Sprecher, Arno

Network decomposition techniques for resource-constrained project scheduling

Manuskripte aus den Instituten für Betriebswirtschaftslehre der Universität Kiel, No. 505

Manuskripte
aus den
Instituten für Betriebswirtschaftslehre
der Universität Kiel

No. 505

Network Decomposition Techniques for Resource-Constrained Project Scheduling [1]

Arno Sprecher

Abstract

Numerous exact algorithms have been developed for solving the resource-constrained project scheduling problem. Experimental studies have shown that currently even projects with only 60 activities cannot be optimally solved within a reasonable amount of time. Therefore heuristics employing genetic concepts, sampling strategies, simulated annealing or taboo search have been developed. Additionally truncated versions of the branch-and-bound algorithms are studied. By limiting the CPU-time, the total number of node evaluations, or the number of branching alternatives, the solution time is reduced at the expense of the quality of the generated schedules.

The purpose of this paper is to study a combination of exact and heuristic elements. The project to be considered is decomposed into subprojects, the related subproblems are optimally solved, and the solutions are concatenated. The solution strategy has been implemented and tested on the benchmark instances provided by ProGen.

The numerical results show that the decomposition approach outperforms the truncated version of the branch-and-bound algorithm employed. On average, the quality of the overall solution depends on the size of the subproblems, and the quality of the solutions of the subproblems. Consequently the approach will benefit from the progress made in the development of exact solution procedures.

Keywords: Project Scheduling, Resource Constraints, Branch-and-Bound, Decomposition, Computational Results.

1 Introduction

In resource-constrained project scheduling the projects consist of activities that have a deterministic duration and a deterministic usage of renewable resources. The resource availability is limited. Precedence relations between some of the activities define further constraints on the temporal arrangement of the activities. The objective is the minimization of the makespan. Since the resource-constrained project scheduling problem is NP-hard (cf. [5]) branch-and-bound algorithms (see, e.g., [3], [4], [12], [14], [16]) and heuristics (see, e.g., [1], [6], [8], [9]) have been developed.

The purpose of this paper is to study a combination of heuristic and exact solution approaches. The project to be considered is decomposed into subprojects, the related subproblems are solved optimally, and the solutions of the subproblems are concatenated to a solution of the original problem.

We proceed as follows: Section 2 describes the problem more thoroughly and summarizes the problem parameters. Section 3 sketches the exact solution procedure employed. Section 4 presents the decomposition into subprojects, the subproblem definition strategy and the concatenation strategy. Section 5 presents the computational results. Conclusions are drawn in Section 6.

2 The Model

We employ the classification scheme and notation suggested in [2]. The project we consider consists of n activities (jobs, tasks). Due to technological requirements, precedence relations between some of the activities

enforce that an activity j may not be started before all its predecessors h, $h \in Pred(j)$, are finished. The structure of the project is depicted by a so-called activity-on-node (AON) network where the nodes and the arcs represent the activities and precedence relations, respectively. The network is acyclic and numerically labeled, that is an activity j has always a higher number than all its predecessors. W.l.o.g. activity 0 is the dummy start activity (source) and activity $n+1$ is the dummy finish activity (sink). We can denote the precedence relations by the set H of pairs (h,j) with $h \in Pred(j)$, $j = 0,\ldots,n+1$.

The activities may not be preempted, i.e., an activity once started has to be completed without interruption. Performing activity j takes p_j periods and uses renewable resources from a set $\mathcal{R}^\rho$. Given a horizon, that is, an upper bound T on the project's makespan, R^ρ_{kt} units of renewable resource k, $k \in \mathcal{R}^\rho$, are available in a period t, $t = 1,\ldots,T$. An activity j, $j = 1,\ldots,J$, requires r^ρ_{jk} units of renewable resource k, $k \in \mathcal{R}^\rho$, each period activity j is in process. The objective is to find a makespan minimal schedule that meets the constraints imposed by the precedence relations and the limited availability of the renewable resources. The problem can be modeled as a linear program. Following the classification suggested in [2] the problem is referred to as $PS \mid prec \mid C_{max}$.

