An Experimental Analysis of Bottleneck-Centered Opportunistic Scheduling

Nicola Muscettola

CMU-RI-TR-93-06

The Robotics Institute
Carnegie Mellon University
Pittsburgh, Pennsylvania 15213

March 1993

This work was sponsored in part by the National Aeronautics and Space Administration under contract # NCC 2-707, the Defense Advanced Research Projects under contract #F30602-91-F-0016, and the Robotics Institute.

Table of Contents

List of Figures

List of Tables

Abstract

Bottleneck-centered, opportunistic scheduling is an effective problem solving paradigm based on a fundamental Analyze-Focus-Decide cycle. Opportunistic schedulers have shown better overall performance than other scheduling systems adopting different paradigms. However, in order to make more significant progress, we need to better understand how the different phases of the opportunistic cycle interact during problem solving. This paper reports an extensive analysis of a specific embodiment of the opportunistic scheduling paradigm, the Conflict Partition Scheduling procedure. By analyzing the role of each phase of CPS, proposing variations and anticipating expected effects on the performance of the algorithm, we aim to develop a better understanding of the importance of the different phases and of their interactions. The hypotheses on the performance of CPS are then verified in an experimental analysis. The results of this study provide hard evidence that supports the fundamental assumptions made by opportunistic scheduling.

1 Introduction

Bottleneck-centered, opportunistic scheduling is a problem solving paradigm that addresses a scheduling problem's specific sources of complexity. When compared with other systems, schedulers that follow this paradigm have shown superior performances [1] [11] [10]. However comparing the global performance of one scheduler against another can only show that the combination of techniques assembled in one system is superior to those assembled in another. The questions that remain open are: (1) what the importance of each technique is; (2) how the different techniques interact to carry out the computation. Answers to these questions are crucial if we are to understand the principles behind the functioning of a scheduling system. This understanding is essential in order to guide the design of better schedulers.

This paper reports an empirical study on Conflict Partition Scheduling (CPS) [9] [10], a scheduling methodology that implements the opportunistic scheduling paradigm. Previous research [10], which involved comparisons on a suite of benchmark scheduling problems, has shown that CPS performs better than other state of the art schedulers [11] [7]. In this paper we take a more in-depth look at each step of the methodology. First we analyze the role of each step of CPS. Then we discuss modifications of each step and make hypotheses on their effects on the performance of the algorithm. These hypotheses are based on our qualitative understanding of the importance of the different phases and of their interactions. Finally, we verify the qualitative hypotheses against the results of an empirical analysis. Each step of CPS has a nature similar to that of corresponding steps in other opportunistic schedulers; also, the performance hypotheses are applicable to comparable modifications in other scheduling algorithms. As a consequence, we believe that the results of this empirical study can be applied to other opportunistic schedulers and can therefore provide the basis for a better understanding of the entire class of opportunistic schedulers beyond CPS.

CPS is one possible implementation of the opportunistic scheduling paradigm. When given a network of activities and the set of resources needed to carry out the activities, an opportunistic scheduler builds a schedule (i.e., an assignment of time and resources to each activity that avoid resource over-subscription) by repeatedly applying the following basic *opportunistic scheduling cycle*:

1. **Analyze**: analyze the current problem solving state;
2. **Focus**: select one or more activities that are expected to participate in a critical interaction among problem constraints;
3. **Decide**: add constraints to avoid negative interactions among critical activities.

Although opportunistic schedulers differ on the specific techniques used to implement each step [12] [2] [11] [1], some fundamental characteristics are common across all of them. In all schedulers, for example, the **Analyze** step consists of building estimates of demand/supply ratios for the different resources and/or activities. These estimates, or capacity analyses, are usually conducted on relaxed versions of the problem obtained by dropping some of its original constraints. During the **Focus** step, opportunistic schedulers use bottlenecks as the primary means to indicate and select critical interactions. While the exact definition of a bottleneck varies among different opportunistic schedulers, all agree on relating this concept to a resource and time interval with a high demand/supply ratio. Critical activities are usually defined as those that are likely to request the use of a bottleneck. The constraints posted during the **Decide** phase allow the arbitration among conflicting capacity and time requests. This is the phase where opportunistic schedulers differ the most, with respect to the type of constraint posted (assigning a value to a variable versus imposing a precedence among activities) and to the granularity of the decision making process (the number of decision taken at each cycle).

