ABA: An Assignment Based Algorithm for Resource Allocation

Carla Pedro Gomes
Rome Laboratory / C3CA*
525 Brooks Rd.
Rome Laboratory NY 13441-4505
gomes@ai.rl.af.mil

Julie Hsu
Department of Computer Science
Louisiana State University
Baton Rouge, LA 70803
hsu@bit.csc.lsu.edu

Abstract

In this paper we describe an approach to allocating a set of tasks to a set of resources or processors. The novelty of the approach has to do with the way decisions are performed. Rather than making one decision about one resource (or one task) at a time, several decisions concerning multiple resources and multiple tasks are made at a time. The algorithm incorporates the formulation of the assignment problem. Furthermore, knowledge about temporal constraints between activities is exploited to improve the computational efficiency of the algorithm. The algorithm is very flexible, allowing for the incorporation of different types of constraints as well as the consideration of non-equivalent resources.

1 Introduction

Most resource allocation problems are NP-complete in the strong sense [6]. The practical consequence of dealing with an NP-complete problem is that realistic dimensions of it cannot be optimally solved in a "reasonable" amount of time. Nonetheless, despite the difficulties, solutions have to be found for real world problems and therefore heuristic approaches have to be adopted, with some sort of guarantees on the quality of the solution.

In this paper we address the problem of allocating a set of tasks to a set of identical resources or processors. We describe an approach to resource allocation based on a framework proposed by [10]. This approach was inspired by previous work in crew scheduling ([9, 8, 1]. Its novelty has to do with the way decisions are performed. Rather than making one decision about one resource (or one task) at a time, like most AI-based approaches to scheduling ([13, 24, 23, 17, 21, 20, 16]), several decisions concerning multiple identical resources and multiple eligible tasks are made at a time. The algorithm incorporates the formulation of the assignment problem [19]. Furthermore, knowledge about temporal constraints between activities is exploited to improve the efficiency of the algorithm. The algorithm is very flexible, allowing for the incorporation of different types of constraints as well as the consideration of non-equivalent resources. Since resource allocation problems have been extensively addressed in factory scheduling we will adopt its terminology. Nevertheless, the issues discussed in this paper also apply to planning and scheduling domains in general, in which resource allocation is relevant.

2 Formulation of the problem

Figure 1 depicts the problem that we address in this paper. Operations denoted by $O_i(i = 1 \ldots n)$ have to be assigned to a set of m identical, non-equivalent resources, denoted by $R_j(j = 1 \ldots m)$, some evaluation criteria. can be represented by a graph, $A = (\{O, R\}, E)$, in which $\{O, R\}$ is the set of nodes of the graph, O the set of operations (or jobs or tasks) and R the set of resources (or machines). E is the set of disjunctive arcs. An arc between an operation and a resource means that the operation can be performed on that resource. Each resource can process only a single operation (or task) at a time. By identical resources we mean that all the resources can perform the same type of operation. Resources are non-equivalent in the sense that an operation may require different processing times depending on which resource it is assigned to. Each operation has a time window defining its ready time and due time as well as a processing time (duration) associated with each resource it can potentially be assigned to. Different types of constraints on operations, including but not limited to precedence constraints, may also be defined. The model also allows constraints excluding certain operations from being performed on certain resources. In the example shown in Figure 1, there are three identical resources and nine operations to be assigned to them. Operation O_2 can only be assigned to resource R_1 or R_2, and operations O_4 and O_6 can only be allocated to resources R_2 and R_3, respectively. O_8 can only be assigned to resource R_1 or R_3, and operation O_9 can only be assigned to resource R_2 or R_3. All of the other operations can be assigned to any of the resources. This figure does not display constraints between operations.

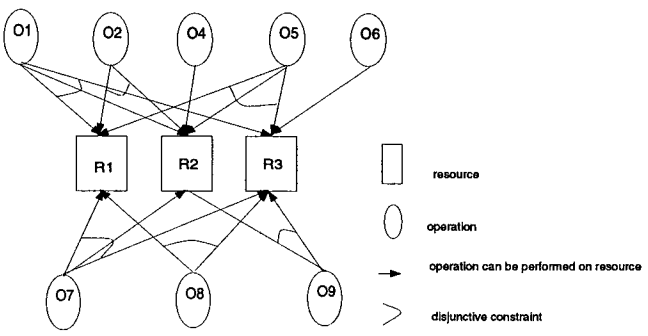


Figure 1: The resource allocation problem

3 Assignment Based Algorithm

The disjunctive graph depicted in Figure 2 gives another perspective of this problem. We refer to this graph as graph $D = (\{O, R\}, C, E)$. The difference between graph D and the graph shown in Figure 1 is the set of arcs represented by dashed arrows denoting precedence constraints between operations (C). Each arc represented by a dashed arrow means that the operations that it links can be performed on the

*Carla O. Pedro Gomes works for Rome Laboratory as a research associate.

