A Distributed Scheduling Framework

Carla P. Gomes*
Dept. of Artificial Intelligence
University of Edinburgh
Scotland, United Kingdom

Austin Tate
AIAI
University of Edinburgh
Scotland, United Kingdom

Lyn Thomas
Dept. of Business Studies
University of Edinburgh
Scotland, United Kingdom

Abstract

A distributed problem solving approach to job shop scheduling is described in this paper. The approach views the system as an Organisation. Agents are assigned different roles and functions depending on their position within the structure of the Organisation. In this Organisation, agents of the same level state their interests independently of each other and therefore Conflict is likely to occur. A major thesis of the research reported here is that not only is it important to deal with conflict but also that conflict as a consequence of the scheduling process should be exploited as a way of integrating different scheduling perspectives, as a way of allowing agents to express their own interests independently of each other and, thus, as a way of guaranteeing pluralism by providing agents with both empirical knowledge (heuristics, dispatch rules) and theoretical knowledge (optimal algorithms).

1 Introduction

Scheduling is defined by [1] as the allocation of resources over time to perform a collection of tasks. The job shop scheduling problem consists of assigning times and individual machines to a set of jobs that have to be performed on a finite set of resources, considering some metrics. Each job, also called order, consists of a set of operations related to each other according to a certain process plan that specifies a partial ordering among the operations.

A distributed problem solving approach to job shop scheduling is described in this paper. Hereafter we will refer to it, as well as the system which embodies it, as EXPLICIT.

EXPLICIT can be compared to a hierarchical organisation with three main levels: the Strategic level, the Tactical level and the Operational level. The overall structure of EXPLICIT and an outline of the scheduling process regarding the agents of the systems and their scheduling functions is presented in the next section. A more detailed description of the scheduling process is presented in section 3. Section 3 includes a simple example to illustrate the scheduling process adopted by EXPLICIT. Section 4 presents some results concerning the performance of EXPLICIT. Section 5 summarises the main features of EXPLICIT. A discussion of future research directions is also included in this section.

2 The Overall Structure of the System

Figure 1 displays the overall structure of the job-shop scheduling framework. This structure is inspired by DAS (Distributed Asynchronous System), a system developed at the University of Strathclyde [3], [2]. However, although there are some similarities between EXPLICIT and DAS in terms of the general structure of the system, there are substantial differences in terms of the processes associated with the different agents of the systems, i.e., the functional organisation of the systems, and in terms of the techniques and methods used in both systems[1].

At the Strategic Level, the Strategic Agent is responsible for the whole problem, particularly for assigning work to the Tactical Level and for detecting and solving conflicts that occur from the scheduling decisions performed by the Tactical Agents. At the intermediate level, the Tactical Level, there are two categories of Tactical Agents: the Resource Tactical Agents and the Job Tactical Agents. The Job Tactical Agents are responsible for the jobs. There are as many Job Tactical Agents as the number of jobs to be scheduled. The Resource Tactical Agents are responsible for the aggregate resources. An aggregate resource

*Presently, Carla O. Pedro Gomes is working for Rome Lab., 525 Brooks Rd. Griffiss AFB NY 13441-4505, as an independent consultant.

[1] For a detailed comparison between the two systems see [5].

is a set of identical machines capable of performing a certain operation. There are as many Resource Tactical Agents as the number of aggregate resources.