The techniques used in each step of the opportunistic cycle follow guidelines widely accepted in

the practice of scheduling applications (e.g., manufacturing scheduling). However, these guidelines have not so far been validated by hard evidence of their effectiveness. Although it would be most desirable to validate these guidelines with respect to a a formal model, at present no model can realistically capture the deep structure of scheduling problems. In its absence, hard evidence of performance can be gathered with empirical studies. This experience is also necessary to guide the search for an appropriate formal model.

Previous empirical studies have analyzed the impact of different granularities of the **Decide** phase [11]. In this paper we consider all the steps of the opportunistic cycle. In particular, we answer the following performance questions:

1. In the Analyze phase, what is the effect of using different constraint relaxation assumptions?
2. In the Focus phase, how important is it to focus problem solving on bottleneck areas?
3. In the Decide phase, what is the best decision making granularity?

The study suggests the following conclusions:

1. It is important to analyze capacity by considering as many currently known constraints as possible;
2. Bottleneck focusing plays a central role in scheduling;
3. To achieve acceptable overall performance, it is important to take as many scheduling decisions as possible on the basis of a single analysis step.

In the rest of the paper we first describe the basic structure of the CPS procedure (section 2). Then we describe the details of each phase of CPS and make hypotheses regarding their effect on the performance of the procedure (sections 3, 4, and 5). In these sections we also identify variations to the steps that, once empirically tested, can confirm or negate the performance hypotheses. Section 6 reports the results of the experimental analysis and section 7 concludes the paper.

2 Conflict Partition Scheduling

Perhaps the main characteristic that sets CPS apart from all other opportunistic schedulers is the search space that it explores. Other schedulers follow a *value commitment* approach, i.e., proceed by assigning precise values to the start and end time of each activity.* Instead, CPS adopts a *constraint posting* approach, i.e., it operates by posting temporal precedences among activities (activity a_i must follow activity a_j). During problem solving each activity has an associated window of opportunity for its execution; this can be deduced by propagation [3] across the activity network, the network activities and precedence constraints. At any point during problem solving, CPS leaves open a greater number of possible start and end time for each activity compared to a value commitment scheduler; this suggests a lower risk of the scheduler "getting lost" in blind alleys. Previous empirical studies have shown that CPS does perform better than other value commitment approaches [9] [10].

The outline of the basic CPS procedure is the following:

---

*Although CPS can deal with activities with flexible durations (i.e., the duration of α must fall in the range $[d_\alpha, D_\alpha]$), to simplify the presentation we will only consider activities with fixed durations

1. **Capacity Analysis**: estimate activity demand and resource contention.
2. **Termination Test**: If the resource contention for each resource is zero over the entire scheduling horizon, then exit. The current activity network is the solution.
3. **Bottleneck Detection**: Identify the resource and time with the highest contention;
4. **Conflict Identification**: Select the activities that are most likely to contribute to the bottleneck contention.
5. **Conflict Arbitration**: Sort the set of conflicting activities according to the activity demand, by inserting appropriate precedence constraints.
6. **Constraint Propagation**: Update the time bounds for the execution of each activity as a consequence of the introduction of the new precedence constraints.
7. **Consistency Test**: If the activity network is inconsistent, signal an inconsistency and exit.
8. Go to 1.

The basic CPS procedure is strictly monotonic. If it generates an inconsistency, CPS backtracks by resetting the activity network to the initial state and restarting the procedure. As we will see later, CPS's capacity analysis is stochastic in nature; therefore, each repetition explores a different path in the problem solving space and can possibly lead to success. If a solution has not been found after a fixed number of repetitions (usually 10), CPS terminates with an overall failure.