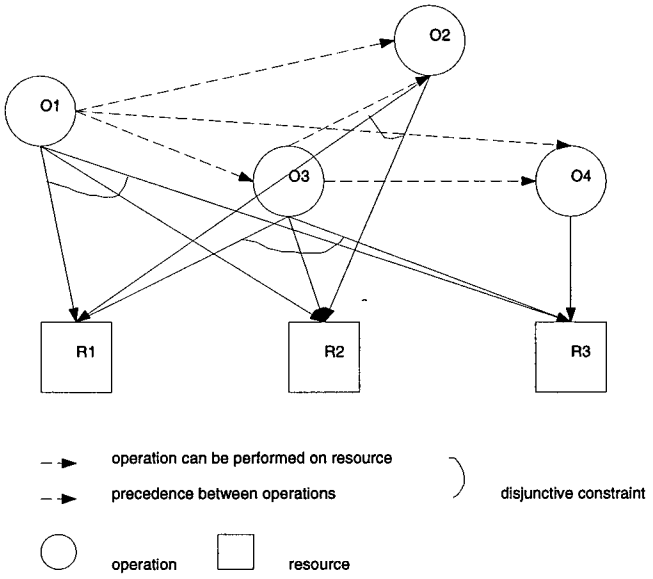


Figure 2: Graph D

same resource in sequence. An arc between two operations only reflects the constraints between the operations it links. In other words, if there is an arc between operation A and B and another arc between operation B and C, we can infer that A and B can be performed in sequence on the same resource, B and C can also be performed in sequence on the same resource, but we cannot infer that A, B, and C can be performed in sequence on the same resource. In addition to the precedence constraints that are part of the initial definition of the problem, the arc set C includes a (generated) special type of precedence constraint between operations - time precedence. There exists an arc denoting temporal precedence between operation A and operation B if the earliest start time of operation B is greater than or equal to the earliest finish time of operation A. Arcs between operations defining time precedence relationships are added to the initial formulation of the problem. We refer to the subgraph $P = (O, C)$, obtained from D by removing the node set R and the arc set E, as precedence graph P.

In the following paragraphs we describe an algorithm to assign operations to resources. We refer to it as the Assignment Based Algorithm (ABA). ABA consists of two main steps. The first step consists of partitioning the node set O of graph P into levels. The second step of ABA consists of solving a sequence of assignment problems (see e.g. [19]) to allocate operations to the resources.

3.1 Partitioning of O into levels

Taking into consideration the precedence relationships between operations, we can partition the node set O of graph P into levels. The level of a node is the length of the longest path (in number of arcs) to reach that node. The calculation of the level of a node is quite trivial considering that graph P is a directed graph without circuits (see, e.g., [11]). Figure 3 illustrates the concept of levels of a graph.[1]

[1]Node S is a dummy node that represents the start. Dummy arcs between each operation and S represent precedence of the start over every operation.

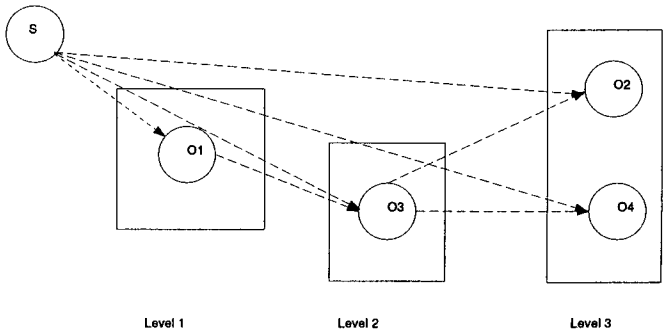


Figure 3: Levels of a graph

3.2 The Assignment Problem

After partitioning the set of operations O of graph P into levels, we solve at least as many assignment problems (see e.g. [19]) as the number of levels of the graph. Figure 4 depicts an assignment problem, with three node sets. The assignment problem is a particular case of the matching problem which involves a complete bipartite graph with n node sets, i.e., n source nodes and n target nodes. Associated with each arc of the graph is a utility, u_{ij}. Solving an assignment problem entails finding a one-to-one pairing of source and target nodes such that the total utility of the assignment is maximized. If the original graph is not complete, i.e., if there is an imbalance in the number of source or target nodes, dummy nodes and arcs with utility of $-\infty$ are added to make the graph complete. Any source node assigned to a dummy target is considered unassigned. Likewise, any target node assigned to a dummy source is considered unassigned. In terms of resource allocation, we can view source nodes as resources and target nodes as operations we want to allocate the resources to.