3 The Optimal Solution Procedure

In this section we summarize the branch-and-bound algorithm employed to solve the partial projects. For the details, we refer to [14]. The search is guided by the precedence tree introduced by Patterson et al. (cf. [13]). The nodes of the precedence tree correspond to the nodes of the branch-and-bound tree. The root node 1 of the tree is given by the single start activity and the leaves are copies of the only finish activity J. The descendents of a node j within the precedence tree are built by the activities that are eligible after scheduling the activities on the path leading from the root node 1 to node j. Thereby, an unscheduled activity is called eligible, if all its predecessors are scheduled. An activity is firstly considered for scheduling when all of its predecessors are scheduled. The start time of the activity under consideration is the lowest feasible start time, which (a) is not less than the start time of the activity most recently scheduled, and (b) does not violate the precedence or resource constraints. If scheduling of the current activity is not feasible w.r.t. the bounds imposed by the latest start time then backtracking is performed. On this level the next untested eligible activity is selected. If there is no untested eligible activity left then backtracking to the previous level is performed. At this level the next eligible activity is chosen. The procedure terminates if backtracking leads to level 0 of the branch-and-bound tree or if the makespan of a solution found coincides with a lower bound.

In order to speed up the convergence we employ the following lower bounds and dominance concepts (cf. [14]). First, the **critical path based bound** is implemented by calculating the latest starting times of the activities for a given makespan. If an improved solution is found then the latest starting times are adapted to guarantee that solutions to be found later will improve the current best solution.

Second, the **bound LB3Mod** is a variant of the bound provided by Mingozzi et al. (cf. [12]). The bound is valid when resource availability is constant. In a preprocessing routine we generate a priority list of the set

of activities. Each activity g in the list is assigned another list of activities containing all the activities that can be performed simultaneously with activity g without violating the precedence and resource constraints, and which are not placed before activity g in the priority list. During the enumeration at each node of the branch-and-bound tree an activity which is scheduled is eliminated from the priority list, and again added to the list through backtracking. The remaining activities in the priority list are considered from the beginning to the end of the priority list. Starting with $LB = 0$, and no activity marked, the first non-eliminated and non-marked activity is selected and its duration is added to LB. Afterwards the elements from its list are marked. The procedure continues as far as unmarked elements can be found in the priority list. The finally valid LB is then a bound on the minimal time necessary to complete the partial schedule. In our implementation we have built three priority lists and determined the bound LB3 as the maximum of the bounds obtained from all the priority lists. The lists have been constructed as described in [14].

Third, the **Extended Single Enumeration Rule** excludes multiple enumeration of one and the same (partial) schedule. Identical partial schedules would be obtained if two activities that are eligible on a certain level have identical starting times, and if the starting times are unaffected by the sequence the activities are scheduled in.

Fourth, the **Local Left Shift Rule** reduces the enumeration to the so-called semi-active schedules (see [15]), where the start time of none of the activities can be reduced by a single period without delaying any other activity or producing a constraint violation.

Fifth, the **Global Left Shift Rule** studies global left-shifts, i.e., left shifts that cannot be obtained by a local left. If an activity, eligible on the current level, can be globally left-shifted, then all the continuations of the current partial schedule are dominated. Consequently backtracking can be performed.

Sixth, the **Contraction Rule** employs completion times of earlier considered activities to define an upper bound on the starting time of the current activity. The rule reduces feasibility of global left-shifts on later levels. The bounds are derived from the completion times that have been obtained through scheduling, locally or globally left-shifting.

Seventh, the **Set-Based Dominance Rule** compares the quality of the current partial schedule with the quality of an earlier studied partial schedule. The partial schedules have identical sets of scheduled activities. If the starting time of the current activity is greater than or equal to the maximum completion time of the partial schedules studied earlier then the current activity can be skipped.

Eighth, the **Permutation Rule** excludes a partial schedule from continuation if by permutation of two activities an earlier studied schedule can be derived.