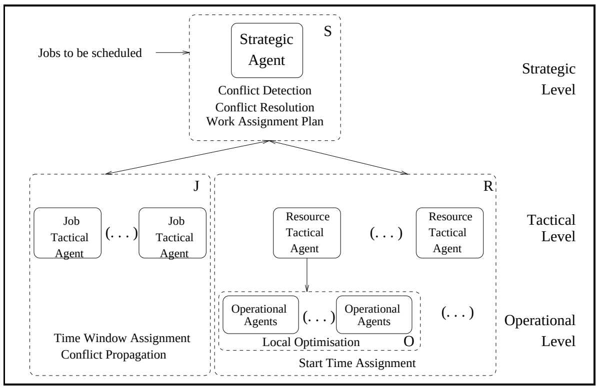

Figure 1: The agents of EXPLICIT and the schedule generation process

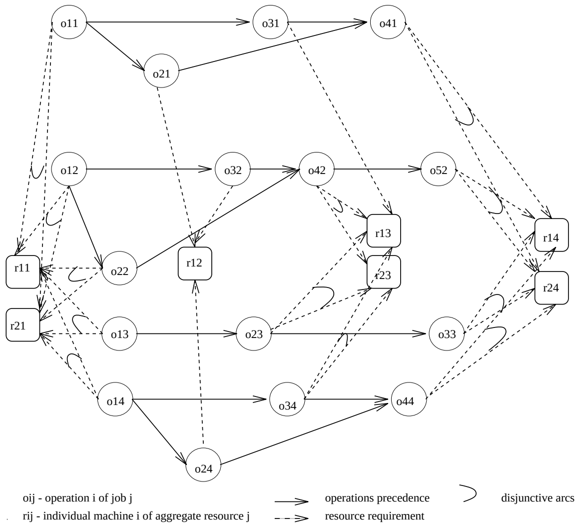

Figure 2: The representation of the job shop scheduling problem

Figure 1 also depicts the process of schedule generation. The jobs enter the system via the Strategic Agent. The Strategic Agent sends each job to the corresponding Job Tactical Agent for the assignment of time windows to the operations that constitute the job. The Job Tactical Agents assign time windows to their operations independently of each other. This task is performed assuming unlimited resources. The Job Tactical Agents send the results of the time window assignment to the Strategic Agent.

Once the Strategic Agent receives the time windows for all the operations of all the jobs, it assigns work to the Resource Tactical Agents. Operations to be performed on the same aggregate resource are sent to the respective Resource Tactical Agent with the respective time windows. The Resource Tactical Agents assign individual machines and start times to the operations to be performed on their aggregate resources. Analogously to the Job Tactical Agents, the Resource Tactical Agents perform their scheduling tasks independently of each other. If the system is provided with Operational Agents, the Resource Tactical Agents assign work to each of their Operational Agents. Each Operational Agent receives the information concerning the set of operations to be performed on its individual machine. The Operational Agents are responsible for locally improving the schedules proposed by their superior Tactical Agent. The Operational Agents carry out their scheduling tasks independently of each other. Operational Agents send their schedules to the Strategic Agent. If the system does not include Operational Agents, the Resource Tactical Agents send their schedules directly to the Strategic Agent. When the Strategic agent gets the times assigned to the operations by different RTAs, it is responsible for: (1) identifying the conflicts that exist among the schedules proposed by the different Tactical Agents; (2) generating Plans to solve the detected conflicts; (3) generating Plans to coordinate the scheduling activity of the Tactical Agents, in particular the redifinition of time windows of the operations that have to be rescheduled.

This framework is very suitable for parallel implementation since the different Tactical and Operational Agents perform their tasks independently of each other and so they can perform their tasks concurrently.

3 Scheduling with Explicit

3.1 Formulation of the problem

It is useful to represent the whole problem as a graph $\mathcal{G} = (\mathcal{O}, \mathcal{R}, \mathcal{A}, \mathcal{E})$, with node set $\{\mathcal{O}, \mathcal{R}\}$, and ordinary (conjunctive) arc set $\mathcal{A}$ and disjunctive arc set $\mathcal{E}$. Figure 2 illustrates this graph.

The node set $\mathcal{O}$ of $\mathcal{G}$, $\mathcal{O} = \{o_{ij} : i \in N_j, j \in N\}$, N the set of indices of the jobs to be scheduled and N_j the set of indices of the operations of a given job j, corresponds to the operations of the jobs to be scheduled (represented by a circle in the graph). The node set $\mathcal{R}$ of $\mathcal{G}$, $\mathcal{R} = \{r_{ij} : i \in M_j, j \in M\}$, M the set of indices of the different aggregate resources or machine types and M_j the set of indices of the individual machines of a given aggregate resource j, corresponds to the different machines on which the different jobs have to be scheduled (represented by a square in the graph). The arc set $\mathcal{A}$ of $\mathcal{G}$, $\mathcal{A} = \{(o_{ij} \;\; o_{lj}) : i, l \in N_j, j \in N\}$, corresponds to precedence relations between operations of the same job (represented by full arrows in the graph). The disjunctive arc set $\mathcal{E}$, $\mathcal{E} = \{(o_{ij} \;\; r_{lk}) : i \in N_j, j \in N, l \in M_k, k \in M\}$, denotes the alternative machines where a given operation can be performed (represented by dashed arrows in the graph).