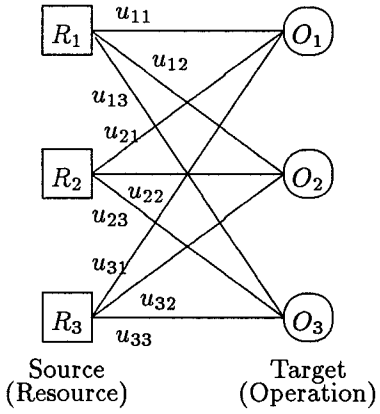


Figure 4: An assignment problem with three node sets

The power of incorporating an assignment problem algorithm (e.g. the efficient Hungarian Algorithm [19]) as the underlying driver of our algorithm lies in several aspects. First of all, each assignment decision involves the consideration of several eligible operations and the entire set of resources, rather than making a decision about a single operation (or resource) at a time. AI approaches to resource allocation emphasize decisions involving a single operation and/or resource at a time (e.g., [22, 4, 20]). Even though our approach

is greedy and heuristic in nature, at each decision point the utility of the assignments is maximized, considering all the candidate operations and the entire set of resources. Furthermore, when solving each assignment problem different types of constraints between operations and between operations and resources can be incorporated, which makes the algorithm very flexible.

3.3 Assignment Based Algorithm ABA

Given a problem in which n operations are to be scheduled on m resources:

Create precedence graph P and partition the set of operations O into levels
Begin Loop
If set O of P is empty, terminate and return the current schedule
Else
- Solve an assignment problem with the resource set R and the current set of candidate operations, i.e., operations at the current level;
- Filter operations from the set of recommended allocations, if necessary;
- Update P: remove all scheduled operations, propagate constraints (delay any unscheduled operations to the next level), and increment the level

End Loop

Initially, graph P is generated and its operations are partitioned into levels. After the partitioning of the operations into levels, the method iterates through each level solving an assignment problem. The assignment problem is defined considering all the resources and the operations at the current level. An arc between an operation and a resource means that the operation can be assigned to that resource considering all the constraints, in particular other operations already assigned to that resource, and assuming the start time to be the current earliest start time of the operation. The utility function for the assignment problem favors allocations that finish earlier. There is one characteristic of the assignment problem that causes the global results to be rather poor. Each application of the assignment problem returns a set of "recommended" allocations that maximizes the overall utility given the current set of resources and operations, which means that it will greedily allocate all resources that can be allocated. But it is often the case that we do not want to force an operation to be allocated to a resource just because the resource is available. To control this behavior, we perform a filtering operation on the set of recommended allocations that marks certain operations as "delayable" if, after a small look-ahead, there appears to be another allocation that would result in a better finish time than the suggested finish time. After performing the filtering operation on the set of recommended allocations, any leftover operations not scheduled or scheduled to dummy resources or marked for delay in the current iteration are added to the set of candidate operations at the next level, and propagation updates their earliest start times.[2] This process continues until all operations have been assigned to individual resources.

In a simplified variation of ABA, which we refer to as ABA-II, time precedence between operations is not taken into account when performing the assignment of operations to resources. Therefore, when using ABA-II, there is no need to generate time precedence arcs between operations. At each iteration all unscheduled operations are candidates to be assigned to the resources, rather than only considering the operations of the current level. Again, the filtering operation is performed to see if any operations would yield better results if delayed to the next level. Any operations not assigned at the current iteration or assigned to dummy resources or marked for delay are candidates in the next iteration. The time performance of ABA-II is worse than ABA. With ABA-II, at each iteration we solve larger assignment problems than with ABA, which only considers the operations of the current level for the assignment problem. Since the algorithm for solving the assignment problem has complexity $O(n^3)$, using ABA rather than ABA-II has considerable impact in terms of reducing the run time of the overall algorithm for resource assignment. The quality of the schedules produced by ABA is nearly identical to that produced by ABA-II (see Section 5).