In relation to the basic opportunistic cycle described in section 1, step 1 corresponds to **Analyze**, steps 3 and 4 to **Focus**, and step 5 to **Decide**. We will now discuss each step separately and analyze its impact on overall performance.

3 Capacity Analysis

In opportunistic scheduling, the goal of the Analyze phase is to generate an estimate of the structure of the remaining search space without engaging in detailed problem solving. The resulting problem space metrics give an indication of (1) the location of the critical constraints, those that are most likely to endanger the problem solving process, and (2) the level of preference on open alternatives, showing which decisions are most likely to yield successful problem solving.

The problem space metrics gathered by CPS's capacity analysis are statistics on a set of complete value assignments for the start and end times of all the activities in the network. Generating a complete value assignment correspond to making a specific assumption on when each activity will be in execution. A complete value assignment is a consistent schedule only if it satisfies all the precedence and capacity constraints of the problem. An appropriate stochastic simulation process generates each complete value assignment. The process used by CPS differs from other stochastic simulation techniques [4, 5] that estimate the possible outcomes of executing a detailed schedule in an uncertain environment. Having to insure executability, these simulations must introduce additional constraints to complete an intermediate problem solving state into a consistent schedule. Therefore, they end up considering many more details than are useful or necessary for an aggregate capacity analysis. Instead, CPS's stochastic simulation [8] considers only the constraints that are explicit in the current intermediate state, with very weak assumptions on how this is going to be completed into a complete schedule.

The steps of CPS's stochastic simulation are the following. Initially a temporal constraint

propagation [3] establishes the range of possible values for the start time of each activity in the network. At each simulation step i an activity α_i is selected from the network. Then, a value is chosen randomly within the range of possible start times, and is assigned to the start time of α_i. The random choice follows a given probability distribution, or selection rule; possible selection rules include the uniform distribution (i.e., all currently available times have equal weight) and the linearly biased distribution (i.e., the weight of the currently available times increases or decreases linearly over the time bound). The consequences of the start time assignment are propagated through the network to restrict the range of other activities' start times. Then α_i is extracted from the network and the cycle is repeated until all the activities of the network have been assigned a start time. Figure 1 illustrates a step of a stochastic simulation that uses a linear selection rule.

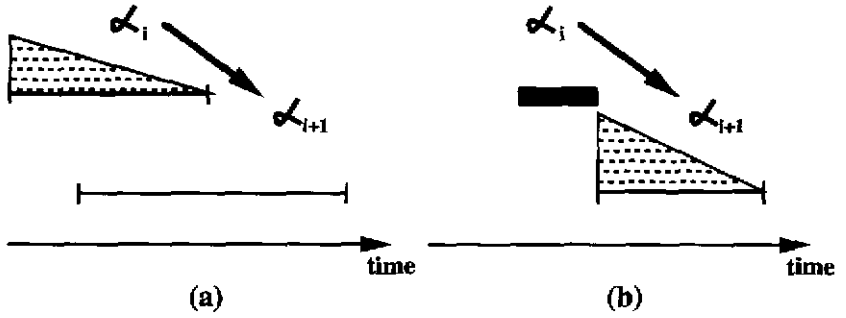

Figure 1: Simulation step: (a) before step i; (b) after step i

In the following $EST(\alpha)$ and $LFT(\alpha)$ will denote, respectively, the earliest start time and the latest finish time of the activity α, H will denote the overall scheduling horizon, and R will be the set of resources.

If the stochastic simulation is repeated N times, we will obtain a sample of N complete value assignments. This sample is then used to estimate the the following two problem space metrics:

- **activity demand**: for each activity α and for each time $EST(\alpha) \le t_i < LFT(\alpha)$, activity demand $\Delta(\alpha, t_i)$ is equal to n_{t_i}/N, where n_{t_i} is the number of complete value assignments in the sample for which the activity is in execution at time t_i.
- **resource contention**: for each resource $\rho \in R$ and for each time $t_j \in H$, resource contention $X(\rho, t_j)$ is equal to n_{t_j}/N, where n_{t_j} is the number of complete value assignments in the sample for which ρ is requested by more than one activity at time t_j.