4 Decomposition and Concatenation Strategies

In order to describe the decomposition strategies we need some definitions. First, for sake of simplicity, we use $J = n + 1$, and denote a precedence feasible sequence of the activities $0, \ldots, n + 1$, by $Seq_J = [g_1, \ldots, g_J]$, that is, it is $Pred(g_{i+1}) \subseteq \bigcup_{j=1}^{i} \{g_i\}$ for $i = 1, \ldots, J - 1$. Second, for a given precedence feasible sequence $Seq_J = [g_1, \ldots, g_J]$, we define by $ESS(Seq_J)$ the earliest start schedule obtained by successively

scheduling the activities $g_1, \ldots, g_J$. An activity g_{i+1} is thereby scheduled at its earliest precedence and resource feasible starting time with respect to the left-over capacity of the already scheduled partial sequence $Seq_i = [g_1, \ldots, g_i]$. Note, for a given sequence $ESS(Seq)$ is unique. We denote the starting time and the completion time of an activity g_i by S_{g_i} and C_{g_i}, respectively. Third, for a given schedule S with starting times $S_i, i = 1, \ldots, J$, we use $Seq(S) = [g_1, \ldots, g_J]$, as a sequence of the activities $g_1, \ldots, g_J$ with $S_{g_i} \leq S_{g_{i+1}}$, $i = 1, \ldots, J-1$. Note, for a given schedule S the sequence $Seq(S)$ is not unique in general.

A precedence feasible sequence can be generated randomly: We define probabilities p_j to select randomly an activity j out of the eligible set E_i, $i = 1, \ldots, J$. The probabilities are specified in accordance with given non-negative priority value Π and criterion σ, $\sigma \in \{min, max\}$. The criterion decides to prefer the selection of activities associated with smaller (min) or larger (max) priority values. For a certain level i and eligible set E_i, the probabilities prob_j (minimization) and $\overline{\text{prob}_j}$ (maximization), respectively (cf. [9]), $j \in E_i$ are

$$\text{prob}_j := \frac{\max\{\Pi(k); k \in E_i\} - \Pi(j) + 1}{\sum\limits_{e \in E_i} (\max\{\Pi(k); k \in E_i\} - \Pi(e) + 1)} \quad , \quad \overline{\text{prob}_j} := \frac{\Pi(j) - \min\{\Pi(k); k \in E_i\} + 1}{\sum\limits_{e \in E_i} (\Pi(e) - \min\{\Pi(k); k \in E_i\} + 1)}.$$

Considering a priority rule we can now generate a precedence feasible sequence $Seq_J = [g_1, \ldots, g_J]$ by selecting g_i out of E_i, $i = 1, \ldots, J$, in accordance with the specified probabilities. The sequence will be used to determine the subproblems we need to solve.

For the definition of the subproblems we have to determine (1) the size of the subprojects, (2) the activities of the subprojects, and (3) the constraints of the subproblems.

The subproblems can be of deterministic size (DS) or stochastic size (SS). Assuming that the non-negative integers $n_1, \ldots, n_s$ fulfill $\sum_{l=1}^{s} n_l = n$, the deterministic sizes of s subprojects are denoted by $[s; 0; n_1, \ldots, n_s]$ and the stochastic sizes of s subprojects are denoted by $[s; \delta; n_1, \ldots, n_s]$. In the deterministic case subproject l, $l = 1, \ldots, s$, consists of $\tilde{n}_l = n_l$ activities. In the stochastic case subproject l, $l = 1, \ldots, s$, consists of $\tilde{n}_l$, $\tilde{n}_l \in [n_l - \delta; n_l + \delta]$, activities. We call the tuple $[s; \delta; \tilde{n}_1, \ldots, \tilde{n}_s]$ a size specification, and the parameter δ disturbance. If the disturbance is non-zero we proceed as described in Table 1 to guarantee $\sum_{l=1}^{s} \tilde{n}_l = n$.

Step 1: If all the subprojects have been assigned a size, then STOP;

Step 2: If there is only one subproject l to which no size has been assigned then define $\tilde{n}_l := n_l$ and goto Step 1;

Step 3: Randomly select two subproject l and l' the size of which is not fixed;

Step 4: Randomly select $\bar{\delta} \in [-\delta, +\delta]$;

Step 5: Define $\tilde{n}_l := n_l - \bar{\delta}$ and $\tilde{n}_{l'} := n_{l'} + \bar{\delta}$;

Step 6: Remove subprojects l and l' from the set of projects the size of which is not fixed and goto Step 1.