In the next section a simple example illustrating how ABA works is presented.

4 An Example

Carla, Julie, Jackie, Karen, Nancy, and Tonette share an apartment in Rome. Every Sunday they like reading the New York Times. The New York Times is delivered to their apartment at 8:30. They also have access to the New York Times on-line. Each person wants to read the newspaper - hardcopy or on-line version - within a time window and for a certain period, depending on the type of medium. Table 1 summarizes the data for the example.[3]

Reader	*Gets*	*Finish*	*Reading Time*	
	Up At	At	Hardcopy	On-line
Tonette	602	625	-	5
Carla	525	630	75	50
Nancy	590	610	20	5
Julie	570	650	60	70
Karen	570	636	10	10
Jackie	628	658	30	10

Table 1: The data for the newspaper problem

In this example, the reading of the newspaper by one person corresponds to an operation. The time each reader gets up is the ready time of the associated operation. The time each reader wants to finish her reading corresponds to the due-time of the operation. Newspapers correspond to resources in a job-shop scheduling problem. In this case we assume that we have two resources that can perform the same operation but with different execution times depending on the medium (hardcopy vs. on-line). Notice that we can actually specify different durations for the same operation depending on the resource, in this case each reader can have different reading times depending on the medium they use to read the New York Times. In the example Tonette does not read the hardcopy.

Figure 5 displays graph P for the newspaper problem. Note that in Figure 5 the operations of P are partitioned into levels according to time precedence between operations[4].

[2] The new earliest start time of an operation delayed to a new level is the finish time of the resource that finishes the earliest.

[3] Time is converted into sequential minutes (e.g., 525 = 8 hours * 60 minutes + 45 minutes).

[4] Node S is a dummy node that represents the start. Dummy arcs between each operation and S represent precedence of the start over every operation.

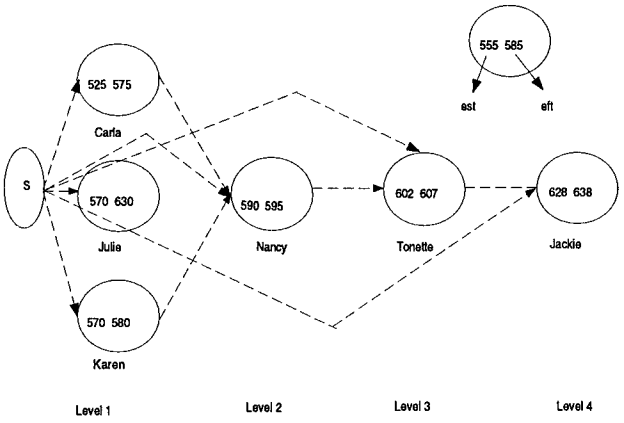


Figure 5: Graph P

Figure 6 depicts the different iterations of ABA for the newspaper problem. In iteration 1, Carla, Julie, and Karen are the candidates for reading the New York Times. Since the utility function favors assignments that finish earlier, Carla and Karen are assigned to the online and hardcopy versions respectively. A quick look-ahead is performed to see if delaying either of the recommended allocations will result in a better finish time: if we allocate Carla to the online version, her finish time is 525+50=575. Now suppose we delay Carla to the next level and assign her after Karen to the hardcopy. Her finish time for the hardcopy would be 580 (Karen's finish time) + 75 (Carla's time spent reading the hardcopy) = 655. The recommended finish time of 575 is better than this delayed finish time, so we take the allocation as recommended. The same look-ahead comparison is performed for Karen, and in this example, the recommendation also stands. Since Julie was not assigned to either version of the New York Times at iteration 1, her reading is delayed to the next level. Note that the new earliest start time for Julie has changed to the finish time of the resource that finishes the earliest in the previous iteration, i.e., the time that Carla finishes reading the online version (575). Julie and Nancy now compete for the resources at iteration 2. They both get assigned at this iteration. Again, the look-ahead operation is performed, and the recommended allocation is worth keeping. Note that at iteration 3 there is no arc between the hardcopy version and Tonette since Tonette only wants to read the on-line version of the New York Times. Finally, Jackie is the only candidate at the last iteration, and gets assigned to the online version. In the resulting schedule, all deadlines are met and the final completion time (makespan) is 640.

Experiments with ABA and ABA-II are discussed in the next section.