Activity demand and resource contention represent two different aspects of the current problem solving state. Activity demand is a measure of preference; it indicates how much the current constraints bias an activity toward being executed at a given time. Resource contention is a measure of potential inconsistency; it indicates how likely it is that the current constraints will generate capacity request congestion on a resource at a given time.

In the capacity analysis procedure that we have just described, activity demand and resource contention are computed on the basis of the same N sample. Moreover the stochastic simulation uses all the constraints currently in the network, since at each cycle it propagates the effects of a start time assignment throughout the network. However, we can generalize this capacity analysis by allowing (1) to use different samples to compute different statistics and (2) to generate a sample considering a less constraints than those present in the current problem solving state. In

fact, it is possible to use simplified simulations that drop some or all of the activity precedence constraints during the start time selection process. Micro-opportunistic scheduling [11] follows this approach. Resource contention is based on the assumption that the range of possible start times for each activity has a *subjective probability* of being executed. Each subjective probability is computed independently. If we assume a stochastic simulation semantics, this is equivalent to dropping all the precedence constraints during the iterative start time selection. Conversely, micro-opportunistic scheduling uses an activity demand measure that is based on a complete enumeration of all possible start time assignments consistent with all the constraints in the current activity network. Again assuming a stochastic simulation semantics, this is equivalent to considering all the precedence constraints during the iterative start time selection. Since micro-opportunistic scheduling makes some additional simplification assumption and uses value commitment to make scheduling decisions, resource contention and activity demand can be computed deterministically by a fast, complete enumeration of all the possible value assignments.

In general, the less that constraints are considered in the simulation, the faster capacity analysis is going to be. However, speed has a potential cost. The higher the number of dropped constraints, the higher the percentage of complete value assignments that will potentially violate already imposed precedence constraints. Therefore, when we drop constraints we increase the distance between the space of complete value assignments used by the capacity analysis and the original search space. This makes the problem space metrics less reliable and can negatively affect the scheduling process.

The previous discussion leads to the first question on the performance of an opportunistic scheduling procedure:

- In the Analyze phase, what is the effect of using different constraint relaxation assumptions?

To answer this question, we tested two different configurations of CPS's capacity analysis:

- **complete (COMP)**: this is the original capacity analysis of CPS. Both activity demand and resource contention are evaluated with respect all the constraints in the current activity network.
- **simplified (SIMP)**: only activity demand uses the complete network, while resource contention is computed as in [11], by dropping all precedence constraints in the current activity network. Both simulations use the same type of stochastic start time selection rule.

4 Bottleneck Detection

A widely accepted principle in problem solving is to focus on the most tightly interacting variables at each step, i.e., those with the lowest flexibility on the set of possible values. For example, in constraint satisfaction search [6] the "most constrained first" heuristic minimizes the expected length of any path in the search tree and therefore increases the probability of achieving a solution in less time. In opportunistic scheduling this principle translates into looking for bottleneck resources, i.e., resources that have a high ratio between some measure of aggregate demand and some measure of aggregate capacity. The precise definition of bottleneck given by each scheduling system depends on the specific nature of its problem space metrics.

In the case of CPS, each problem solving cycle focuses on a set of activities that are potentially in conflict, also called the conflict set; more precisely, a conflict set is a set of activities that request the same resource, have overlapping execution time bounds, and are not necessarily sequential, according to the precedence constraints of the current activity network. To detect a

conflict set, CPS first identifies a bottleneck on the basis of the resource contention measures:

- **Bottleneck**: Given the set of resource contention functions $\{X(\rho, t)\}$ with $\rho \in R$ and $t \in H$, we call bottleneck a pair $\langle \rho_b, t_b \rangle$ such that:

  $$X(\rho_b, t_b) = \max \{X(\rho, t)\}$$

  for any $\rho \in R$ and $t \in H$ such that $\{X(\rho, t) > 0\}$.

