly non-feasible, solution is found which is then repaired through search, if necessary; see (Minton 1992), for instance). It is based on a set of three steps performed by each individual capacity agent for each scheduling problem involving the agent (specifically, involving an agent client-supplier tree that includes the agent), occurring after the problem is known by the agent, *i.e.*, after receiving the respective client `request` message. In the first two steps the approach emphasises scheduling from a temporal perspective and results in an overall agent cooperative scheduling behaviour.

In Figure 3 a set of temporal parameters used in the approach are represented along timelines for a processor agent and for a store agent; processor g_7 and store g_1, involved in the job depicted in Figure 2, are used as an example. Besides the agent task (labelled $\mathbb{O}$) and the requests from the client and to the supplier(s) (labelled $\mathbb{d}$), a set of temporal

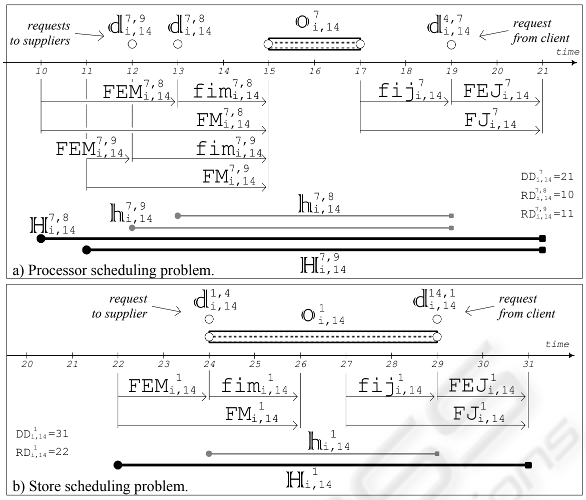


Figure 3: Scheduling problem parameters: a) for processor agent g_7, and b) for store agent g_1 (no relationship is intended for the values in the two timelines). Symbols with two upper indexes refer to two agents, *e.g.*, a request from g_4 to g_7 in job $\mathbb{RT}_{i,14}$ is denoted by $\mathbb{d}^{4,7}_{i,14}$.

intervals (labelled $\mathbb{h}$ and $\mathbb{H}$) and slacks (labelled FJ, FEJ, fij, fim, FEM and FM) is shown. These are defined in the following. In the case of g_7, two suppliers are needed so, there are two $\mathbb{h}$ and two $\mathbb{H}$ intervals:

$$\mathbb{h}^{7,j}_{i,14} = \langle \mathrm{TIME}(\mathbb{d}^{7,j}_{i,14}), \mathrm{TIME}(\mathbb{d}^{4,7}_{i,14}) \rangle \quad (j=8,9)$$

$$\mathbb{H}^{7,j}_{i,14} = \langle \mathrm{RD}^{7,j}_{i,14}, \mathrm{DD}^{7}_{i,14} \rangle \quad (j=8,9)$$

where $\mathrm{DD}^{7}_{i,14}$ is the local hard temporal limit for the end time of task $\mathbb{O}^{7}_{i,14}$ (*i.e.*, considering all agent tasks downstream g_7 scheduled as late as possible), and $\mathrm{RD}^{7,j}_{i,14}$ is the local hard temporal limit for the start time of the $\mathbb{O}^{7}_{i,14}$ concerning to supplier g_j (*i.e.*, considering all agent tasks upstream g_7, starting on supplier g_j, scheduled as early as possible); in determining these local temporal limits, a minimum duration of 1 time unit for store tasks (which have flexible duration) is considered.