Table 1: Determination of the Subproject Size

The activities of the subprojects are determined with respect to a given size specification $[s; \delta; \tilde{n}_1, \ldots, \tilde{n}_s]$ and a given sequence $Seq_J = [g_1, \ldots, g_J]$. The set of activities $\mathcal{J}_l$ of subproject l, $l = 1, \ldots, s$, is determined

by

$$\mathcal{J}_l = \bigcup_{j=\sum_{p=1}^{l-1} n_p+2}^{\sum_{p=1}^{l} n_p+1} \{g_j\}$$

For the definition of subproblem l, $l = 1, \ldots, s$ we use the reduced set of precedence relations $H_l := \{(i,j) \in H; (i,j) \in \mathcal{J}_l \times \mathcal{J}_l\}$. We define $\mathcal{J}_0 := \{0\}$, and $S_0 := C_0 := 0$. If the solution, i.e., the start times S_j, $j \in \mathcal{J}_{l-1}$, of the preceeding subproblem $l-1$ are known, we can formulate the constraints of subproblem l. Using the binary decision variables

$$x_{jt} = \begin{cases} 1 & , \text{ if activity } j \text{ is completed at the end of period } t \\ 0 & , \text{ otherwise.} \end{cases}$$

we obtain the subproblems as described in Table 2. By $S^{max}(l-1)$ ($C^{max}(l-1)$) we denote the maximum start time (completion time) of the activities out of $\bigcup_{v=0}^{l-1} \mathcal{J}_v$ within the optimal solutions of subproblems v, $v = 0, \ldots, l-1$. The subproblems are successively solved. The solutions of subproblems $0, \ldots, l-1$,

---

$$\text{Minimize } \Phi_l(x) = \max_{j \in \mathcal{J}_l} \{ \sum_{t=EF_j}^{LF_j} t \cdot x_{jt} \} \tag{1}$$

s.t.

$$\sum_{t=EF_j}^{LF_j} x_{jt} = 1 \qquad j \in \mathcal{J}_l \tag{2}$$

$$S^{max}(l-1) \leq S_j \qquad j \in \mathcal{J}_l \tag{3}$$

$$\sum_{t=EF_h}^{LF_h} t \cdot x_{ht} \leq \sum_{t=EF_j}^{LF_j} (t - p_j) x_{jt} \qquad (h,j) \in H_l \cup ((\bigcup_{p=1}^{l-1} \mathcal{J}_p \times \mathcal{J}_l) \cap H) \tag{4}$$

$$\sum_{j \in \bigcup_{p=1}^{l} \mathcal{J}_p} r_{jk}^{\rho} \sum_{q=\max\{t, EF_j\}}^{\min\{t+p_j-1, LF_j\}} x_{jq} \leq R_{kt}^{\rho} \qquad k \in \mathcal{R}^{\rho}, t = S^{max}(l-1), \ldots, T \tag{5}$$

$$x_{jt} \in \{0,1\} \qquad j \in \mathcal{J}_l, t = EF_j, \ldots, LF_j \tag{6}$$

Table 2: Constraints and Objective of Subproblem l

are explicitly taken into account to define the constraints of subproblem l. The objective (1) realizes the minimization of the makespan. The constraints (2) ensure that each task is assigned a completion time. The constraints (3) define the lower bound on the start time by the maximum start time of an activity out of the preceding subproject. The constraints (4) guarantee precedence feasibility of the compound solution. The constraints (5) ensure that resource availability is met by the compound solution. Note, since the constraints (3) and (5) couple the constraints of successive subprojects the problems have to be solved consecutively. The general scheme of the algorithm is displayed in Table 3. In Step 1 the variables are initialized. In Step 2, the sequences are generated with respect to the given priority rule. In Step 3, a sequences $Seq[q]$ is selected,