The conflict set is extracted among the activities with their current time bounds overlapping the bottleneck time. See [9] for a more detailed description of the procedure.

Although focusing on bottlenecks is widely accepted in opportunistic scheduling, there is little quantitative evidence of its effectiveness. As we mentioned in section 1, the performance of an opportunistic scheduler depends on the combination of several techniques and on the use of several kinds of information. For example, activity demand provide a measure of preference on the execution times for each activity. Both in micro-opportunistic scheduling and in CPS (as we will see in section 5) this measure is used to decide how conflicting activities should be sorted. One could wonder if in fact this preference information would not be sufficient. If so, the performance of the scheduler would not change if it focussed on any set of activities, either associated to a bottleneck or not. If this were true, one could save the additional cost required to compute resource contention. In summary, the additional cost of finding bottlenecks is justified only if there is a return in terms of additional problem solving power.

The previous discussion leads to the second question on the performance of an opportunistic scheduling procedure:

- In the Focus phase, how important is it to focus problem solving on bottleneck areas?

To answer this question we considered two different configurations of CPS's Bottleneck Identification step:

- **maximum contention bottleneck (BTL)**: the original method used in CPS. The set of conflicting activities is selected around the bottleneck;
- **random (RAND)**: the resource contention measure is completely disregarded and the conflict set is selected around a randomly chosen resource and time.

5 Conflict Arbitration

At each Conflict Arbitration step, CPS introduces additional precedence constraints between pairs of activities. The new constraints restrict the mutual position of activities and their time bounds. However, unlike in value commitment scheduling time bounds are not completely collapsed.

An important aspect that characterizes different opportunistic schedulers is the granularity of decision making. At one end of the spectrum there are schedulers that make the minimum possible decision at each scheduling cycle; this follows the spirit of micro-opportunistic scheduling [11]. At the other end of the spectrum there are schedulers that follow a macro-opportunistic approach [12] [1]. These schedulers make decisions that completely eliminate any possibility of conflicts among all the activities in the conflict set.

Within CPS it is possible to explore the consequence of different granularities of the scheduling decision phase. For example, a micro-opportunistic approach translates into adding a single precedence constraint at each cycle. The corresponding pair of activities is extracted from the conflict set. Conversely, a macro-opportunistic approach could be implemented by totally

ordering all activities in the conflict set.

The initial implementation of CPS [9] proposes an approach with intermediate granularity. It consists of partitioning the conflict set into two subsets, A_{before} and A_{after}, and then constraining every activity in A_{before} to occur before any activity in A_{after}. The bi-partition of the original conflict set relies on a clustering analysis of the activity demands (Figure 2). For details see [9].