For g_7 *internal downstream* and *upstream slacks*:

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

$$\mathrm{fim}^{7,j}_{i,14} = \mathrm{START}(\mathbb{O}^{7}_{i,14}) - \mathrm{TIME}(\mathbb{d}^{7,j}_{i,14}) \quad (j=8,9)$$

For g_7 *external downstream* and *upstream slacks*:

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

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

For g_7 *downstream* and *upstream slacks*:

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

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

(j=8,9)

For g_1 intervals (stores have one $\mathbb{h}$, and one $\mathbb{H}$):

$\mathbb{h}_{i,14}^{1} = \langle \mathtt{TIME}(\mathbb{d}_{i,14}^{1,4}), \mathtt{TIME}(\mathbb{d}_{i,14}^{14,1}) \rangle$

$\mathbb{H}_{i,14}^{1} = \langle \mathtt{RD}_{i,14}^{1,4}, \mathtt{DD}_{i,14}^{1} \rangle$

(with a meaning for $\mathtt{DD}_{i,14}^{1}$ and $\mathtt{RD}_{i,14}^{1,4}$ equivalent to those of g_7). g_1 temporal slacks are defined similarly, except for the internal slacks, which are defined considering $\mathtt{INTERVAL}(\mathbb{O}_{i,14}^{1}) = \mathbb{h}_{i,14}^{1}$, 1 time unit minimum duration for the task and the rest of the effective duration considered as internal slack, being:

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

Additionally, for any g_k, we define the *total slack*:

$\mathtt{FT}_{i,14}^{k} = \mathtt{FJ}_{i,14}^{k} + \mathtt{FM}_{i,14}^{k}$

where $\mathtt{FM}_{i,14}^{k} = \mathtt{FM}_{i,14}^{k,j}$, using for $\mathtt{FM}_{i,14}^{k,j}$ the upstream slack corresponding to the most restrictive $\mathbb{H}_{i,14}^{k,j}$, for processors with more than one supplier.

Assuming agents always maintain non negative internal (fij and fim) slacks and, in the case of store agents, the minimum task duration is 1 time unit, for temporal constraints to be respected, the following conditions must hold. For processor g_7:

$\mathtt{DURING}(\mathtt{INTERVAL}(\mathbb{O}_{i,14}^{7}), \mathbb{h}_{i,14}^{7,j}) \wedge$

$\mathtt{DURING}(\mathbb{h}_{i,14}^{7,j}, \mathbb{H}_{i,14}^{7,j})$

(j=8,9)

Similar conditions must hold for store g_1. The conditions mean that, for an agent scheduling problem, an $\mathbb{O}$ interval must be contained in the $\mathbb{h}$ interval(s), and each $\mathbb{h}$ interval must be contained in the corresponding (same supplier) $\mathbb{H}$ interval. This is equivalent to say that all values for slacks FJ, FM, FEJ and FEM must be non negative. If any of the conditions described doesn't hold, an agent must engage in a re-scheduling activity, involving communicative actions to agree on acceptable temporal values of requests with the client or the supplier(s) and, possibly, re-scheduling actions to correct the temporal position of the task interval (which must be, at least, inside the $\mathbb{H}$ interval). However, before engaging in such activity, an agent must be sure that the problem is time-feasible (otherwise it must be rejected), *i.e.*, that the most restrictive $\mathbb{H}$ interval duration is greater than or equal to the task duration (using for stores a minimum of 1). This is ensured if, for any agent g_k:

$\mathtt{FT}_{i,14}^{k} \geq 0$

In order to be able to determine the values for the end-points of $\mathbb{H}$ intervals, an agent receives from the client (*via* request message) the value of the FEJ slack; then, ensuring non negative internal slack values for the task to be scheduled, it will send to each supplier the supplier FEJ value (*via* request messages); the agent FEM slack values are received from the suppliers (*via* acceptance messages), if they accept the requests; in the case the problem is time-feasible, the agent finally schedules its task and sends to the client the client FEM value (*via* acceptance message). For instance, for agent g_7, the $\mathbb{H}$'s end-points are given by $\mathtt{DD}_{i,14}^{7} = \mathtt{TIME}(\mathbb{d}_{i,14}^{4,7}) + \mathtt{FEJ}_{i,14}^{7}$ and $\mathtt{RD}_{i,14}^{7,j} = \mathtt{TIME}(\mathbb{d}_{i,14}^{7,j}) - \mathtt{FEM}_{i,14}^{7,j}$ (j=8,9); supplier g_j FEJ value is given by $\mathtt{FJ}_{i,14}^{7} + \mathtt{fim}_{i,14}^{7,j}$, and client FEM value by $\underset{j=8,9}{\mathtt{MIN}}(\mathtt{FM}_{i,14}^{7,j}) + \mathtt{fij}_{i,14}^{7}$; retail agent g_{14} passes the value of $\mathtt{DD} - \mathtt{TIME}(\mathbb{d})$ to its supplier capacity agent g_1, as g_1 FEJ value, and each of the raw-material agents g_m (m=17,18,19) passes the value of $\mathtt{TIME}(\mathbb{d}_{i,14}^{k,m}) - \mathtt{RD}$ to its client capacity agent g_k (k=8,11,12), as g_k FEM value.