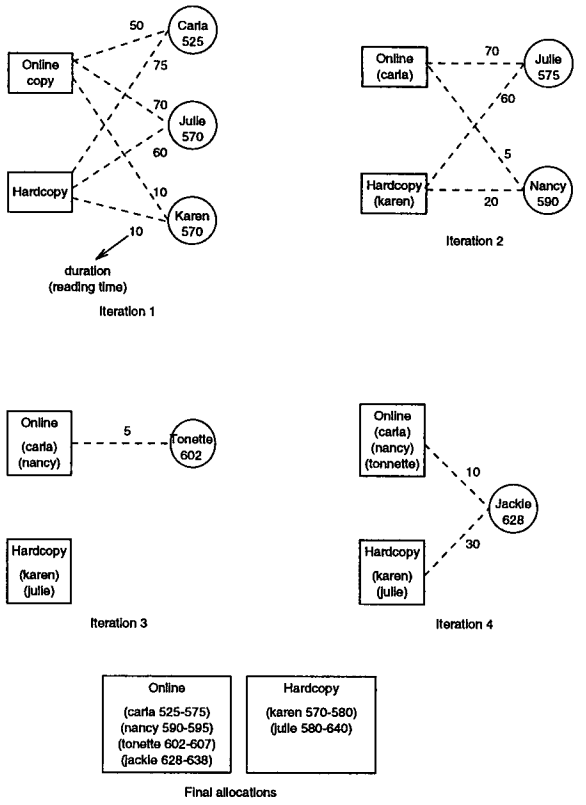


Figure 6: The Assignment Based Algorithm for the Newspaper Example

5 Experimental Results

5.1 Evaluation Criteria

In order to evaluate the performance of ABA, we employed a test set of 60 randomly generated scheduling problems, varying certain parameters of the problem. For each problem, the number of jobs was fixed at 200 with the number of resources increasing in size by increments of 5, i.e., either 5, 10, 15, or 20 resources, the idea being that the smaller the number of available resources, the increased chance of bottlenecks and hence a larger makespan (completion time of the last job on all resources). Three deadline ranges were considered: tight (T), median (M), and wide (W); the tighter a job's due date, the greater the chance that the job will be tardy. Interleaving the different deadline ranges and the number of resources yields 12 categories of scheduling problems: $T5$, i.e., tight with 5 resources, and similarly for $T10$, $T15$, $T20$, $M5$, $M10$, $M15$, $M20$, $W5$, $W10$, $W15$, and $W20$. Five problems in each of the 12 categories were randomly generated resulting in a total of 60 test cases. More broadly, there were 20 problems in each deadline range T, M, and W. Constraints excluding certain operations from being processed by certain resources were randomly generated in each test problem. In addition, because ABA is capable of handling non-equivalent resources, different processing times (durations) for each combination of operation and resource were generated.

[3] identify three classes of performance criteria: completion time based, due date based, and flow time based measures. Completion time based criteria involve the makespan, which is important under the assumption that the system cannot be shut down until all resources have finished processing. Completion time is also a measure of overall resource utilization. Due date based measures evaluate the overall tardiness, the total number of tardy jobs, etc. These are usually quite important factors since there is often a cost associated with delayed production and deliverables. Of less importance in some domains are the flow time based measures which include measures such as total resource idle time.

Because the use of dispatch rule heuristics is quite common for resource allocation purposes in real world factory scheduling environments and due to the lack of results using other approaches that handle non-equivalent identical resources, the performance of ABA was compared against popular dispatch

rules. Dispatch heuristics are often employed in real-world applications because of their computational simplicity. The set of jobs are first sorted according to the chosen dispatch rule; e.g., with the earliest due date first heuristic (EDD), activities are sorted by due date in ascending order. A schedule is then created by "dispatching" the jobs one at a time in the sorted order until all jobs are assigned to individual resources. Dispatch heuristics use very limited knowledge about inter-job relationships, only the ordering of the jobs according to the selected rule. There is a vast literature about dispatch rules (e.g., [7, 14, 18, 5, 12, 2, 15]). We compared ABA against four popular dispatch rules: Earliest Due Date (EDD), Shortest Processing Time (SPT), First Come First Served (FCFS), and Minimum Slack (MINSLACK). In order to cope with different durations of an operation depending on the resources, we implemented the dispatch rules in the following manner. With regards to selecting the next job to be dispatched for EDD, the jobs are sorted by due date in ascending order; for FCFS, the jobs are sorted by ready time in ascending order; for MINSLACK, jobs are sorted by $slack = duedate - eft$ in ascending order; for SPT, jobs are sorted by minimum duration in ascending duration. Concerning the task of selecting of which resource to assign the operation to, our criterion was to always choose the resource that results in the earliest finish time for that operation, the same criterion that we used in ABA.