The set of arcs $\mathcal{A}$ decomposes the graph $\mathcal{G}$ into subgraphs (O_j, A_j), where $O_j = \{o_{ij} : i \in N_j, j \in N\}$, $\mathcal{O} = \bigcup(O_j : j \in N)$ and $A_j = \{(o_{ij} \;\; o_{lj}) : i, l \in N_j, j \in N\}$, $\mathcal{A} = \bigcup(A_j : j \in N)$. Each subgraph (O_j, A_j) corresponds to a job j, $j \in N$. The set of disjunctive arcs $\mathcal{E}$ decomposes the graph $\mathcal{G}$ into subgraphs (O^k, R_k, E_k), where $O^k = \{o_{ij} : (\exists \; (o_{ij} \;\; r_{lk})), i \in N_j, j \in N, l \in M_k, k \in M\}$[2], $O = \bigcup(O^k : k \in M)$, $R_K = \{r_{ik} : i \in M_k, k \in M\}$, $R = \bigcup(R_k : k \in M)$, one for each aggregate resource or machine type, $\mathcal{M}$ the set of indices of the aggregate resources. The subgraph (O^k, R_k, E_k) corresponds to the problem associated with the aggregate resource $k \in M$, i.e., the set of individual machines of a certain type, and the operations that have to be scheduled on it.

The *job shop scheduling or machine sequencing problem* can be defined as follows. Times (start and finish times) and individual machines have to be assigned to each operation of a set of jobs, satisfying a set of constraints and considering a certain objective. Referring to the figure 2, the *job shop scheduling problem* can be stated as how to partition the node set $\mathcal{O}$ into subsets such that operations that are members of the same subset are assigned to the same individual machine r_{ij}, with a given start time and finish time, satisfying all the constraints and considering a certain objective (typically the minimisation or maximisation of a certain function).

The approach adopted in EXPLICIT is "divide and conquer", i.e., the decomposition of the whole problem into smaller and more manageable problems in order to reduce the overall computational complexity of the scheduling problem. Different agents are assigned different (sub)problems. Each Job Tactical Agent is responsible for assigning time windows to the operations of its job. The Job Tactical Agent responsible for job j is denoted by JTA_j. The problem associated with JTA_j is represented by the subgraph (O_j, A_j). Each Resource Tactical Agent is responsible for assigning start times and individual machines to the operations to be performed on its aggregate resource. The Resource Tactical Agent responsible for the aggregate resource k is denoted by RTA_k. The problem associated with RTA_k is represented by the subgraph subgraph (O^k, R_k, E_k). The Strategic Agent is responsible for the whole problem, in particular for coordinating the scheduling tasks of the Tactical and Operational Agents.

3.2 The Newspaper Example

Alan (A), Carla (C) , Flavio (F) , Ian (I), Nelson (N) and Suresh (S) share a flat. Every Saturday they have delivered at their flat two copies of the following newspapers: the European (E), the Financial Times (F), the Guardian (G), the Scotsman (S). Each flatmate gets up at a certain time and insists on reading all the papers in a particular order (precedence constraints). Each flatmate wants to leave the flat by a given time (due-time). Table 1 summarises the data for the example. In this example, each reader represents a job. The availability time for each job corresponds to the time the reader gets up. The due-time of a job corresponds to the latest time the reader wants to leave the flat. Operations of a job correspond to the act of reading a newspaper by a reader. The precedence constraints, i.e., the order that each reader wants to read the newspaper, are reflected in the order of the columns in table 1. The Financial Times and the Guardian are delivered at 8.30 a.m (510), the European at 8.40 (520) and the Scotsman at 8.45 (525). Newspapers correspond to resources in a job-shop scheduling problem. Each copy of a particular newspaper corresponds to an individual machine[3]. The agents for the newspaper problem are: the Strategic Agent (SA), responsible for the whole scheduling problem; the Job Tactical Agents (JTAs), one per job (reader), Alan, Carla, etc; the Resource Tactical Agents (RTAs), one per type of resource (newspaper), the European, the Financial Times, etc; and the Operational Agents (OAs), one per individual machine i.e., one per copy of a newspaper.

[2] Note that O^k denotes the set of operations that are assigned to the same aggregate resource k, while O_k is the set of operations that belong to the same job k.