4 STEPS OF THE APPROACH

The approach we propose is a *minimal approach*, *i.e.*, agents will only modify a scheduling problem solution if it contains constraint violations and, in that case, they operate minimal re-scheduling corrections. The approach is composed of the following sequence of three agent steps:

Step 1, Acceptance and initial solution - If any request to a supplier was rejected, reject the request from the client, cancel the accepted requests to suppliers, and terminate the procedure (with failure). Otherwise, see if the problem is temporally over-constrained; if it is, terminate the procedure

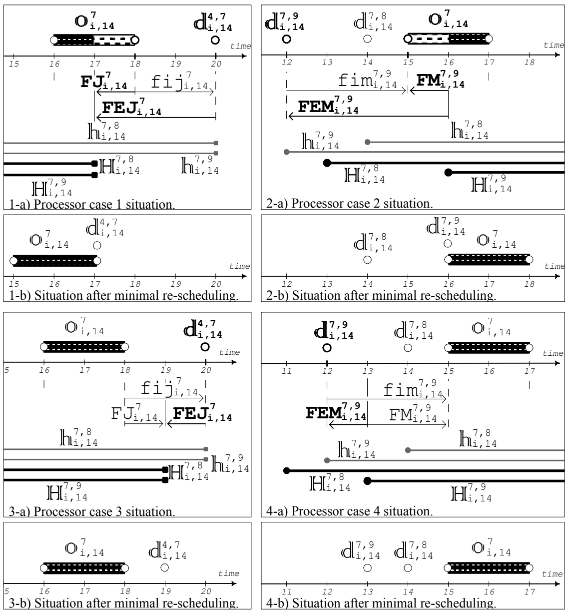


Figure 4: Examples of Step 2 re-scheduling cases 1, 2, 3 and 4, for a processor agent, with situations before, and after, minimal re-scheduling actions.

(with failure) by rejecting the problem, *i.e.*, reject the request from the client and cancel all accepted requests to suppliers; if it isn't, establish an initial solution and proceed to Step 2;

Step 2, Re-schedule to find a time-feasible solution - If the established solution is time-feasible, proceed to Step 3. Otherwise, re-schedule requests, or requests and task, to remove all temporal constraint violations;

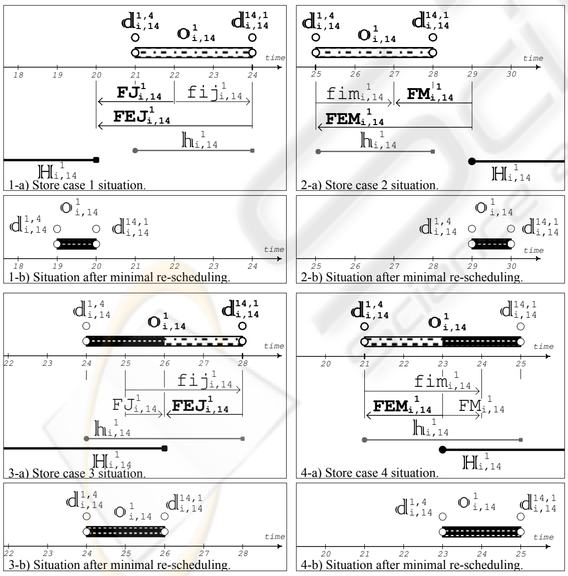
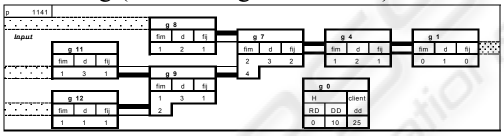


