An Iterative Sampling Procedure for Resource Constrained Project Scheduling with Time Windows

Amedeo Cesta
IP-CNR
National Research Council
Viale Marx 15
I-00137 Rome, Italy
amedeo@pscs2.irmkant.rm.cnr.it

Angelo Oddi
IP-CNR
National Research Council
Viale Marx 15
I-00137 Rome, Italy
oddi@pscs2.irmkant.rm.cnr.it

Stephen F. Smith
The Robotics Institute
Carnegie Mellon University
5000 Forbes Avenue
Pittsburgh, PA 15213, USA
sfs@isl1.ri.cmu.edu

Abstract

In this paper, we extend and integrate previously reported techniques for resource constrained scheduling to develop a CSP procedure for solving RCPSP/max, the resource constrained project scheduling problem with time windows (generalized precedence relations between start time of activities). RCPSP/max is a well-studied problem within the Operations Research community and the presence of a large set of benchmark problems provides a good opportunity for comparative performance analysis. Our base CSP scheduling model generalizes previous profile-based approaches to cumulative scheduling by focusing on global analysis of minimal conflicting sets rather than pairwise conflict analysis. This generalization increases the tendency for more effective conflict resolution. Since RCPSP/max is an optimization problem, other ideas from prior work are adapted to embed this base CSP model within a multi-pass, iterative sampling procedure. The overall procedure, called ISES (Iterative Sampling Earliest Solutions), is applied to the above mentioned set of benchmark problems. ISES is shown to perform quite well in comparison to current state-of-the-art procedures for RCPSP/max, particularly as search space size becomes limiting for systematic procedures.

1 Introduction

In recent years, CSP scheduling research has been increasingly concerned with development of models for solving cumulative (i.e., multi-capacity resource) scheduling problems. Such models are important because they are much better matched to the requirements of many practical scheduling environments than traditional CSP scheduling models. Much of the work in this area has focused on constraint propagation techniques that exploit the structure of cumulative resource constraints, and are hence capable of stronger inference. Less attention has been paid to development of effective heuristics for managing the scheduling search process.

In this paper, we focus on this latter issue. Drawing from previous work in resource-constrained scheduling, we develop a base CSP resolution procedure for cumulative scheduling. Our procedure proceeds by iteratively detecting and leveling "resource contention peaks", i.e., periods where demand is projected to exceed resource capacity. The conflict selection and resolution heuristics used in our procedure to direct the search generalize those employed in previous "profile-based" scheduling approaches, replacing localized, pair-wise analysis of competing resource requests with more global analysis of minimal conflict sets. Such extended analysis can be expected to lead to more informed resolution decisions.

Our specific interest in this paper is application of this scheduling approach to the resource constrained project scheduling problem with time windows (RCPSP/max), a well-studied and difficult makespan minimization problem. To this end, extensions are developed to enable use of the base resolution procedure within a larger "optimizing" search process. Below, we first define the RCPSP/max problem (Section 2). Next, we describe elements of our composite RCPSP/max solution procedure (Sections 3 and 4). Finally we report comparative performance results on a previously studied set of benchmark problems (Section 5).

2 The RCPSP/max Problem

The RCPSP/max scheduling problem can be formalized as follows:

- a set V of n activities to be executed, where each activity j has a fixed duration D_j. Each activity has a start-time S_j and a completion-time C_j that satisfies the constraint $S_j + D_j = C_j$.
- a set E of temporal constraints between activity pairs $< i, j >$ of the form $S_j - S_i \in [T_{ij}^{min}, T_{ij}^{max}]$, called start-to-start constraints (time lags or *generalized precedence relations* between activities). [1]
- a set R of renewable resources, where each resource r_k has a integer capacity $c_k \geq 1$.

[1]Note that since activity durations are constant values, end-to-end, end-to-start, and start-to-end constraints between activities can all be represented in start-to-start form.

- execution of an activity j requires one or more resources. For each resource r_k the integer $rc_{j,k}$ represents the required capacity (or *size*) of activity j.

A schedule S is an assignment of values to the start-times of all activities in V $(S = (S_1, \ldots, S_n))$. A schedule is *time-feasible* if all temporal constraints are satisfied (all constraints $S_j - S_i \in [T_{ij}^{min}, T_{ij}^{max}]$ and $S_j + D_j = C_j$ hold). A schedule is *resource-feasible* if all resource constraints are satisfied (let $A(S,t) = \{i \in V | S_i \leq t < S_i + D_i\}$ be the set of activities which are in progress at time t and $r_k(S,t) = \sum_{j \in A(S,t)} rc_{j,k}$ the usage of resource r_k at that same time; for each t the constraint $r_k(S,t) \leq c_k$ must hold). A schedule is *feasible* if satisfies both type of constraints. The RCPSP/max optimization problem, then, is to find a feasible schedule with *minimum makespan* MK (where $MK(S) = max\{C_i\}$).