[3] This example takes inspiration from [4]

R	up	out	$1st$		$2nd$		$3rd$		$4th$	
A	510	630	F	60	G	30	E	2	S	5
C	525	660	E	5	G	15	F	10	S	30
F	585	690	G	5	S	5	E	5	F	60
I	570	690	S	90	F	1	G	1	E	1
N	540	660	F	30	S	10	E	4	G	10
S	525	640	G	75	E	3	F	25	S	10

Table 1: The data for the newspaper reading problem (in minutes)

3.3 Functions, Roles and Algorithms - The Newspaper Example

In this section the scheduling tasks performed by each agent are analysed from a functional point of view. Some of the algorithms assigned to the problem solving agents are also outlined. The scheduling process is illustrated with the newspaper example.

At the beginning of the scheduling process all the jobs are in conflict since operations do not have start times assigned to them. The Strategic Agent (SA) initiates the scheduling process by sending all the operations to the respective Job Tactical Agenta (JTAs) for time window assignment. The Strategic Agent passes all the necessary information to each JTA for time windows assignment. Each JTA assigns time-windows to its operations solving a critical path method problem (see e.g., [8] and [6]) independently of the other JTAs. At this stage the availability of resources is not considered or, in other words, resources are considered unlimited. SA collects all the data from the different JTAs and sends the operations' time-windows to the corresponding Resource Tactical Agents (RTAs). The first time the SA performs this operation all the operations with the respective time-windows are sent to the corresponding RTAs in order to have start times assigned to them. The time suggested by SA for each operation is the earliest start time assigned by the corresponding JTA to that operation. Information on the slack allowed for each operation is also sent to the RTAs. RTAs assign start times to the operations to be performed on their resources independently of each other. In order to assign start times and individual machines to its operations, each RTA solves the "Assignment Based Algorithm" (see [5]) which involves the following steps: (1) the generation of a graph of the operations assigned to RTA and the partition of its nodes into levels[4]; (2) for each level, RTA solves an assignment problem to assign machines (and start times) to each operation of that level. Operations that cannot be assigned to a machine are delayed to the next level. After solving the "Assignment Based Algorithm", operations have start times and individual machines assigned to them. At this stage RTA sends the operations to the corresponding Operational Agent (OA) responsible for an individual machine. Operational Agents are responsible for optimising the schedule of the operations assigned to them. Figure 3 illustrates

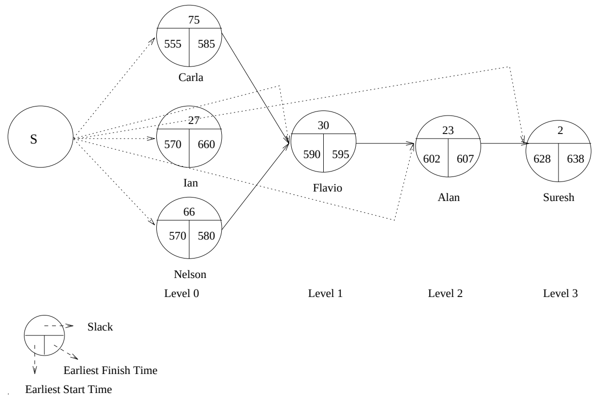

Figure 3: The Resource Tactical Agent problem. Graph S_{sco} for the Scotsman

the graph corresponding to the problem assigned to RTA responsible for the Scotsman (RTA_{sco}). Figure 3 also illustrates the different levels of the graph.

R	SA			RTA		OA		#
	ST	FT	SLK	ST	FT	ST	FT	
A	602	607	23	602	607	602	607	1
C	555	585	75	555	585	555	585	1
F	590	595	30	590	595	595	600	1
I	570	660	27	570	660	570	660	2
N	570	580	66	607	617	585	595	1
S	628	638	2	628	638	628	638	1

Table 2: The times assigned by the different agents to each reader (1st iteration - in minutes; ST - Start Time; FT - finish Time; SLK - slack)

Each Operational Agent is responsible for locally optimising the schedule proposed by the Resource Tactical Agent. The algorithm provided to each Operational Agent minimises maximum lateness of the jobs

[4] The level of a node is the length, in number of arcs, of the longest path from the source node (S) to that node.

assigned to it [7][5]. It is an implicit enumeration algorithm that uses a branch-and-bound technique. In the case of the Scotsman, there are two Operational Agents (OAs): the OA responsible for copy number 1 (OA_{sco1}) and the OA responsible for copy number 2 (OA_{sco2}). Since only one reader was assigned to OA_{sco2}, there is no optimisation process for OA_{sco2}. In the new assignment performed by the OA_{sco1}, Nelson reads the Scotsman at 585, instead of 607 as proposed by the RTA, and the reader Flavio reads the Scotsman at 595, instead of 590 as proposed by the RTA. Table 2 displays the different times assigned to each operation by each agent. The other RTAs (and OAs) associated with the other newspapers schedule their readers using a scheduling process identical to the one described for the Scotsman.