5.2 Discussion

With the problem test set, we compared ABA against several popular dispatch heuristics as well as its own variation ABA-II. The dispatch heuristics include Earliest Due Date First (EDD), Minimum Slack First (MINSLACK), First Come First Served (FCFS), and Shortest Processing Time First (SPT). The percent reduction in makespan[5] and the reduction in the number of tardy jobs[6] using ABA versus each of the other dispatch heuristics and also ABA-II were calculated and plotted (see Appendix). The results of all 60 test problems are interpreted in the following manner: positive results indicate that ABA created a schedule with a better value with respect to the corresponding performance criteria.

Comparing the makespan, we see that ABA on the whole performed better than each of the dispatch rules. There were a few problems in which the schedule generated by the dispatch rules resulted in a smaller makespan than that generated by ABA. But there were many more cases in which the difference between the makespans was markedly significant in favor of ABA. The reduction in makespan induced by ABA is over 20% in some cases. The outperformance of ABA in comparison to the dispatch rules is also apparent with respect to the number of tardy jobs. A pattern of behavior is evident by looking at both these measures. Recall that for each deadline range *T*ight, *M*edian, and *W*ide, there were 4 sets of 5 problems each, with each set increasing in the number of resources. The makespan graphs reveal that ABA's ability to generate schedules with smaller makespan improves as the number of resources increases (less constrained problems). In the comparison of number of tardy jobs, the cases where the performance of ABA equals that of the dispatch rules are problems in which a feasible solution was found in which all due dates were met. These cases, taken with the makespan plots, show strong evidence of the power of ABA over the dispatch heuristics.

The performance of ABA-II was nearly identical to that of ABA-I with respect to makespan and number of tardy jobs, confirming that the quality of the resulting schedules can be attributed to the incorporation of the assignment algorithm and that the partitioning scheme is more of a contribution to runtime efficiency. The last graph in the Appendix depicts the time performance of ABA-I, ABA-II, and MINSLACK. The time performance of the other dispatch rules is similar to that of MINSLACK. As one would expect, the dispatch rules take less time than ABA due to their simplicity. Nevertheless, it is difficult to outperform dispatch rules systematically and therefore they are frequently used in real world applications. ABA is systematically faster than ABA-II, as expected, due to the fact that at each iteration smaller assignment problems are solved. Another interesting aspect is the fact that as the problems become less constrained (those with more resources), both ABA algorithms take a noticeably less amount of time to execute. From these factors, we can see the benefits of employing a partitioning scheme into our algorithm.

6 Conclusions

In this paper we describe ABA, an approach to allocating a set of identical resources to a set of jobs which creates higher quality schedules than those generated by dispatch heuristics, yet still without the undesirable consequences of exponential algorithms. ABA incorporates an efficient assignment algorithm that allows several allocation decisions to be made at a time. Furthermore, by exploiting knowledge about time precedences among jobs contending for a resource, smaller problems are solved at each iteration, which reduces the runtime of the algorithm without affecting the quality of the schedules. ABA is very flexible, allowing for the incorporation of different types of constraints as well as the consideration of non-equivalent resources. ABA can be incorporated into a system responsible for maintaining a resource-based perspective. We compared ABA against four popular dispatch rules. The results are very positive. ABA consistently outperforms all the dispatch rules in our experiments especially with regards to resource utilization (makespan) and tardiness (reduction in the number of tardy jobs). In addition, as problems become less constrained, i.e., those involving a larger resource set, ABA is clearly able to generate schedules in less amount of time.

Acknowledgements

We would like to thank Claude Le Pape and Robert P. Graham for helpful comments on an earlier draft of this paper.