In [Bartusch *et al.*, 1988], it is shown that finding a feasible schedule alone for RCPSP/max is NP-hard. This difficulty is due to the presence of maximum separations T_{ij}^{max}. The RCPSP/max problem has been the subject of recent investigation within the Operations Research (OR) Community, yielding lower-bound analysis [Heilmann and Schwindt, 1997] and a number of branch and bound (B&B) solution procedures (e.g., [De Reyck and Herroelen, 1998; Möhring *et al.*, 1998; Schwindt, 1998; Dorndorf *et al.*, 1998]). A problem generator PROGEN/max [Schwindt, 1996] has been made available on the Web together with a set of reference problems [Kolisch *et al.*, 1998].

3 Constructing a Feasible Solution

We begin by specifying a core CSP resolution procedure for generating feasible solutions to instances of RCPSP/max. Our approach follows the same general schema as other constraint-posting scheduling approaches (e.g., [Cheng and Smith, 1994; Cesta *et al.*, 1998a]). An initial, time-feasible solution is first computed (ignoring all resource constraints). Resource constraints are then super-imposed and an iterative search is performed to resolve all resulting resource *conflicts* (i.e., sets of activities competing for the same resource capacity over some time interval). On each iteration a particular conflict is selected to resolve, and one additional precedence relation is posted to eliminate the contention. The search continues until either a feasible solution is found (i.e., all conflicts have been eliminated), or until an unresolvable conflict has been discovered (in which case the search fails).

Clearly, one key to the effectiveness of this greedy search procedure will be the strength of the heuristics used to select and resolve pending conflicts. In this regard, our approach integrates ideas from a previously reported clique-based approach to resource reasoning [Laborie and Ghallab, 1995] to improve on previously developed heuristics for cumulative scheduling. A second way in which the effectiveness of above (partial) procedure can be enhanced is through the addition of backtracking or restarting mechanisms, which serve to broaden the search in the event of failure. Exploiting the approach taken in [Oddi and Smith, 1997], we develop a randomized variant of our conflict selection heuristic and embed the core resolution procedure within a larger, iterative sampling search. Before elaborating on the design of these aspects of our feasible solution generator, we first briefly summarize the constraint propagation machinery used to support the search.

3.1 Constraint Propagation

A CSP-based resolution procedure integrates two basic components: a *search procedure* responsible for refining the current CSP by taking decisions on possible values for variables, and a *constraint propagation procedure* that, after each choice of the search, computes implications by updating the status of the CSP and eliminating inconsistent values. In a scheduling problem like RCPSP/max both temporal and resource constraints can be sources of propagation:

Time constraints. Our search procedure manipulates a temporally consistent network of time points, constrained initially by the set of temporal constraints identified in Section 2 and further constrained as additional precedence constraints are posted during the search. Path consistency in this network is dynamically maintained *via* all pair shortest path computation, making distance information between any pair of time variables available for use in focusing the search.

Resource constraints. We do not make use of resource constraint propagation as is done in some other recent approaches to disjunctive and cumulative scheduling problems (e.g., [Nuijten and Aarts, 1996; Nuijten and Le Pape, 1998]). This form of deduction can be seen as complementary to our approach; it could be added in a transparent way to further prune the search space (and potentially improve reported results). This is one direction of current research.

3.2 ESA: A Basic Resolution Algorithm

Figure 1 specifies the basic "conflict removal" procedure used to generate a feasible solution, referred to as ESA (Earliest Start Algorithm). The algorithm accepts a problem instance (`Problem`) and an upper bound on the overall makespan (`Horizon`). It first computes an earliest start time solution that assumes "infinite capacity" ($\texttt{Problem}_{EST}$ at line 1) and then attempts to incrementally transform this initial time-feasible solution into a resource-feasible solution. At each step, a resource conflict still present in the current solution is selected and resolved, by posting a precedence constraint that delays the earliest start time of one of the competing activities.

Contention Peaks on Resource Profiles. Given a time-feasible, earliest start time schedule (`ESTsol`), it is straightforward to identify time instants t where the resource capacity constraint of a given resource is violated. We say that there is a *contention peak* on resource r_k at time t if condition $r_k(\texttt{ESTsol}, t) > c_k$ holds. Intuitively, a contention peak on resource r_k characterizes a conflict,

```
ESA(Problem, Horizon)
1.  ESTsol ← Problem_EST
2   SetHorizon(ESTsol, Horizon)
3.  loop
4.    if ResourceFeasible(ESTsol)
5.    then return(ESTsol)
6.    else
7.      if ExistUnsolvableConflict(ESTsol)
8.      then return(EmptySolution)
9.      else
10.       Conflict ← SelectConflict(ESTsol)
11.       PrecC ← SelectPrecedence(Conflict)
12.       Post&Update(PrecC,ESTsol)
13. end-loop
14. end
```

Figure 1: The One-Pass Greedy Algorithm

identifying a set of activities that simultaneously require r_k but have a combined capacity requirement $> c_k$.