Step 1	$T^* := MaxInt$; read size specification $[s; \delta; n_1, \ldots, n_s]$, $l = 1$;
Step 2	Use priority rule Π to generate sequence $Seq[q]$, $q = 1, \ldots, NrOfSeq$; $q := 1$;
Step 3	Select sequence $Seq[q]$; determine $ESS(Seq(q))$ and $Seq' = Seq(ESS(Seq[q]))$;
Step 4	Define solution of subproblem SP_0; $l = 1$;
Step 5	Use Seq' and solutions of subproblems SP_v, $v = 0, \ldots, l-1$, to define subproblem SP_l; solve subproblem SP_l with at most nps node evaluations; store best solution found;
Step 6	If $(l < s)$ then $\{l := l + 1$; goto Step 5;$\}$
Step 7	Compose solutions of subproblems $SP_1, \ldots, SP_s$ to a schedule S of the original problem; build earliest start schedule $ESS(Seq(S))$ and obtain makespan S_J;
Step 8	If $(S_J < T^*)$ then $\{T^* := S_J$; store schedule $ESS(Seq(S))$;$\}$
Step 9	If $(q < NrOfSeq)$ then $\{ q := q + 1$; goto Step 3;$\}$ else STOP;

Table 3: Algorithmic Scheme

the earliest start schedule $ESS(Seq[q])$ is built, subsequently, a related start time monotone sequence Seq' is determined. In Step 4, subproblem 0 and its solution is initialized. In Step 5, the solutions of the previously solved subproblems are employed to define the current subproblem SP_l. Note, since in Step 5 we allow only nps node evaluations to solve the subproblem, it can happen that the problem is not optimally solved, or that, due to application of dominance rules no feasible solution is found. In the former case we use the best solution found so far. In the latter case we proceed with the next sequence. If for all the subproblems a feasible solution has been found then, in Step 7, a solution of the original problem is obtained by the earliest start schedule related to the start time monotone sequence of schedule S.

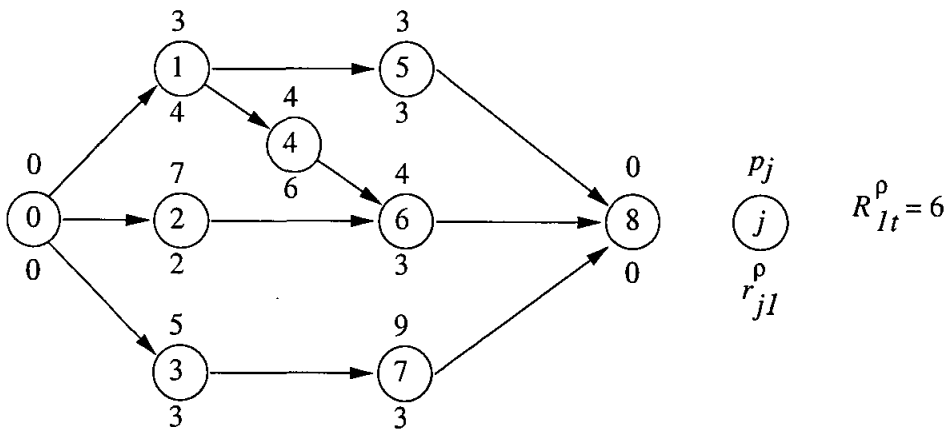

Figure 1: Example Network

We consider the project instance of Figure 1. The project consists of $n = 7$ non-dummy activities. The only renewable resource 1 has an availablility of 6 units per period. We assume that the sequence $Seq = [0, 1, 5, 4, 2, 6, 3, 7, 8]$ has been stochastically determined by using a priority rule Π. The related earliest start