Figure 5: Examples of Step 2 re-scheduling cases 1, 2, 3 and 4, for a store agent, with situations before, and after, minimal re-scheduling actions. In order to detect cases 1 and 3, the minimum duration task interval is considered shifted to the extreme left, and for cases 2 and 4 shifted to the extreme right, relatively to the effective task interval.

Step 3, Re-schedule to find a feasible solution - If the solution is resource-feasible (*i.e.*, it has no capacity constraint violation), terminate the procedure (with success). Otherwise, try to re-schedule to remove all capacity constraint violations, without violating temporal constraints; if this is possible terminate (with success). As a last choice, resort to cancellation, together with task un-scheduling (terminating with failure).



a) Input data.

Table 1	Step 1: Scheduling an Initial Solution				p
dd = 25					1141
H = <0,10>	Messages exchanged among agents				
message type	from	to	contents		
request	g14	g1	dd = 25	FJM / FEJ =	-15
request	g1	g4	dd = 24	FJM / FEJ =	-15
request	g4	g7	dd = 20	FJM / FEJ =	-13
request	g7	g8	dd = 13	FJM / FEJ =	-9
request	g7	g9	dd = 11	FJM / FEJ =	-7
request	g9	g11	dd = 6	FJM / FEJ =	-5
request	g9	g12	dd = 5	FJM / FEJ =	-4
request	g8	g17	dd = 9	FJM / FEJ =	-7
request	g11	g18	dd = 1	FJM / FEJ =	-3
request	g12	g19	dd = 2	FJM / FEJ =	-2
acceptance	g17	g8		FMJ / FEM =	9
acceptance	g18	g11		FMJ / FEM =	1
acceptance	g19	g12		FMJ / FEM =	2
rejection	g11	g9			
acceptance	g12	g9		FMJ / FEM =	4
acceptance	g8	g7		FMJ / FEM =	11
rejection	g9	g7			
rejection	g7	g4			
rejection	g4	g1			
rejection	g1	g14			
cancelation	g7	g8			
cancelation	g8	g17			
cancelation	g9	g12			
cancelation	g11	g18			
cancelation	g12	g19			

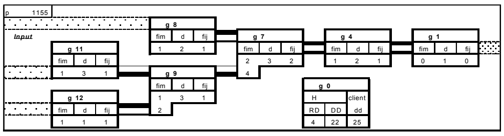
b) Messages exchanged among agents during Step 1. Note the rejection and cancellation messages.

Table 2			Step 1: Scheduling an Initial Solution													p
dd = 25			Local scheduling problem							Temporal slacks						1141
H = <0,10>			(for each capacity agent)													
capacity agent	suppliers	client	dates				task									
			RD	rd	dd	DD	s	d	e	FM	FEM	fim	fij	FEJ	FJ	FT
g8	g17	g7	0	9	13	4	10	2	12	10	9	1	1	-9	-8	2
g11	g18	g9	0	1	6	1	2	3	5	2	1	1	1	-5	-4	-2
g12	g19	g9	0	2	5	1	3	1	4	3	2	1	1	-4	-3	0
g9	g11 g12	g7	-- 1	6 5	11	4	7	3	10	-- 6	-- 4	1 2	1	-7	-6	--
g7	g8 g9	g4	2 --	13 11	20	7	15	3	18	13 --	11 --	2 4	2	-13	-11	--
g4	g7	g1	--	20	24	9	21	2	23	--	--	1	1	-15	-14	--
g1	g4	g14	--	24	25	10	24	1	25	--	--	0	0	-15	-15	--

c) Schedule data in Step 1 (incomplete, as Step 1 was terminated with failure). Note the negative value for total slack FT=-2, detected by agent g_{11}.