Obviously feasible solutions do not contain peaks, and one possible way to transform `ESTsol` into a feasible solution is to detect and remove all peaks. This is the basic idea behind *profile-based* scheduling procedures [Cesta *et al.*, 1998a]. A peak can be removed by "leveling" it, that is by posting one or more precedence constraints between pairs of activities contributing to the peak (such constraints are posted between the end-time of one activity and the start-time of the other to avoid overlap). In leveling a conflict, the choice of which pair(s) of activities to order can be quite crucial to the quality of the solution, as it is possible to add precedence relations that do not have a strong peak leveling effect. For example, assume resource r_k with capacity $c_k = 4$ has a peak that includes the following activities (capacity requirements in brackets): $\{a_1[3], a_2[2], a_3[2], a_4[1]\}$. Two pairs of activities that might be selected for ordering are $< a_2, a_4 >$ or $< a_2, a_1 >$. In the case of $< a_2, a_1 >$ the combined requirement exceeds c_k and hence *any feasible solution* must have a_2 and a_1 reciprocally ordered. Alternatively, an ordering of the pair $< a_2, a_1 >$ may or may not be essential to leveling the conflict. One observed shortcoming of previous profile-based approaches to cumulative scheduling has been their tendency to post unnecessary ordering constraints, following from the use of conflict selection and resolution heuristics that rely strictly on pairwise analysis of competing activities [Cesta *et al.*, 1998a].

Conflicts are Minimal Critical Sets of Activities. An alternative approach to conflict analysis that can overcome this problem has been proposed in [Laborie and Ghallab, 1995]. Under this scheme, an activity "intersection graph" is constructed and systematically searched for particular cliques. Such a clique is called a *Minimal Critical Set* (MCS). It specifies *a set of activities that simultaneously require a resource r_k with a combined capacity requirement $> c_k$, such that the combined requirement of any subset is $\leq c_k$*. The important advantage of isolating MCSs is that a single precedence relation between any pair of activities in the MCS eliminates the resource conflict.

Unfortunately, the exponential nature of the intersection graph search prohibits use of this basic approach on scheduling problems of any interesting size. In [Cesta *et al.*, 1998b], it is shown that much of the advantage of this type of global conflict analysis can be retained by using an approximate procedure for computing MCSs. But the pragmatic cost of recomputing MCSs across all resources at each iteration of the CSP resolution procedure nonetheless remains high and significantly limits scalability.

In ESA, we achieve an even better computational tradeoff by instead integrating the use of MCS analysis into a profile-based scheduling framework. On each iteration of the search, we first compute contention peaks (which is quadratic in the number of activities) to isolate those areas of the solution where conflicts (i.e., MCSs) should be computed. Next we generate a set of MCSs for each peak. (These two steps are embedded respectively in the `ResourceFeasible(ESTsol)` predicate and `ExistUnsolvableConflict` function of lines 4 and 7 in the algorithm description of Figure 1.) The number of MCSs contained in a given peak can still be quite large (in the worst case $\binom{|V|}{c_k+1}$), and, accordingly, ESA also utilizes an approximate (heuristic) scheme for computing MCSs.

Computation of Critical Minimal Sets. Two heuristic sampling schemes for computing the set of MCSs associated with a given contention peak P are evaluated in this paper: *linear sampling* and *quadratic sampling*. In both cases, the activities $j \in P$ are first sorted in order of decreasing size of their capacity requirements (i.e. largest first), to promote detection of the "most critical" MCSs. The two schemes are summarized as follows:

Linear sampling. Under this scheme, a queue Q is used to select an MCS in P. Activities $j \in P$ are sequentially considered (in sorted order) and inserted in Q until the sum of the resource requirements exceed the resource capacity. At this point, the set Q (the current MCS) is collected in a list of MCSs and the first element from Q is removed. The previous steps are iterated and MCSs are collected until there are no uninserted activities in P.

Quadratic sampling. This scheme can be seen as an extension of linear sampling in which the second step is expanded as follows. Once the current MCS has been collected, instead of immediately removing the first element from Q, a forward search through the remaining uninserted activities in P is first performed to also collect all MCSs that can obtained by dropping the last item placed in Q and substituting with single subsequent activities in P until an MCS is still composed.

Consider again the peak from the previous example $\{a_1[3], a_2[2], a_3[2], a_4[1]\}$ and resource r_k with $c_k = 4$. Although the peak contains three MCSs linear sampling collects only two: $\{a_1[3], a_2[2]\}$ and $\{a_2[2], a_3[2], a_4[1]\}$. Quadratic sampling, on the other hand, would also find the third: $\{a_1[3], a_3[2]\}$.

Conflict Removal. Lines 10-12 of the ESA algorithm in Figure 1 remove an MCS from the current solution. The basic steps involve selecting an MCS from the current conflict set (Line 10), selecting a precedence constraint that resolves the selected MCS (Line 11), and posting the constraint in `ESTsol` to perform temporal constraint propagation (Line 12).

The heuristics that govern both conflict selection and conflict resolution in ESA are based directly on least commitment concepts previously utilized for other problems with maximum time lags (e.g., [Cheng and Smith, 1994; Laborie and Ghallab, 1995]). Candidate MCSs are ordered according to the temporal flexibility they contain (a function of the degree to which constituent activities can be reciprocally shifted in time). The less flexibility a MCS has, the more critical it is to resolve first. The greater the flexibility that is retained after posting a precedence constraint that resolves a MCS, the more desirable it is to post that constraint.