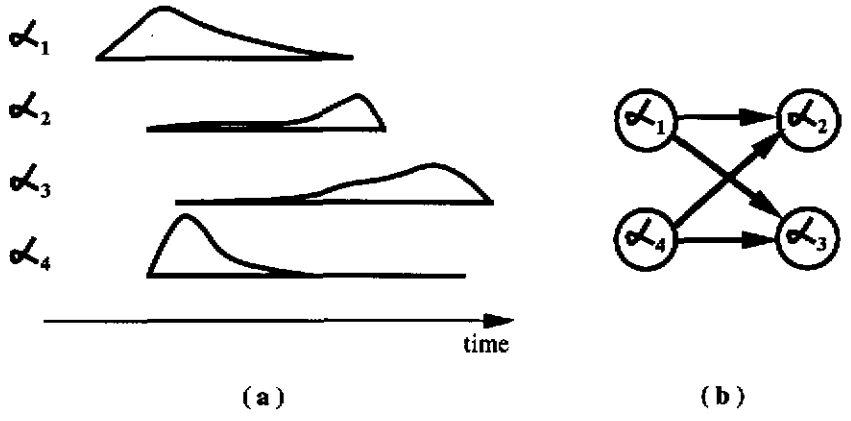

Figure 2: A Conflict Arbitration step

The choice of the appropriate decision granularity requires to evaluate a trade-off. The more scheduling decisions are made in a step, the quicker the search space will be pruned. A larger granularity will therefore increase the speed with which the scheduler reaches either a solution or a failure point. However, the more decisions are made, the greater the change in the topology of the activity network after the step. The resulting network could in fact be significantly different than the starting one. Therefore, an analysis done before the step could give little information on the structure of the destination state. This can increase the likelyhood of failure and consequent backtracking, and therefore slow down the scheduler. In summary, the important trade-off involves on one hand the speed of convergence and on the other the number of backtracking steps needed.

The previous discussion leads to the third question on the performance of an opportunistic scheduling procedure:

- In the Decide phase, what is the best decision making granularity?

To answer this question we considered two distinct Conflict Arbitration rules:

- **conflict bi-partition (BIP)**: the technique originally used in CPS. The conflict set is separated into two A_{before} and A_{after} subsets.
- **most separated activities (MS)**: a *micro* arbitration technique. It introduces a single precedence between two activities extracted from the conflict set. To minimize the impact of sampling noise, we select the two activities whose demand profiles are maximally separated.

6 Experimental Results

The experimental analysis was conducted on the Constraint Satisfaction Scheduling benchmark proposed in [11]. The benchmark consists of 6 groups of 10 problems, each with 50 activities, 5 resources, and non-relaxable release and due date constraints. Each set is identified by the two following parameters that describe its expected difficulty: (1) the spread of the release and due

dates, which can assume the three levels (in order of increasing expected difficulty) wide (w), narrow (n) and null (0); (2) the number of expected bottleneck resources, either 1 or 2. For more details see [11].

Tables 1 and 2 report the performance of the configurations of CPS obtained by combining the alternatives described in the previous sections in every possible way. Each column is marked with three labels identifying each of the steps in CPS. Table 1 shows the average number of problems solved over 5 independent runs of the procedure; table 2 reports the corresponding processing times. The processing times are the average of the weighted number of opportunistic cycles needed to process a problem in the subset (either finding a solution or failing). The number of cycles is weighted according to the different complexities of complete (*COMP*) and simplified (*SIMP*) stochastic simulation. More precisely, each *SIMP* cycle is assumed to take 85.64% of the time needed by a single *COMP* cycle. The speed-up factor was evaluated on the current CPS implementation and takes into account known implementation inefficiencies. In fact, the speed-up factor is optimistic since we considered the time needed to compute the *SIMP* resource contention as equal to zero.

	<COMP, BTL, BIP>	<COMP, BTL, MS>	<SIMP, BTL, BIP>	<SIMP, BTL, MS>	<SIMP, RAND, BIP>	<SIMP, RAND, MS>
w/1	10	10	10	10	10	9.4
w/2	10	10	10	10	10	9.2
n/1	10	10	10	10	9.2	8.6
n/2	10	10	10	10	8.2	5.8
0/1	10	10	9.8	10	8.4	9
0/2	8.8	10	9	8.4	4.8	4
TOT	58.8	60	58.8	58.4	50.6	46

Table 1: Experimental results: number of problem solved

	<COMP, BTL, BIP>	<COMP, BTL, MS>	<SIMP, BTL, BIP>	<SIMP, BTL, MS>	<SIMP, RAND, BIP>	<SIMP, RAND, MS>
w/1	45.12	121.78	56.16	107.51	59.21	280.14
w/2	45.42	137.62	56.42	139.18	79.87	487.91
n/1	46.38	129.74	57.91	111.98	114.71	403.00
n/2	46.58	134.96	59.18	158.31	139.23	815.89
0/1	48.16	147.30	57.46	118.58	112.10	380.99
0/2	64.10	153.48	79.73	270.74	213.29	927.21
AVG	49.29	137.48	61.14	151.05	151.05	549.19

Table 2: Experimental results: processing time

Both demand and contention measures were computed from samples of $N = 10$ complete value assignments. The stochastic simulation had the following configuration. At each simulation cycle the selected activity was one among those with no predecessor activities with unassigned

start times (forward temporal dispatching). The start time value was selected according to a linearly biased probabilistic selection rule. The rule gave the highest preference to the earliest time and the lowest (0) preference to the latest time. See [9] for details.