Figure 6: Scheduling problem 1141 input and Step 1 data. Step 1, was terminated with failure, in this case.

For a time-feasible scheduling problem, this procedure results in the agents of the client-supplier tree building first, an initial, possibly flawed, solution (in Step 1), which can then be repaired (in Step 2), if necessary. Steps 1 and 2 are oriented to a temporal perspective and concern only to a single problem of an individual agent; Step 3 is oriented to a resource perspective and involves all problems of the agent at Step 3, as all the tasks of the agent compete for its resource capacity.



a) Input data.

Table 1	Step 1: Scheduling an Initial Solution				p
dd = 25					1155
H = <4,22>	Messages exchanged among agents				
message type	from	to	contents		
request	g14	g1	dd = 25	FJM / FEJ =	-3
request	g1	g4	dd = 24	FJM / FEJ =	-3
request	g4	g7	dd = 20	FJM / FEJ =	-1
request	g7	g8	dd = 13	FJM / FEJ =	3
request	g7	g9	dd = 11	FJM / FEJ =	5
request	g9	g11	dd = 6	FJM / FEJ =	7
request	g9	g12	dd = 5	FJM / FEJ =	8
request	g8	g17	dd = 9	FJM / FEJ =	5
request	g11	g18	dd = 1	FJM / FEJ =	9
request	g12	g19	dd = 2	FJM / FEJ =	10
acceptance	g17	g8		FMJ / FEM =	5
acceptance	g18	g11		FMJ / FEM =	-3
acceptance	g19	g12		FMJ / FEM =	-2
acceptance	g11	g9		FMJ / FEM =	-1
acceptance	g12	g9		FMJ / FEM =	0
acceptance	g8	g7		FMJ / FEM =	7
acceptance	g9	g7		FMJ / FEM =	1
acceptance	g7	g4		FMJ / FEM =	7
acceptance	g4	g1		FMJ / FEM =	9
acceptance	g1	g14		FMJ / FEM =	9

b) Messages exchanged among agents during Step 1. There are no rejection or cancellation messages.

Table 2			Step 1: Scheduling an Initial Solution														p
dd = 25 H = <4,22>			Local scheduling problem (for each capacity agent)							Temporal slacks							1155
capacity agent	suppliers	client	dates				task										
			RD	rd	dd	DD	s	d	e	FM	FEM	fim	fij	FEJ	FJ	FT	
g8	g17	g7	4	9	13	16	10	2	12	6	5	1	1	3	4	10	
g11	g18	g9	4	1	6	13	2	3	5	-2	-3	1	1	7	8	6	
g12	g19	g9	4	2	5	13	3	1	4	-1	-2	1	1	8	9	8	
g9	g11 g12	g7	7 5	6 5	11	16	7	3	10	0 2	-1 0	1 2	1	5	6	6	
g7	g8 g9	g4	6 10	13 11	20	19	15	3	18	9 5	7 1	2 4	2	-1	1	6	
g4	g7	g1	13	20	24	21	21	2	23	8	7	1	1	-3	-2	6	
g1	g4	g14	15	24	25	22	24	1	25	9	9	0	0	-3	-3	6	

c) Schedule data after Step 1. No agent detected a negative value for total slack `FT`.

Figure 7: Scheduling problem 1155 input and Step 1 data. Step 1, was terminated with success, in this case.