To quantify the notion of temporal flexibility, the heuristic estimator K suggested in [Laborie and Ghallab, 1995] is used. Given a candidate MCS and a set $\{pc_1 \ldots pc_k\}$ of precedence constraints that could be posted between pairs of activities in the MCS, $K(\text{MCS})$ is defined as follows:

$$\frac{1}{K(\text{MCS})} = \sum_{i=1}^{k} \frac{1}{1 + commit(pc_i) - commit(pc_{min})}$$

where $commit(pc_i)$ ranges from 0 to 1 and estimates the loss in temporal flexibility as a result of posting constraint pc_i, and pc_{min} is the precedence constraint with the minimum value of $commit(pc)$.

Note that $K(\text{MCS})$ takes on its highest value of 1 in those cases where only one specific precedence constraint can be feasibly posted to resolve the conflict. In general, the closer an MCS is to being unresolvable, the higher the value of $K(\text{MCS})$. The conflict selection heuristic (`SelectConflict`) chooses the MCS with the highest K value, and the conflict resolution heuristic (`SelectPrecedence`) simply chooses pc_{min}.

3.3 Randomizing ESA

The ESA resolution procedure, as defined above, is a deterministic (partial) solution procedure with no recourse in the event that an unresolvable conflict is encountered. To provide a capability for expanding the search in such cases without incurring the combinatorial overhead of a conventional backtracking search, we define a random counterpart of our conflict selection heuristic (in the style of [Oddi and Smith, 1997]) and embed the resulting "Random-ESA" procedure within an iterative sampling search framework. Specifically, the heuristic employed by `SelectConflict` is modified as follows. We define an acceptance band β and consider as equivalent the set of MCSs for which $K_{max}(1-\beta) \leq K(\text{MCS}) \leq K_{max}$. The conflict to be resolved next is randomly selected from this set, resulting in a non-deterministic yet heuristically-biased choice.

Figure 2 depicts the larger iterative sampling algorithm. It is designed simply to invoke the random-ESA resolution procedure a fixed number (`MaxRestart`) of times, rather than predicating any restarts on a failure to produce a feasible solution. Given that the broader objective in this paper is makespan minimization, each restart provides a new opportunity to produce a different feasible solution with lower makespan.

```
mresESA(Problem, MaxRestart, Horizon)
1.  $S_{best}$ ← EmptySolution
2.  Sol ← EmptySolution
    /* MK(EmptySolution)=+∞ */
3.  Counter ← 1
4.  While Counter ≤ MaxRestart
          and MK(Sol) > $mk_0$ do
5.     Sol ← ESA(Problem, Horizon)
6.     if MK(Sol) < MK($S_{best}$)
7.       then $S_{best}$ ← Sol
8.     Counter ← Counter + 1
9.  end-while
10. return($S_{best}$)
11. end
```

Figure 2: The Restarting ESA Procedure

4 An Optimization Procedure

Though the restarting procedure just described does in fact retain the smallest makespan solution generated across calls to **mres**ESA, its principal role is to produce a feasible solution relative to a given upper-bound horizon. In this section, we define an RCPSP/max optimization procedure based on use of this feasible solution generator, and relate it to previous OR approaches to RCPSP/max.

4.1 The ISES Algorithm

Similar to other CSP procedures for makespan minimization (e.g., [Cheng and Smith, 1997]), we adopt a multi-pass approach; the feasible solution generator is repeatedly applied to solve problems with increasingly smaller temporal horizons, until it is no longer possible to find a feasible solution or until a lower-bound solution is found.

Figure 3 shows the specific multi-pass version of the base ESA procedure we have defined, called ISES (Iterative Sampling Earliest Solutions). ISES is composed of two basic steps. First, a feasible solution is found by invoking **mres**ESA with a horizon value (`MaxH`) much greater than the lower bound mk_0. (mk_0 is the initial "infinite capacity" solution to `Problem`). Successive calls are then made to **mres**ESA each time substituting the new best makespan found on the previous call as the new problem horizon. The iteration stops when either (1) a call to **mres**ESA returns an empty solution, (2) a lower bound solution is obtained, or (3) a solution is returned which does not improve the previous best.

4.2 Comparison to OR Approaches

As previously noted, the work of [Bartusch *et al.*, 1988] investigates the mathematical properties of the problem and similarly characterizes the solution space using the temporal constraints. The notion of *forbidden sets* is introduced to represent resource conflicts, equivalent to

```
ISES(Problem, MaxRestart, MaxH)
1. $S_{best}$ ← EmptySolution
   /* find a first feasible solution */
2. Sol ← mresESA(Problem, MaxRestart, MaxH)
3. if Sol ≠ EmptySolution then
   /* improve the current makespan */
4.   repeat
5.     $S_{best}$ ← Sol
6.     Sol ← mresESA(Problem, MaxRestart, MK($S_{best}$))
7.   until $mk_0$ < MK(Sol) < MK($S_{best}$)
8. return($S_{best}$)
9. end
```

Figure 3: The ISES Optimization Algorithm

our definition of contention peaks. The paper also defines a concept similar to MCS (called *reduced forbidden set*). This is used to sketch a systematic B&B procedure which similarly starts from an infinite capacity solution and extends the set of precedence relations in the problem. Unfortunately the computational analysis is quite limited. Only small instances are mentioned and this approach has not influenced later OR computational approaches to RCPSP/max. ISES can be seen as an approximation algorithm based on the same ideas (even if it has evolved from other origins).