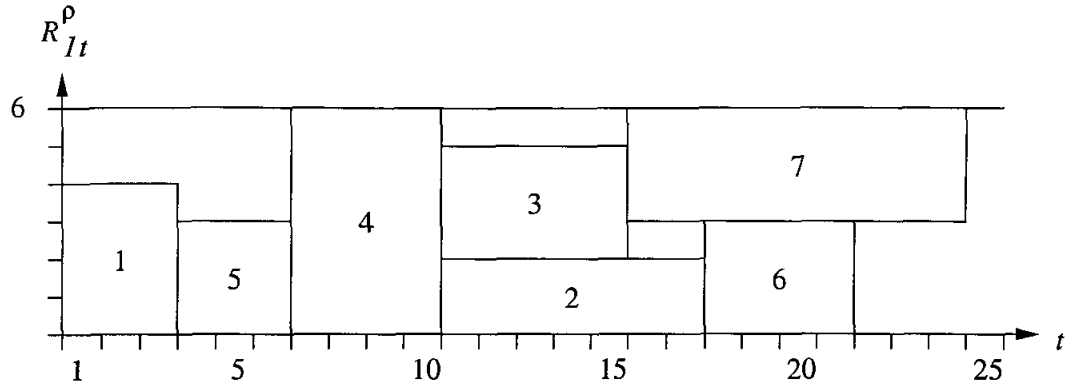

Figure 2: Earliest Start Schedule of Sequence $Seq = [0, 1, 5, 4, 2, 6, 3, 7, 8]$

schedule $ESS(Seq)$ is displayed in Figure 2. It has a makespan of 24 periods. A sequence Seq' that relates to the earliest start schedule is given by $Seq' = [0, 1, 5, 4, 3, 2, 7, 6, 8]$. Note, the sequence Seq' that relates to the earliest start schedule does not coincide with the sequence Seq. We use $s = 2$, $\tilde{n}_1 = 4$, and $\tilde{n}_2 = 3$, and obatin two non-trivial subprojects. The first one consists of actvities $\mathcal{J}_1 = \{1, 3, 4, 5\}$, the second one consists of activities $\mathcal{J}_2 = \{2, 6, 7\}$. We determine a makespan minimal solution of the first subproject and obtain $S_0 = 0$, $S_1 = 0$, $S_3 = 3$, $S_5 = 3$, and $S_4 = 8$ with optimal makespan of 12 periods. Subsequently, the second subproject is optimally solved, and we determine $S_2 = 12$, $S_7 = 12$, and $S_6 = 19$ with optimal makespan of 11 periods. The compound solution is displayed in Figure 3. It has a makespan of 23 periods. Note, when solving the second subproject, according to the subproblem definition (see Table 2), additional

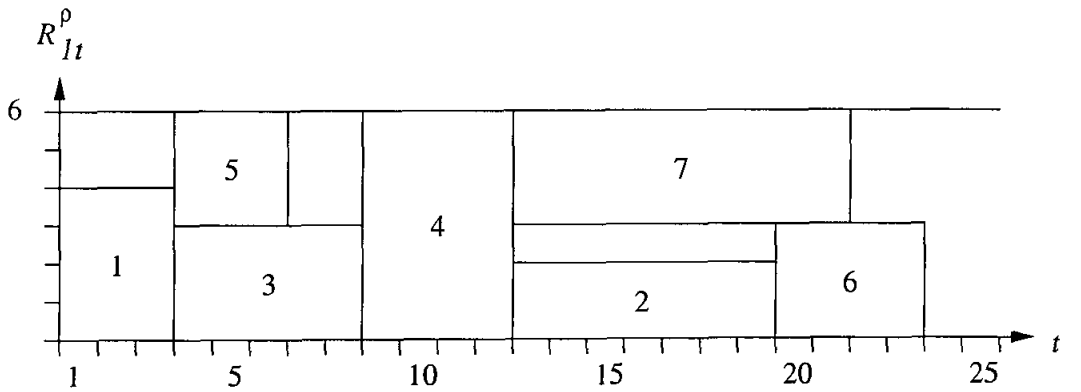

Figure 3: Compound Solution

constraints have to be considered: First, the precedence constraints of (3), i.e., $S_4 \leq S_2$, $S_4 \leq S_7$, and the constraint $S_4 \leq S_6$, which is implicated by the precedence relation $(4, 6)$. Second, the reduced availabilities $R^{\rho}_{1t}{}'$ of resource 1, in periods $S^{max}(1) = 9, \ldots, C^{max}(1) = 12$.

However, the compound solution does not necessarily have a minimal makespan. A solution with a lower makespan is displayed in Figure 4.