Appendix

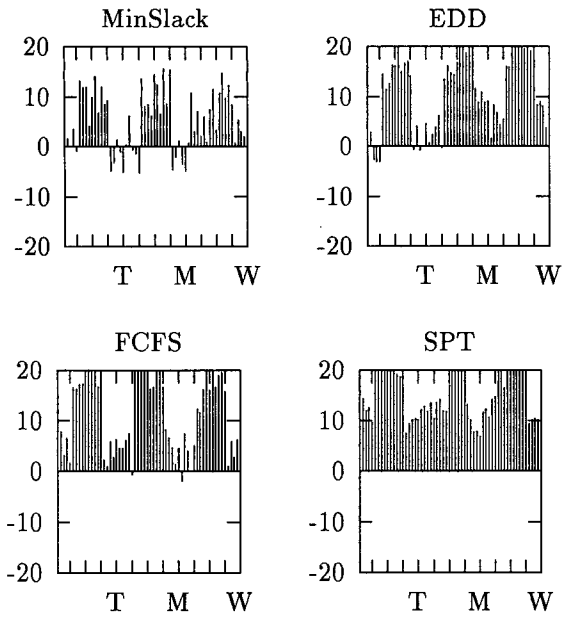
Each of the following graphs shows the results of comparing ABA to the indicated dispatch rule for the 60 scheduling problems in the test set. The 60 problems each consist of 200 operations and are categorized as follows: Problems #1-5 are of type $T5$ (tight deadline range with 5 resources), problems #6-10 of type $T10$ (tight with 10 resources), problems #11-15 of type $T15$ (tight with 15 resources), problems #16-20 of type $T20$ (tight with 20 resources), and so forth for the *M*edian and *Wide* deadline ranges: $M5$, $M10$, $M15$, $M20$, $W5$, $W10$, $W15$, and $W20$. The results of all 60 test problems are interpreted in the following manner: positive results indicate that ABA created a schedule having a better value with respect to the corresponding performance criterion - the positive value corresponds to the reduction induced by ABA

[5]The reduction in makespan was calculated with respect to the makespan of the dispatch rule involved in the comparison.

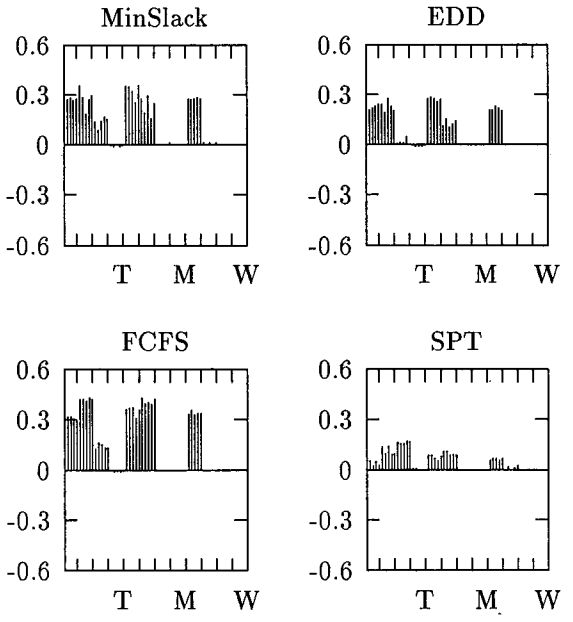
[6]The reduction in number of tardy jobs was calculated with respect to the total number of jobs.

regarding that measure, e.g., percentage reduction in completion time.

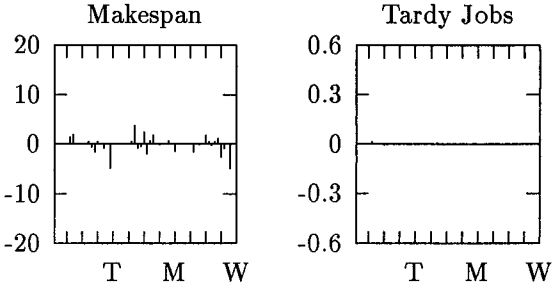
Makespan (Completion Time) Comparison
x axis = Problem Set (each consisting of 5 problems)
y axis = % reduction induced by ABA in makespan with respect to the makespan of the corresponding dispatch rule.



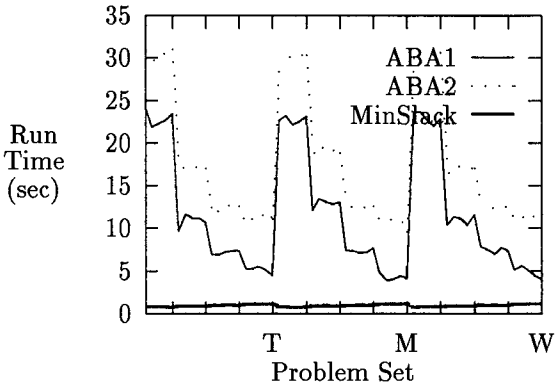
Comparison of Number of Tardy Jobs
x axis = Problem Set (each consisting of 5 problems)
y axis = reduction induced by ABA in the number of tardy jobs with respect to the total number of jobs.