More recent B&B approaches have retained the idea of extending a time-feasible solution by adding precedence relations, but analyze the current solution differently. In [De Reyck and Herroelen, 1998] and [Schwindt, 1998], peaks are considered in increasing chronological time order and, for each of them, a set of "minimal delaying activities" are detected for resolving conflicts. Alternatively, ESA does not depend on the chronological time order, but opportunistically acts on the more constrained part of the solution. Other B&B algorithms work differently. For example [Möhring *et al.*, 1998] solves resource conflicts by increasing release dates (minimum time lags relative to beginning of the project) for certain activities instead of introducing precedence relations. A very recent proposal [Dorndorf *et al.*, 1998] obtained strong results using a combination of techniques: it couples a set of temporal and resource constraint propagation rules with a binary branching schema that exploits particular properties of the current partial solution to successively fix and/or delay the start-times of various activities.

Approximate approaches are also defined in the OR literature. One heuristic analysis [Franck and Neumann, 1998] very recently obtained very good results on the RCPSP/max benchmark problems. Despite being classified as a "priority rules" approach, this work uses a two step method: (a) a sophisticated decomposition analysis is performed to identify "critical sub-components" which can be scheduled independently, and (b) the scheduled sub-components (partial schedules) are integrated into one using a set of priority rules. Note that the first step uses properties due to the presence of maximum time lags. With respect to ISES this approach is deterministic and relies on a quite different analysis of the problem.

5 Experimental Evaluation

Our experimental evaluation of ISES focuses on two sets of reference problems taken from the RCPSP/max problem repository [2]:

Problem set A. This is the benchmark problem set described in [Kolisch *et al.*, 1998]. It consists of three sets of 270 problems each, named $J10$, $J20$ and $J30$, with problems of 10, 20 and 30 activities respectively and 5 resources. The only known results for this problem set to date have been obtained using the B&B procedure of [Schwindt, 1998]

Problem set B. This is the benchmark problem set introduced in [Schwindt, 1998] and used by the recent B&B approaches mentioned in Section 4.2. It consists of 1080 problems with 100 activities and 5 resources, of which 1059 are feasible and the rest are provably infeasible. For this set, lower bounds are known for each problem [Heilmann and Schwindt, 1997], providing a common reference point for measuring deviation from optimal solutions.

Both above problem sets were generated by PROGEN/max, a flexible random networks generator [Schwindt, 1996] that allows generation of project scheduling problems of varying structure, constrainedness and difficulty. However, due to differences in the generation parameters used in each case, there are important differences in the characteristics of the problems in each set. The parameter settings used to generate Problem Set A are closer than Problem Set B's to the settings which produce the "hardest possible" problems [Franck and Neumann, 1998]. In particular, the problems in Set A exhibit higher levels of resource contention (i.e., higher contention peaks) than those in Set B, along with increased parallelism in project activities (i.e., increased sequencing flexibiliy). Given these properties, we can expect deeper search trees (i.e., a larger search space) in solving problems from Set A than for problems in Set B, since a greater number of ordering decisions are needed to build up a feasible solution. Hence, even though Problem Set B contains larger (100 activity) problems, the problems in Problem Set A are actually very challenging.

5.1 Experimental Design.

Results are obtained with ISES on Problem Sets A and B using both linear and quadratic MCS sampling schemes. For all experiments, the randomization factor β in MCS selection is set to 0.1, so all MCSs within the 10% of the maximum ranked are considered as equivalent. The number of restarts of the resolution procedure at a given horizon (`MaxRestart`) is set to 10 and 30 for problem set A, and to 10 for problem set B. The maximum horizon `MaxH` is set to 5× mk_0.

All algorithms compared to ISES below have been implemented in C++ and run on a Pentium 200 with an imposed time limit of 100 seconds per problem. Our current

[2]Available at `ftp://ftp.wior.uni-karlsruhe.de/pub/ProGen-max/pspmaxlib/`

implementation of ISES is in Allegro Common Lisp and the reported results are obtained on a SUN UltraSparc 30 (266MHz). The C++ and Lisp implementations are not directly comparable, but we have nonetheless imposed the same 100 second time limit. For problem set B we also compute the performance of ISES for 300 and 500 second limits to better understand the behavior of the algorithm.

Given the non-deterministic nature of ISES each problem is solved multiple times using different random seeds (similar to [Nuijten and Le Pape, 1998] which also evaluates a random restart algorithm). We report two results: (1) the best result ($ISES_{best(n)}$), obtained using the best result on any single problem over the n different runs, and (2) the average result ($ISES_{avg(n)}$), obtained using the average result on each problem over its n runs.