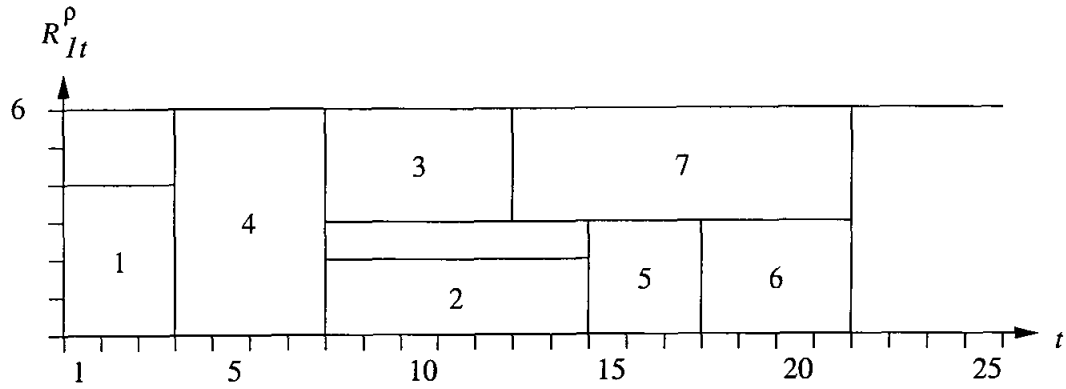

Figure 4: Optimal Solution

5 Computational Results

In this section we present the results of our computational analysis. The algorithm has been coded in GNU C, Version 2.7.2.1, and implemented on a personal computer (Pentium, 166 MHz, 64 MB) operating under LINUX. The data structures allow, through parameter adjustment, to deal with projects of any size. The algorithm studied in [14] served as the basis of our implementation. The algorithm has been tested on the instance sets J30, J60, J90, and J120 that have been generated by ProGen (see [10] and [11]). We have tested several priority rules to build the initial sequences (schedules) for the decomposition of the problems. The rules are derived from the precedence relations as well as from the resource usages and availabilities. For the instance set J30 and size specification $[3; 0; 10, 10, 10]$ each of the priority rules has been employed to generate 200 sequences (schedules). The best solutions have been compared after 10, 20, 50, 100 and 200 sequences have been decomposed, that is, after 30, 60, 150, 300, 600 subprojects of 10 non-dummy activities have been solved. The lowest average deviations have been achieved by using minimum latest finish time (MinLFT) and maximum resource-based bound on reflexive transitive successors (MaxRB|$\overline{Succ}$|). We denote the set of reflexive, transitive successors of an activity j by $\overline{Succ}(j)$ and obtain the priority values

$$\Pi_j = LFT_j \quad \text{and} \quad \Pi_j = \max_{k \in \mathcal{R}^{\rho}} \Big(\sum_{s \in \overline{Succ(j)}} p_s \cdot r^{\rho}_{sk} / R^{\rho}_k \Big)$$

Table 4 shows the results for the set J30 (30 non-dummy activities, 480 instances). The sequences have been generated by using MinLFT. Different size specifications have been selected. Rows 1 and 2, specify the decomposition, i.e., the sizes of the subprojects and the disturbance. Row 3, displays the precision, i.e., the number of nodes of the branch-and-bound tree that have been evaluated at most to solve a subproblem.

The remaining rows show the results after different numbers of decomposed sequences, i.e., the average (avg.) and maximum (max.) percentage deviation of the makespan found from the MPM-bound ($\Delta(MPM)$), the average (avg.) and maximum (max.) percentage deviation of the makespan found from the minimal makespan ($\Delta(opt)$), the average (avg.) and maximum (max.) CPU-time (CPU) to solve an instance. Finally, the last row specifies the average (avg.) and maximum (max.) number of subproblems that have been optimally solved NOS. We notice: First, if a (maximum) disturbance $\delta = 3$ is allowed then the average