In Step 2, with temporal constraint violations, there are four possible re-scheduling cases: <u>case 1</u>, negative `FJ` slack; <u>case 2</u>, negative `FM` slack(s); <u>case 3</u>, negative `FEJ` slack, and <u>case 4</u>, negative `FEM` slack(s) (cases 1 and 2 must be tested first by the agent, as they involve also, less critical, negative

Table 3	Step 2: Re-scheduling for a Time-Feasible Solution				p
dd = 25					1155
H = <4,22>	Messages exchanged among agents				
message type	from	to	contents		
re-request	g1	g14	dd = 22	FJ = -3	
re-request	g1	g4	rd = 21	FJ = -3	
re-request	g4	g1	dd = 21	FJ = -2	
re-request	g4	g7	rd = 19	FJ = -2	
re-request	g7	g4	dd = 19	FEJ = -1	
re-request	g9	g11	rd = 7	FEM = -1	
re-request	g11	g9	dd = 7	FM = -2	
re-request	g11	g18	rd = 4	FM = -2	
re-request	g12	g19	rd = 4	FM = -1	

a) Messages exchanged among agents during Step 2.

Table 4			Step 2: Re-scheduling for a Time-Feasible Solution														p
dd = 25 H = <4,22>			Local scheduling problem (for each capacity agent)							Temporal slacks							1155
capacity agent	suppliers	client	dates				task										
			RD	rd	dd	DD	s	d	e	FM	FEM	fim	fij	FEJ	FJ	FT	
g8	g17	g7	4	9	13	16	10	2	12	6	5	1	1	3	4	10	
g11	g18	g9	4	4	7	13	4	3	7	0	0	1	1	6	6	6	
g12	g19	g9	4	4	5	13	4	1	5	0	0	1	1	8	8	8	
g9	g11 g12	g7	7 5	7 5	11	16	7	3	10	0 2	0 0	1 2	1	5	6	6	
g7	g8 g9	g4	6 10	13 11	19	19	15	3	18	9 5	7 1	1 4	1	0	1	6	
g4	g7	g1	13	19	21	21	19	2	21	6	6	1	1	0	0	6	
g1	g4	g14	15	21	22	22	21	1	22	6	6	1	1	0	0	6	

b) Schedule data after Step 2.

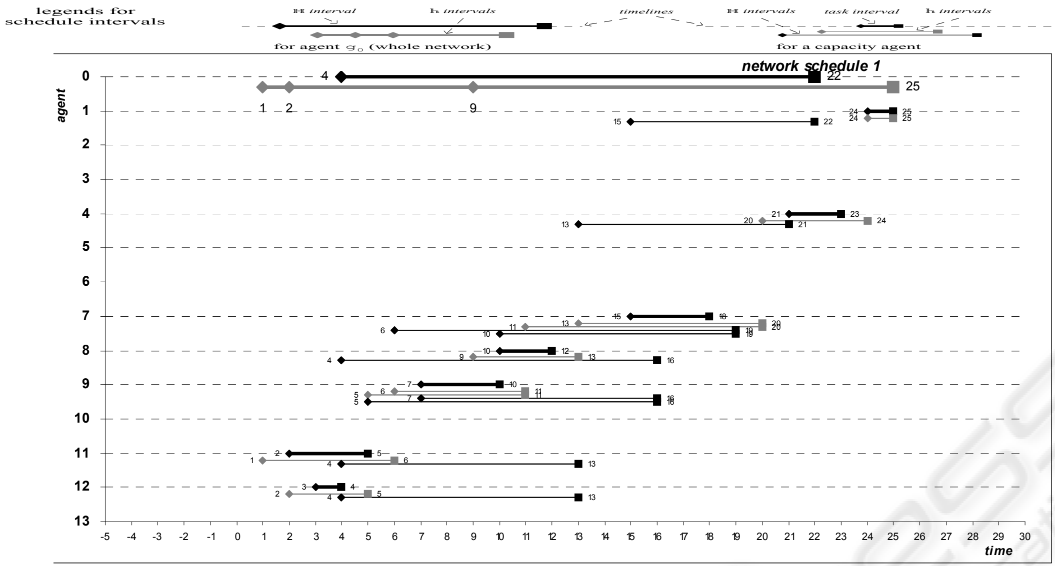
Figure 8: Scheduling problem 1155 Step 2 data. Some requests were re-scheduled to remove temporal constraint violations (the negative slacks in Figure 7-c).