5.2 Problem Set A

Table 1 gives overall performance results on the J10, J20 and J30 problem classes of Problem Set A for the best performing ISES configuration (**MaxRestart**=30, quadratic sampling) and the B&B approach of [Schwindt, 1998] (labeled $B\&B_{S98}$). The first column, (N_{opt}), is interpreted differently in the case of each algorithm: for $B\&B_{S98}$, it indicates the number of problems solved to optimality; for ISES, it indicates the number of problems for which the same solution as $B\&B_{S98}$ was obtained. The second column, N_{feas}, gives the number of problems feasibility solved by both approaches. The last column, N_{impr}, makes sense only for ISES, indicating the number of problems for which ISES finds better solutions than $B\&B_{S98}$. Average ISES solution times on J10, J20, J30 problems were 0.87, 5.76 and 17.97 seconds respectively.

We can observe the following:

- On the smaller J10 and J20 problems the exact $B\&B_{S98}$ procedure dominates ISES, but on both the sets ISES finds all feasible solutions.
- on J30 ISES finds 2 feasible solutions (on all runs) that elude $B\&B_{S98}$ and (on average) improves on 20 solutions produced by $B\&B_{S98}$.
- as problem size increases, from J10 to J30, ISES finds lower-makespan solutions than $B\&B_{S98}$ on increasing numbers of problems.

Table 2 shows the relative deviation of the makespans produced by ISES from those produced by B&B for both $ISES_{best(n)}$ and $ISES_{avg(n)}$ (with same configuration as above). We can see that:

- the average deviation in solution quality across all problems is less that 1.6%.
- the differential in solution quality progressively decreases as problem size is increased.

In Table 3 we indicate the effects on solution quality of varying the MCS sampling strategy and the number of restarts on the J30 problem subset. It can be seen that increasing the number of restarts always has a positive effect, and that the more accurate (quadratic) sampling produces a significant improvement.

Table 1: Set A (5 runs, 100 secs, quadratic sampling)

Set	Algorithm	N_{opt}	N_{feas}	N_{impr}
J10	$B\&B_{S98}^{(100secs)}$	187	187	–
	$ISES_{best(5)}^{(100secs)}$	154	187	0
	$ISES_{avg(5)}^{(100secs)}$	151.38	187.00	0.0
J20	$B\&B_{S98}^{(100secs)}$	157	184	–
	$ISES_{best(5)}^{(100secs)}$	120	184	7
	$ISES_{avg(5)}^{(100secs)}$	114.80	184.0	6.39
J30	$B\&B_{S98}^{(100secs)}$	115	183	–
	$ISES_{best(5)}^{(100secs)}$	101	185	26
	$ISES_{avg(5)}^{(100secs)}$	98.38	185.00	20.79

Table 2: Set A: Average and best deviation (100 secs, quadratic sampling)

270 Problems	J10	J20	J30
$\Delta mk_{best(5)}\%$	1.46	1.26	0.76
$\Delta mk_{avg(5)}\%$	1.54	1.52	1.17

On balance, the comparative results on Problem Set A suggest that ISES provides a scalable alternative to exact B&B solution procedures on "hard" RCPSP/max problems. It is worth mentioning that in [Franck and Neumann, 1998] results with similar size problems (number of activities in the range {10,15,20} with similar parameters for PROGEN/max) are reported, and the heuristic approach proposed there compared quite badly with respect to B&B.

5.3 Problem Set B

Table 4 reports performance results for ISES on Problem set B using the same metrics originally used in the B&B study of [Schwindt, 1998]. These include:

- $\Delta_{LB}\%$ – the average relative deviation from the lower bound computed by using [Heilmann and Schwindt, 1997];
- $N_{opt}\%$ – the percentage of optimal solutions found (i.e., solutions that equal the best known lower bound or are proved optimal by a B&B);
- $N_{feas}\%$ – the percentage of problems solved to feasibility.

We compare the performance of ISES (**MaxRestart**=10, linear sampling) with all recently reported branch and bound approaches for RCPSP/max, including those of [De Reyck and Herroelen, 1998] (labeled $B\&B_{dRH}$), [Möhring *et al.*, 1998] (labeled $B\&B_{M98}$), [Schwindt,

Table 3: J30: Number of restarts vs. type of sampling

Sampling	linear		quadratic	
MaxRestart	10	30	10	30
$\Delta mk_{best(5)}\%$	1.37	1.02	1.11	0.76
$\Delta mk_{avg(5)}\%$	1.77	1.37	1.60	1.17

Table 4: Set B: 5 independent runs (linear sampling)