We will now answer each of the three performance questions on the basis of the results of the empirical analysis.

6.1 Analysis: Effects of different constraint relaxation assumptions

Let us compare each entry in a **<COMP, BTL, ?x>** with the corresponding entry in the **<SIMP, BTL, ?x>** column. We can see a slight advantage for **COMP** with respect to **SIMP**(overall average of 59.4 problems solved against 58.6). The advantage is more definite when comparing processing times; on average, a **SIMP** configuration is 16.19% slower than the corresponding **COMP** configuration. The results show a significant overall loss of performance when constraints are dropped during capacity analysis. Therefore, the experimental results indicate that it is important to consider as many constraints as possible during the Analyze phase.

6.2 Focus: Effects of bottleneck focusing

Any **<SIMP, RAND, ?x>** entry is substantially worse than the corresponding entry in **<COMP, BTL, ?x>** and **<SIMP, BTL, ?x>**, both in terms of number of problem solved (11.1 less problems than **COMP** and 10.3 less than **SIMP**) and processing time (average slowdown of 215.11% against **COMP** and 174.42% against **SIMP**). All things remaining equal, there is a substantial advantage in concentrating on the points that CPS characterizes as bottlenecks. Although, beyond the qualitative description, we lack a deep understanding of what a bottleneck point really represents, it is apparent that bottlenecks are interesting focusing points and the capacity analysis is able to find them. This result is independent from the number of constraints dropped during the Analyze step. Therefore, the experimental results indicate that it is important to concentrate problem solving effort on bottleneck areas.

6.3 Decide: Effects of different decision granularities

The number of problems solved increase slightly when going from **<COMP, BTL, BIP>** to **<COMP, BTL, MS>**. However, this advantage is mostly due to a single problem that **MS** can solve while **BIP** cannot. If we compare the processing times for the two previous configurations, we see an average slowdown of 181.27% going from **BIP** to **MS**. Similar slowdowns can be detected when comparing the other configurations. This result shows that the costs of low (micro) scheduling decision granularities is prohibitive in most cases. The adequacy of macro-granularity decisions might depend on the expected complexity of the problem, falling short as problem complexity increases. However, in these cases resorting to micro decisions could be only one of the possible alternatives. A significantly cheaper approach consists of increasing the accuracy of the capacity analysis by increasing the size of the complete value assignment sample. For example, in another study [10] **<COMP, BTL, BIP>** solved consistently all 60 problems using a sample of size N=20 instead of N=10. Therefore, the experimental results indicate that it is important to make as many decisions as possible on the basis of the capacity analysis information.

The comparison between **<SIMP, RAND, BIP>** and **<SIMP, RAND, MS>** yields a surprising result. It is reasonable to expect that a micro decision step would always be more accurate than a comparable macro decision step. In fact, a macro decision step can always be seen as a sequence of micro steps. If additional analyze steps were allowed before each of the component micro steps, some of them might turn out to be unnecessary or even damaging. However, going from **<SIMP, RAND, BIP>** to **<SIMP, RAND, MS>** the number of problems solved does not increase or remain stable, as one would expect; instead, there is a significant loss of performance

(4.6 less solutions are found). Following the micro approach in this case is in fact damaging. We could explain this surprising result if we hypothesize that the utility of the preferences information used during Conflict Arbitration (i.e., activity demand profiles) is a function of the selected conflict set. For some conflict sets the preference information might in fact be misleading, guiding the scheduler toward a failure. Since the **RAND** focusing method is totally uninformed, it might select a misleading conflict set during any opportunistic scheduling cycle. We can assume that **RAND** has a given probability of selecting a misleading conflict set. The overall probability of failure, then, increases with the expected number of opportunistic cycles. Such probability is the sum of the probabilities of failing after x cycles, with x less or equal to the expected length of a problem solving trace. This substantial degradation is not present in the bottleneck centered configurations (**BTL**). This could be an indication that detecting a bottleneck indeed excludes the selection of misleading conflict sets. The previous discussion is another strong indication of the strength of bottleneck-centered scheduling. Bottlenecks are indeed a strong characterization of the overall structure of a scheduling problem.