`FEJ` or `FEM`). The cases are depicted in Figure 4 (for processor agents) and Figure 5 (for store agents), together with the appropriate minimal re-scheduling actions, using agents g_7 and g_1 as an example.

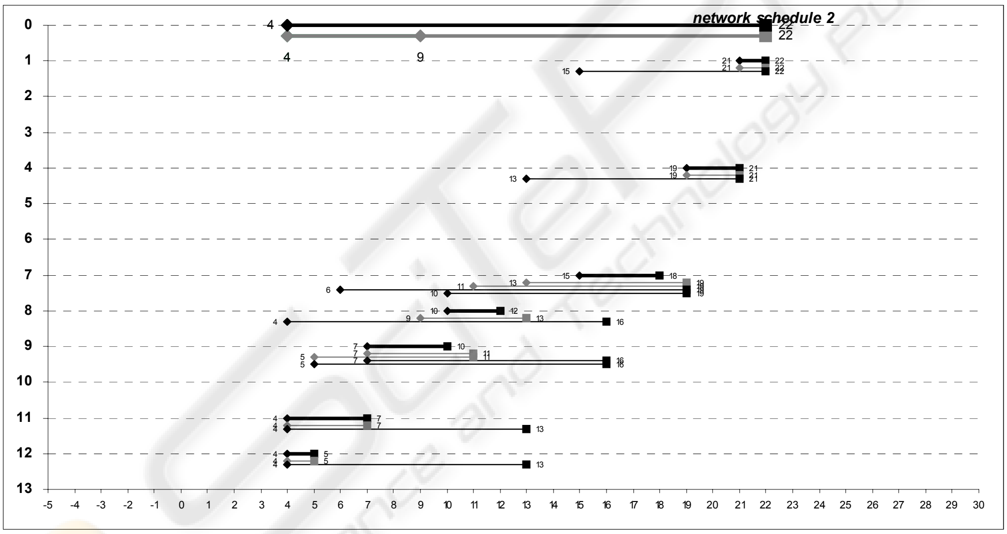
5 EXAMPLES

We now present two network scheduling problem simulation cases, together with the results of the application of the three-step procedure described, assuming the job depicted in Figure 2 (see Figure 6 for the first problem, and Figure 7, Figure 8 and Figure 9 for the second). As input data for problem simulation, global temporal parameters (`RD` and `DD` of global $\mathbb{H}$ interval and date value `dd=TIME(`$\mathbb{d}$`)` of the global request from outside), as parameters for agent g_0, and task durations (`d`) and initial values for internal slacks (`fij` and `fim`, which can be further changed by agents) for each capacity agent are given.

Figure 6 shows the initial data (a), and the messages exchanged (b) and resulting schedule data (c) in Step 1, for the first problem (labelled problem



a) Schedule resulting from Step 1 (built from data in Figure 7-c). Concerning to temporal constraint violation situations experienced by the agents, as shown, agents g_1 and g_4 have a case 1 situation, agent g_7 has a case 3 situation, agent g_9 has a case 4 situation and agents g_{11} and g_{12} have a case 2 situation.



b) Schedule resulting from Step 2 (built from data in Figure 8-b). All temporal constraint violations were removed.

Figure 9: Schedules for problem P 1155: a) after Step 1 (terminated with success), and b) after Step 2.

P 1141). The problem was rejected in Step 1 (with the initiative taken first by capacity agent g_{11}), as it is temporally over-constrained.

Figure 7 shows the initial data (a), and the messages exchanged (b) and resulting schedule data (c) in Step 1, for the second problem (labelled problem P 1155). As Figure 7-c shows, no capacity agent detected a negative value for total slack FT, so the problem is not temporally over-constrained.