1080 Problems	$\Delta_{LB}\%$	$N_{opt}\%$	$N_{feas}\%$
$B\&B_{dRH}^{(100secs)}$	–	56.1	93.40
$B\&B_{M98}^{(100secs)}$	10.30	64.90	94.70
$B\&B_{S98}^{(100secs)}$	7.04	62.5	98.06
$B\&B_{D98}^{(100secs)}$	4.40	71.20	98.06
PR_{FN98}^{best}	8.00	56.80	97.50
$ISES_{best(5)}^{(100secs)}$	7.60	59.35	98.06
$ISES_{avg(5)}^{(100secs)}$	8.18	58.80	97.89
$ISES_{best(5)}^{(300secs)}$	7.01	59.26	98.06
$ISES_{avg(5)}^{(300secs)}$	7.51	58.91	97.94
$ISES_{best(5)}^{(500secs)}$	6.96	59.44	98.06
$ISES_{avg(5)}^{(500secs)}$	7.41	58.98	97.89

1998] (labeled $B\&B_{S98}$), and [Dorndorf *et al.*, 1998] (labeled $B\&B_{D98}$). We also include the above mentioned heuristic procedure of [Franck and Neumann, 1998] (labeled PR_{FN98}^{best}). In this case, no decomposition strategy/priority rule pair was found to outperform all others on all problems, so we include in Table 4 the results obtained by using the best performing decomposition scheme and, for each problem, taking the best solution found by 10 priority rules.

Several observations follow from Table 4:

- ISES is one of three approaches that is able to find all feasible solutions (98.06%) within a 100 second time bound. In this regard, ISES outperforms PR_{FN98}, previously the best heuristic procedure known for RCPSP/max. In fact, ISES also appears to be more robust than PR_{FN98}; across all 15 runs performed with ISES, no more than 3 feasible solutions were ever missed on a single run.
- all B&B approaches except $B\&B_{dRH}$ find higher percentages of optimal solutions than ISES. However, with regard to deviation from lower bound solutions $\Delta_{LB}\%$ ISES ranks third behind $B\&B_{D98}$ and $B\&B_{S98}$, and is in fact fairly comparable to $B\&B_{S98}$ with a 300 second time limit. (Note that $\Delta_{LB}\%$ is influenced by the number of feasible solutions found; solving the more difficult problems typically increases the deviation [Dorndorf *et al.*, 1998]).
- Contrasting the performance of ISES at different time limits, significant improvement is obtained in increasing the time limit from 100 to 300 seconds but the further increase to 500 seconds achieves only limited further improvement.

The clearly dominating procedure on Problem Set B is $B\&B_{D98}$ of [Dorndorf *et al.*, 1998]. It is interesting to note that this approach exploits resource constraint propagation rules that could be straightforwardly added to ISES. We are currently investigating this possibility.

The average solution times obtained for ISES at each different time limit give some additional insight into its behavior on this problem set, and are as follows: 43.91 seconds (100 second limit), 72.21 seconds (300 second limit) and 84.79 seconds (500 second limit). The fact that average solution times rise fairly slowly in proportion to increases in the time limit indicates that there are only a relatively small percentage of problems that cannot be efficiently solved. For example, in examining particular runs of ISES, the percentage of solvable problems that require over 100 seconds for solution is seen to be $\tilde{2}5\%$.

Table 5: Set B: 5 runs (quadratic sampling)

1080 Problems	$\Delta_{LB}\%$	$N_{opt}\%$	$N_{feas}\%$
$ISES_{best(5)}^{(500secs)}$	6.99	59.54	98.06
$ISES_{avg(5)}^{(500secs)}$	7.48	58.87	97.93

Finally, Table 5 indicates the performance effect of incorporating the more-comprehensive quadratic MCS sampling strategy with a 500 second time bound. In this case, the average solution time increases just slightly to 91.49 seconds, and as Table 5 shows, there is virtually no improvement in solution quality. This ineffectiveness can also be explained by recalling the prior discussion of the characteristics of Problem Set B. The level of resource contention in these problems is only small to moderate, resulting in contention peaks that are relatively smooth and contain few MCSs. Accordingly, there are few MCSs that are not found by linear sampling and a more accurate strategy adds very little.

6 Conclusions

This paper investigates the use of an iterative sampling procedure to solve RCPSP/max, a complex optimization problem. The ISES procedure uses a combination of constraint-guided greedy search and randomization. The greedy search procedure utilizes analysis of "minimal conflict sets" (MCSs) to identify where additional ordering constraints are required to avoid resource contention. MCS analysis is integrated with least commitment principles related to retaining temporal flexibility to provide heuristics for focusing the search. Randomization is introduced in a way that incorporates the bias of these greedy search heuristics. This provides a basis for smoothing the decisions of the deterministic algorithm and for broadening the search to include heursitically equivalent search paths in the space, thus enhancing the probability of finding better quality solutions.

A further contribution is represented by the approximate computation of MCSs, which avoids the basic exponential computation and attempts to first select the most critical minimal conflicts at a low polynomial cost.

ISES has been compared with the best existing approaches to RCPSP/max. It outperforms several systematic and heuristic approaches on different reference problem sets. In particular, ISES is shown to perform quite well in situations where, due to problem characteristics such heavy resource contention, the search space is quite large and becomes a serious obstacle for systematic approaches. In this case, our non-systematic random approach demonstrates an advantageous trend in comparison to the best known systematic algorithm (in terms of both the number and the quality of the solutions provided) as problem size is increased. It is worth noting that our method relies mainly on its composite search strategy and heuristics, rather than on a set of propagation rules for early pruning of the search space. The addition of the latter could be an interesting direction for future research.