SA (Strategic Agent) detects the conflicts generated from the independent assignment of times performed by each RTA. SA is essentially provided with a rule based system in order to perform its role and functions (see [5] for more details). SA is responsible for: (1) identifying the conflicts that exist among the schedules proposed by the different Tactical Agents; (2) generating Plans to solve the detected conflicts; (3) generating Plans to coordinate the scheduling activity of the Tactical Agents. The role of SA is very crucial but very simple. A conflict occurs whenever an operation starts later than the earliest start time that was last assigned to it by the corresponding Job Tactical Agent. The idea of conflict is that the current schedule might have to be revised, since all the operations of the same job that come after the operation involved in the conflict need to have their time windows revised. Nevertheless, the fact that an operation is involved in a conflict does not mean that the operation is late. Its new start time might still be within its initial time window. Conflict propagation is done starting with the affected operations with the earliest earliest start time. As a result of the conflict propagation, SA generates a plan for conflict resolution. This plan contains all the operations that are involved in the conflict propagation, either because they belong to a job that had some of its time windows changed or because they are assigned to a resource that had to be rescheduled. As an example, regarding the Scotsman, Nelson is in conflict since the start time that was assigned to it was 585, rather than the earliest start time proposed to it, 570. Table 3 shows the new operations' time windows that were obtained by the propagation of conflicts, regarding the Scotsman. In this case, the operations that have new time windows are: Carla, Flavio and Nelson. All of the other RTAs have their operations and respective time windows revised. Once again they perform the assignment of start times and machines to their operations.

Reader	*Start Time*	*Finish Time*	*Slack*
Alan	602	607	23
Carla	580	610	50
Flavio	605	610	15
Ian	570	660	27
Nelson	585	595	51
Suresh	628	638	2

Table 3: The information sent by the SA to RTA_{sco} (second iteration)

The process goes on until a schedule without conflicts is reached. That means that all the operations have start times and that the start times correspond to the last earliest start time proposed to that operation by the SA. Due date relaxation is implicit in EXPLICIT. If operations cannot start within their initial time windows their due dates are automatically relaxed, as little as possible. That means that if EXPLICIT cannot not find a solution considering the initial due date contraints, a solution is given relaxing some of those constraints. Table 4 summarises the solution for the newspaper problem in terms of start times assigned to each reader for each newspaper. Notice that this solution does not relax any due date. This solution was reached after 6 cycles,[6] assuming a sequential implementation.

Reader	*1st*		*2nd*		*3rd*		*4th*	
Alan	F	510	G	570	E	600	S	615
Carla	E	525	G	530	F	570	S	580
Flavio	G	600	S	610	E	615	F	620
Ian	S	595	F	685	G	686	E	687
Nelson	F	540	S	585	E	595	G	600
Suresh	G	525	E	600	F	603	S	628

Table 4: The start times for the newspaper reading problem (in minutes)

[5] $L_i = C_i - d_i$ where L_i - lateness of job J_i, C_i - completion time of job J_i, d_i - due date of job J_i.

[6] A cycle corresponds to the following scheduling tasks: analysis of the problem (detection of conflicts) and revision of time windows (SA and JTAs), assignment of start times to operations by RTAs and OAs.

4 Perfomance of EXPLICIT

In order to do a preliminary evaluation of EXPLICIT a battery of 54 cases was generated (see [5] for details). Since EXPLICIT was tuned to minimize lateness[7], we chose lateness and tardiness[8] as performance measures to evalutate the performance of EXPLICIT. In this paper we include graphs displaying the behavior of EXPLICIT regarding "Number of Tardy Jobs", one of the most important measures of tardiness. For more information regarding other measures see [5]. Since EXPLICIT was inspired by DAS we would like to compare its performance with DAS's performance. Unfortunately there are no data available, regarding DAS's performance. We analyzed the performance of EXPLICIT considering two versions: (1) without Operational Agents (we call that version EXPLICIT-OAS-OUT) ; (2) with Operational Agents (we call that version EXPLICIT-OAS-IN). We also compared the performance of EXPLICIT against four popular dispatch rules: (1) Shortest Operation First (SOF); (2) Maximum Operation First (MOF); (3) Minimum Slack First (MINSLK) ; and (4) Maximum Slack First (MAXSLK).