7 Conclusions

In this paper we have reported an experimental analysis of different configurations of the CPS procedure. The aim of the analysis is to go beyond a bulk comparison of systems and to identify important design trade-offs among system components. Understanding these trade-offs is crucial for the design of more effective scheduling systems. The trade-offs that we identified are significant for every opportunistic scheduler. The CPS's components tested in this study have a direct correspondent in other opportunistic schedulers. Therefore, we believe that our results have validity for the entire class of opportunistic scheduling systems. Our experimental analysis answers three fundamental performance questions on the Analyze-Focus-Decide opportunistic cycle. In the Analyze phase it is important to consider as many constraints as possible among those contained in an intermediate problem solving state. In the Focus phase, it is extremely important to concentrate on bottlenecks. In the Decide phase, if we want acceptable overall performance, we need to make as many scheduling decisions as possible on the basis of a single analysis step.

References

[1] Adams, J., Balas, E., Zawack, D.
The Shifting Bottleneck Procedure for Job Shop Scheduling.
Management Science 34, 1988.

[2] Biefeld, E., Cooper, L.
Bottleneck Identification Using Process Chronologies.
In *Proceedings of the 12th International Joint Conference on Artificial Intelligence.* 1991.

[3] Dechter, R. and Meiri, I and Pearl, J.
Temporal Constraint Networks.
Artificial Intelligence 49:61-95, May, 1991.

[4] Drummond, M. and Bresina., J.
Anytime Synthetic Projection: Maximizing the Probability of Goal Satisfaction.
In *Proceedings of the 8th National Conference on Artificial Intelligence*, pages 138-144. AAAI Press, 1990.

[5] Hanks, S.
Practical Temporal Projection.
In *Proceedings of the 8th National Conference on Artificial Intelligence*, pages 158-163. AAAI Press, 1990.

[6] Haralick, R.M. and Elliot, G.L.
Increasing Tree Search Efficiency for Constraint Satisfaction Problems.
Artificial Intelligence 14(3):263-313, October, 1980.

[7] Minton, S., Johnston, M. D., Philips, A. B., Laird, P.
Solving Large-Scale Constraint Satisfaction and Scheduling Problems Using a Heuristic Repair Method.
In *Proceedings of the 8th National Conference on Artificial Intelligence.* 1990.

[8] Muscettola, N., S.F. Smith.
A Probabilistic Framework for Resource-Constrained Multi-Agent Planning.
In *Proceedings of the 10th International Joint Conference on Artificial Intelligence*, pages 1063-1066. Morgan Kaufmann, 1987.

[9] Muscettola, N.
Scheduling by Iterative Partition of Bottleneck Conflicts.
Technical Report CMU-RI-TR-92-05, The Robotics Institute, Carnegie Mellon University, February, 1992.

[10] Muscettola, N.
Scheduling by Iterative Partition of Bottleneck Conflicts.
In *Proceedings of the 9th Conference on Artificial Intelligence for Applications.* IEEE Computer Society Press, March, 1993.

[11] Sadeh, N.
Look-Ahead Techniques for Micro-Opportunistic Job Shop Scheduling.
Technical Report CMU-CS-91-102, School of Computer Science, Carnegie Mellon University, 1991.

[12] Smith, S.F., Ow, P.S., Potvin, J.Y., Muscettola, N., and Matthys, D.
An Integrated Framework for Generating and Revising Factory Schedules.
Journal of the Operational Research Society 41(6):539-552, 1990.