Comparison to ABA-II
x axis = Problem Set (each consisting of 5 problems)
y axis = % reduction induced by ABA against ABA-II.



Run Time Comparison



References

[1] M. Ball, L. Bodin, and R. Dial. A Matching Based Heuristic for Scheduling Mass Transit Crews and Vehicles. *Transportation Science*, 17:4–31, 1981.

[2] J. H. Blackstone, D. T. Phillips, and G. L. Hogg. Heuristics in Job Shop Scheduling. *International Journal of Production Research*, 20(1):27–45, 1982.

[3] T. Cheng and C. Sin. A State-of-The-Art Review of Parallel Machine Scheduling Research. *European Journal of Operational Research*, 47(3):271–292, 1990.

[4] Anne Collinot, Claude Le Pape, and Gerard Pinoteau. SONIA: A Knowledge-Based Scheduling System. *Artificial Intelligence in Engineering*, 3(2):86–94, April 1988.

[5] E. W. Davis and J. H. Patterson. A Comparison of Heuristics and Optimum Solutions in Resource Constrained Project Scheduling. *Management Science*, 21(8):944–955, 1975.

[6] M. R. Garey and D. S. Johnson. *Computers and Intractability.* W. H. Freeman and Co., San Francisco, 1979.

[7] W. S. Gere. Heuristics in Job Shop Scheduling. *Management Science*, 13(3):167–190, 1966.

[8] Carla O. Pedro Gomes. Escalonamento de Tripulacoes - Geracao de Cadeias de Voo para uma Frota de Medio Curso. Master Thesis, Universidade *Técnica* de Lisboa, 1987.

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

[10] Carla O. Pedro Gomes. An Assignment Based Algorithm for Resource Allocation. To appear as a technical report of Rome Laboratory, 1994.

[11] Gondran and M. Minoux. *Graphs and Algorithms.* John Wiley & Sons, 1984.

[12] S. C. Graves. A Review of Production Scheduling. *Operations Research*, 29(4):646–675, 1981.

[13] Mark D. Johnston and Steven Minton. Analyzing a heuristic strategy for constraint-satisfaction and scheduling. In M. Fox and M. Zweben, editors, *Intelligent Scheduling.* Morgan Kaufmann, 1994.

[14] Rinnooy Kan. *Machine Scheduling Problems: Classification, Complexity and Computations.* Nijhof, The Hague, 1976.

[15] Kurtulus and Narula. Multi Project Scheduling - Analysis of Project Performance. *IIE Transactions*, 17(1):58–66, 1985.

[16] S. Minton, M.D. Johnston, A.B. Philips, and P. Laird. Solving large-scale constraint satisfaction and scheduling problems using a heuristic repair method. In *Proceedings of AAAI-90*, Boston, MA, 1990.

[17] N. Muscettola. Scheduling by iterative partition of bottleneck conflicts. In *Proc. 9th IEEE Conference on AI Applications*, Orlando, FL, 1993.

[18] S. S. Panwalkar and Wwafik Iskander. A Survey of Scheduling Rules. *Operations Research*, 25(1):45–61, 1977.

[19] C. H. Papadimitriou and K. Steiglitz. *Combinatorial Optimization - Algorithms and Çomplexity.* Prentice-Hall, Inc., 1982.

[20] Norman Sadeh. *Look-ahead Techniques for Micro-oportunistic job shop scheduling.* Phd thesis, School of Computer Science, Carnegie Mellon University, 1991. (CMU-CS-91-102).

[21] Stephen Smith and Cheng-Chung Cheng. Slack-Based Heuristics for Constaint Satisfaction Scheduling. In *Proceedings of The Eleventh National Conference on Artificial Intelligence*, pages 139–145, 1993.

[22] Stephen F. Smith, Mark S. Fox, and Peng Si Ow. Constructing and Maintaining Detailed Production Plan: Investigations into the Developmemt of Knowledge-Based Factory Scheduling Systems. *AI Magazine*, 7(4):45–61, Fall 1986.

[23] M. Zweben, M. Deale, and R. Gargan. Anytime rescheduling. In *Proceedings of the Workshop on Innovative Approaches to Planning, Scheduling, and Control*, pages 251–259. DARPA, 1990.

[24] Monte Zweben, Brian Daun, Eugene Davis, and Michael Deale. Scheduling and Rescheduling with Iterative Repair. In M. Fox and M. Zweben, editors, *Intelligent Scheduling.* Morgan Kaufmann, 1994.