EXPLICIT-OAS-IN outperforms EXPLICIT-OAS-OUT with respect to all the measures of tardiness considered. The outperformance of EXPLICIT-OAS-IN over EXPLICIT-OAS-OUT is not a surprise since EXPLICIT-OAS-IN is provided with the Operational Agents whose task is the optimization of the schedules proposed by the Resource Tactical Agents regarding lateness. The downside is that the scheduling process takes longer when Operational Agents are included in the system.

The comparison of EXPLICIT (more correctly, EXPLICIT-OAS-IN) with the four dispatch rules in terms of "Number of Tardy Jobs" is shown in figures 4, 5, 6, and 7[9].

From the analysis of the graphs, it is clear that EXPLICIT outperforms the four dispatch rules in terms of $NumTardy$. The graphs display, for each case, the reduction in the number of tardy jobs due to EXPLICIT divided by the number of jobs. The dispatch rule MAXSLK performs particularly bad. MAXSLK reduces the number of tardy jobs only in one case, while EXPLICIT reduces the number of tardy jobs in 38 cases, and the magnitude of the reduction in the number of tardy jobs due to EXPLICIT is very significant for each case. From the four dispatch rules, MINSLK is the dispatch rule that performs better, though clearly worse than EXPLICIT. The number of cases for which EXPLICIT reduces the number of tardy jobs is 15, while MINSLK only reduces the number of tardy jobs in 5 cases. Additionally, the magnitude of reduction in the number of tardy jobs due to EXPLICIT for each case

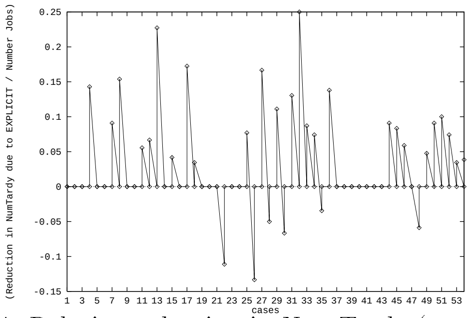

Figure 4: Relative reduction in NumTardy (EXPLICIT vs. SOF)

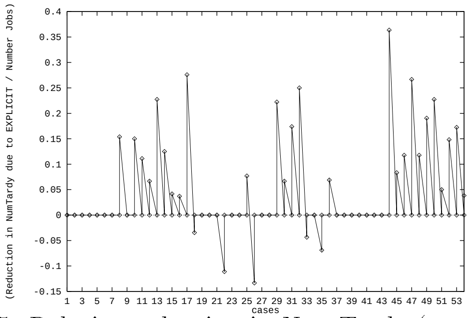

Figure 5: Relative reduction in NumTardy (EXPLICIT vs. MOF)

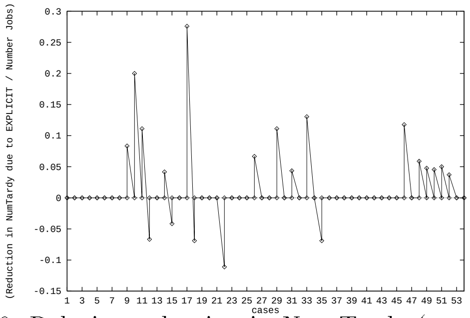

Figure 6: Relative reduction in NumTardy (EXPLICIT vs. MINSLK)

[7]The Operational Agents were provided with an algorithm that minimizes lateness. Furthermore, the objective function provided to the RTAs also tries to minimize lateness.

[8]Lateness of a job i (L_i), is the difference between its completion time and its due-date. Note that when the job is early, i.e., when it completes before its due date, L_i is negative. It is often more useful to have a variable which, unlike lateness, only takes non-zero values when a job is *tardy*, i.e., when it completes after its due-date. Tardiness of a job i (T_i), is $max\{L_i, 0\}$. Maximum tardiness of a set of jobs S ($MaxTard$) is $max_{i \in S} T_i$, while the number of tardy jobs ($NumTardy$) is $\sum_{i=1}^{n} x_i$, where x_i is 1 if $T_i > 0$, 0 otherwise.