As a result, no rejection or cancellation messages are exchanged until the end of Step 1, see Figure 7-b. The resulting network schedule in Step 1 is shown in Figure 9-a. In Step 2, temporal constraint violation situations of case 1 for agents g_1 and g_4, of case 3 for agent g_7, of case 4 for agent g_9, and of case 2 for agents g_{11} and g_{12} are detected. Solution repair is accomplished by agents through inter-agent local request re-scheduling (for all those agents), and additional agent task re-scheduling (only for agents g_1, g_4, g_{11} and g_{12}), according to the minimal actions prescribed. Figure 8 shows the messages exchanged (a) and resulting schedule data (b), and Figure 9-b shows the resulting schedule in Step 2, for this problem. As shown by Figure 9-b, all temporal constraint violations found in the initial solution (Figure 9-a) disappeared.

6 CONCLUSION

We described an approach to multi-agent scheduling in a cooperative supply-chain environment. The approach presupposes the use of an agent interaction protocol (also described), is based on a three-step procedure prescribed for each agent involved in a scheduling problem, and results in an individual cooperative scheduling behaviour. In Step 1 agents detect if the problem is temporally over-constrained and, if it isn't, they schedule an initial, possibly non time-feasible, solution (otherwise, they reject the problem). The exchange of specific temporal slack values, besides product, quantity and due-date information, used as a scheduling coordination mechanism, allows the agents to locally perceive the hard global temporal constraints of the problem, and rule out non time-feasible solutions in the subsequent steps. Each of these pieces of information exchanged in Step 1 corresponds, for a particular agent, to a sum of slacks downstream and upstream the agent in the agent network, and cannot be considered private information of any agent in particular. If necessary, in Step 2, agents repair the initial solution, through re-scheduling, in order to obtain a time-feasible one. In Step 3 any capacity constraint violation must be removed, either through re-scheduling, or by giving up the problem.

No specific details were given for Step 3. In fact, this is the matter of our current and future work. Step 3 can be refined to accommodate additional coordination mechanisms for implementing certain solution search strategies. For instance, strategies based on capacity/resource constrainedness (see [Sycara 1991] or [Sadeh 1994]), to lead the agents on a fast convergence to both time and capacity-feasible solutions, including solutions satisfying some scheduling preferences, or optimising some criteria, either from an individual agent perspective, or from the global perspective of the overall system.

REFERENCES

Baker 1974. Baker, K.R., *Introduction to Sequencing and Scheduling*, Wiley, New York, 1974.

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

Camarinha-Matos 1999. Camarinha-Matos, L.M.; Afsarmanesh, H. (eds.), *Infrastructures for Virtual Enterprises, Networking Industrial Enterprises*, Kluwer Academic Publishers, Dordrecht, The Netherlands, 1999.

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.

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

Minton 1992. Minton, Steven, et al, *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., *Foundations of Distributed Artificial Intelligence*, John Wiley & Sons, Inc., 1996, New York, USA.

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

Rabelo 1998. Rabelo, R.J.; Camarinha-Matos, L.M.; Afsarmanesh, H., *Multiagent Perspectives to Agile Scheduling*, Basys'98 Int. Conf. on Balanced Automation Systems, Prague, Czech Republic, 1998.

Reis 2001a. Reis, J.; Mamede, N.; O'Neill, H., *Locally Perceiving Hard Global Constraints in Multi-Agent Scheduling*, Journal of Intelligent Manufacturing, Vol.12, No.2, April 2001, 227-240.

Reis 2001b. Reis, J.; Mamede, N., *Multi-Agent Dynamic Scheduling and Re-Scheduling with Global Temporal Constraints*, Proceedings of the ICEIS'2001, Setúbal, Portugal, 2001, Miranda, P., Sharp, B., Pakstas, A., and Filipe, J. (eds.), Vol. I, 315-321.

Reis 2001c. Reis, J.; Mamede, N., *Scheduling, Re-Scheduling and Communication in the Multi-Agent Extended Enterprise Environment*, accepted for the MASTA'01 Workshop, EPIA'01 Conference, December, 17-20, 2001, Porto, Portugal.

Reis 2002. Reis, J., *Um Modelo de Escalonamento Multi-Agente na Empresa Estendida*, PhD thesis (in portuguese), ISCTE 2002, Lisbon, Portugal.

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

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

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