Acknowledgments

Authors would like to thank Dr. Christoph Schwindt for creating the repository that made possible the experimental comparison described in this work and for giving prompt clarifications on its use. We also thank the various authors who have made available their updated unpublished works on this topic. Amedeo Cesta and Angelo Oddi's work is supported by Italian Space Agency, by CNR Committee 12 on Information Technology (Project SCI*SIA), and CNR Committee 4 on Biology and Medicine. Stephen F. Smith's work has been sponsored in part by the National Aeronautics and Space Administration under contract NCC 2-976, by the US Department of Defense Advanced Research Projects Agency under contract F30602-97-20227, and by the CMU Robotics Institute.

References

[Bartusch *et al.*, 1988] M. Bartusch, R. H. Mohring, and F. J. Radermacher. Scheduling Project Networks with Resource Constraints and Time Windows. *Annals of Operations Research*, 16:201–240, 1988.

[Cesta *et al.*, 1998a] A. Cesta, A. Oddi, and S.F. Smith. Profile Based Algorithms to Solve Multiple Capacitated Metric Scheduling Problems. In *Proceedings of the Fourth Int. Conf. on Artificial Intelligence Planning Systems (AIPS-98)*, 1998.

[Cesta *et al.*, 1998b] A. Cesta, A. Oddi, and S.F. Smith. Scheduling Multi-Capacitated Resources under Complex Temporal Constraints. Technical Report CMU-RI-TR-98-17, Robotics Institute, Carnegie Mellon University, 1998.

[Cheng and Smith, 1994] C. Cheng and S.F. Smith. Generating Feasible Schedules under Complex Metric Constraints. In *Proceedings 12th National Conference on AI (AAAI-94)*, 1994.

[Cheng and Smith, 1997] C. Cheng and S.F. Smith. Applying Constraint Satisfaction Techniques to Job Shop Scheduling. *Annals of Operations Research*, 70:327–357, 1997.

[De Reyck and Herroelen, 1998] B. De Reyck and W. Herroelen. A Branch-and-Bound Procedure for the Resource-Constrained Project Scheduling Problem with Generalized Precedence Relations. *European Journal on Operations Research*, 1998. to appear.

[Dorndorf *et al.*, 1998] U. Dorndorf, E. Pesch, and T. Phan Huy. A Time-Oriented Branch-and-Bound Algorithm for Resource-Constrained Project Scheduling with Generalized Precedence Relations. Technical report, University of Bonn, Faculty of Economics, 1998.

[Franck and Neumann, 1998] B. Franck and K. Neumann. Resource Constrained Project Scheduling Problems with Time Windows – Structural Questions and Priority-Rule Methods. Technical Report WIOR-492, Universität Karlsruhe, 1998. (Revised November 1998).

[Heilmann and Schwindt, 1997] R. Heilmann and C. Schwindt. Lower Bounds for RCPSP/max. Technical Report WIOR-511, Universität Karlsruhe, 1997.

[Kolisch *et al.*, 1998] R.r Kolisch, C. Schwindt, and A. Sprecher. Benchmark Instances for Project Scheduling Problems. In J. Weglarz, editor, *Handbook on Recent Advances in Project Scheduling*. Kluwer, 1998.

[Laborie and Ghallab, 1995] P. Laborie and M. Ghallab. Planning with Sharable Resource Constraints. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI-95)*, 1995.

[Möhring *et al.*, 1998] R. Möhring, F. Stork, and M. Uetz. Resource Constrained Project Scheduling with Time Windows: A Branching Scheme Based on Dynamic Release Dates. Technical Report 596/1998, Fachbereich Mathematick, Technische Universität Berlin, 1998.

[Nuijten and Aarts, 1996] W.P.M. Nuijten and E.H.L. Aarts. A Computational Study of Constraint Satisfaction for Multiple Capacitated Job Shop Scheduling. *European Journal of Operational Research*, 90(2):269–284, 1996.

[Nuijten and Le Pape, 1998] W.P.M. Nuijten and C. Le Pape. Constraint-Based Job Shop Scheduling with ILOG-SCHEDULER. *Journal of Heuristics*, 3:271–286, 1998.

[Oddi and Smith, 1997] A. Oddi and S.F. Smith. Stochastic Procedures for Generating Feasible Schedules. In *Proceedings 14th National Conference on AI (AAAI-97)*, 1997.

[Schwindt, 1996] C. Schwindt. Generation of Resource Constrained Project Scheduling Problems with Minimal and Maximal Time Lags. Technical Report WIOR-489, Universität Karlsruhe, 1996.

[Schwindt, 1998] C. Schwindt. A Branch and Bound Algorithm for the Resource-Constrained Project Duration Problem Subject to Temporal Constraints. Technical Report WIOR-544, Universität Karlsruhe, 1998.