[9]On all the comparison graphs the following criterion is adopted: for the cases where the performance of the first compared system is better than the performance of the second compared system, a positive value is displayed; a negative value corresponds to the situations where the second compared system performs better than the first one.

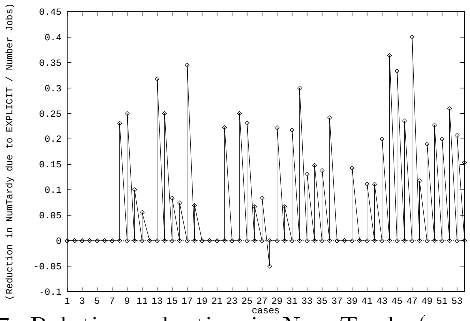

Figure 7: Relative reduction in NumTardy (EXPLICIT vs. MAXSLK)

is greater than the reduction in the number of tardy jobs due to MINSLK, in particular for the case 17. The comparison of EXPLICIT with the four dispatch rules in terms of "Maximum Tardiness" also shows that EXPLICIT performs better than the dispatch rules. The outperformance of EXPLICIT is even more noticeable in terms of "Maximum Tardiness" than is terms of "Number of Tardy Jobs".

The current implementation was designed as a proof of concept rather than an attempt at efficiency. Extensive debugging aids, record keeping, including several sorting routines as part of the record keeping process, hamper its efficiency. Furthermore, the entire system was run under an interpreted LISP. Rule of thumb estimates for a compiled version are at least a thirtyfold increase in execution speed, compared to the interpreted version. Nevertheless, for the sake of reference, the maximum value of the cpu time was 1716 when solving a 30 jobs problem, each job with 6 operations and the total number of 36 machines and for the version that includes the Operational Agents. Regarding the number of iterations required to generate a solution, it is remarkable that, for both version, the *mean* and *range* of the number of cycles required to achieve a solution is very small, 7 and 7 for EXPLICIT-OAS-OUT and 7.648 and 10 for EXPLICIT-OAS-IN.

5 Conclusions

In this paper we described EXPLICIT, a distributed framework to perform scheduling. EXPLICIT can be compared to a hierarchical organisation with three main levels: the Strategic level, the Tactical level and the Operational level. EXPLICIT has a very rich model for resource allocation. We analyzed the performance of EXPLICIT considering two different versions (with and without Operational Agents) and against four popular dispatch rules. The results are very encouraging.

There are a number of ways in which the research reported in this paper can be extended, some of which are briefly outlined below. EXPLICIT is conceptually distributed but implemented on a sequential machine. A natural extension to EXPLICIT is to implement it in a physically distributed environment. The results obtained in terms of the number of cycles and the cpu time required to achieve a solution are very encouraging, in particular the small magnitude of the average number of cycles required to generate the final schedule. It provides an indication that the performance of EXPLICIT could be improved if a physically distributed environment was adopted. Another way of extending EXPLICIT is to refine its resource model. We think that the resource model of EXPLICIT is very rich and it can be used as a module for resource allocation in other systems. One of the refinements that we are currently exploring is the usage of different utility functions.

References

[1] K. R. Baker. *Introdution to Sequencing and Scheduling.* Wiley, 1974.

[2] Peter Burke. *Scheduling in Dynamic Environments.* PhD thesis, University of Strathclyde, 1989.

[3] Peter Burke and Patrick Prosser. A Distributed Asynchronous System for Predictive and Reactive Scheduling. Technical Report AISL-42-89, University of Strathclyde, October 1989.

[4] S. French. *Sequencing and Scheduling: An Introduction to the Mathematics of the Job-Shop.* Ellis Horwood, Chichester, England, 1982.

[5] Carla P. Gomes. *Achieving Global Coherence by Exploiting Conflict: A Distributed Framework for Job Shop Scheduling.* PhD thesis, University of Edinburgh, 1992.

[6] K. G. Lockyer. *Critical path analysis and other project network techniques.* London Pitman, 1984.

[7] Graham McMahon and Michael Florian. On Scheduling with Ready Times and Due Dates to Minimize Maximum Lateness. *Operations Research*, 23(3):475–481, 1975.

[8] R. J. Willis. Critical path-analysis and resource constrained project scheduling - theory and practice. *European Journal of Operational Research*, 21(2):149–155, 1985.