	Decomposition	(10,10,10)				(15,15)			
		$\delta = 0$		$\delta = 3$		$\delta = 0$		$\delta = 3$	
	Precision[nps]	5,000				10,000			
NrOfSeq		avg.	max.	avg.	max.	avg.	max.	avg.	max.
10	$\Delta(MPM)$[%]	15.29	133.33	15.21	135.42	14.35	127.08	14.27	133.33
	$\Delta(opt)$ [%]	1.43	12.37	1.36	12.07	0.75	13.33	0.67	8.82
	CPU [sec]	0.29	0.82	0.29	0.82	0.34	2.53	0.36	2.47
20	$\Delta(MPM)$[%]	14.90	133.33	14.71	133.33	14.10	127.08	14.01	133.33
	$\Delta(opt)$ [%]	1.14	11.29	0.99	9.68	0.55	13.33	0.47	7.27
	CPU [sec]	0.34	1.21	0.35	1.37	0.45	4.73	0.49	4.39
50	$\Delta(MPM)$ [%]	14.51	133.33	14.29	131.25	13.93	127.08	13.72	122.92
	$\Delta(opt)$ [%]	0.84	7.89	0.66	8.62	0.42	13.33	0.26	4.41
	CPU [sec]	0.50	2.34	0.52	2.75	0.77	11.61	0.86	10.21
100	$\Delta(MPM)$[%]	14.26	133.33	14.06	125.00	13.76	125.00	13.60	120.83
	$\Delta(opt)$ [%]	0.65	6.94	0.49	6.58	0.30	13.33	0.17	4.41
	CPU [sec]	0.76	4.47	0.80	5.11	1.31	23.11	1.50	20.09
200	$\Delta(MPM)$ [%]	14.09	125.00	13.92	125.00	13.67	125.00	13.53	120.83
	$\Delta(opt)$ [%]	0.53	6.94	0.40	6.19	0.22	4.41	0.12	4.41
	CPU [sec]	1.28	8.99	1.37	9.64	2.39	45.02	2.75	39.70
	NOS	599.30	600.00	598.65	600.00	398.66	400.00	396.04	400.00

Table 4: J30 - Average Deviation and CPU-Time versus Decomposition and Number of Decomposed Sequences

deviation is less than the average deviation when no disturbance $\delta = 0$ is allowed. This is in line with general observations for stochastic solution procedures. An attribute, here the sizes of the subproblems to be solved, that does not necessarily relate to an optimal solution of the original problem should not be fixed as a characteristic of the solutions. Second, the more sequences are studied the higher the quality of the solutions. Third, the average and maximum CPU-time seem to grow linearly with the number of sequences to be decomposed. Fourth, at comparable average CPU-times, the average deviation of the size specification that allows larger subprojects to be considered, i.e., $[2;0;15,15]$ ($[2;3;15,15]$) produces a lower average deviation than the one that allows smaller subprojects, i.e., $[3;0;10,10,10]$ ($[3;3;10,10,10]$). The results for $[2;3;15,15]$ are the best that have been obtained by heuristic, non-truncated branch-and-bound, strategy (cf. [7]). Note, the precision of 5,000 (10,000) node evaluations for decomposition $(10,10,10)$ ($(15,15)$) was sufficient to solve nearly all the subproblems to optimality.

Table 5 and Table 6 display the results for the instance set J60 (60 non-dummy activities, 480 instances) at different decompositions and different precisions. Results for other numbers of decomposed sequences can be found in Table 11 and in Table 12 of the Appendix. Since not all the optimal solutions of the instance set could be determined up to now, we have calculated the average deviation $\Delta(hrs)$ from the best solutions available when the analysis in [7] has been performed instead of $\Delta(opt)$. Again we notice: The

	Decomposition	(15,15,15,15)				(20,20,20)			
		$\delta = 3$				$\delta = 3$			
	Precision [nps]	25,000		100,000		25,000		100,000	
NrOfSeq		avg.	max.	avg.	max.	avg.	max.	avg.	max.
100	$\Delta(MPM)$[%]	13.04	112.99	12.92	110.39	12.70	112.99	12.45	110.39
	$\Delta(hrs)$ [%]	1.23	10.91	1.15	10.91	0.95	9.09	0.79	9.09
	CPU [sec]	11.88	98.02	27.35	325.92	17.87	114.75	51.02	447.69

Table 5: J60 - Average Deviation and CPU-Time versus Decomposition

	Decomposition	(30,30)							
		$\delta = 3$							
	Precision [nps]	50,000		100,000		200,000		500,000	
NrOfSeq		avg.	max.	avg.	max.	avg.	max.	avg.	max.
100	$\Delta(MPM)$[%]	12.56	116.88	12.27	115.58	12.14	112.99	11.94	111.69
	$\Delta(hrs)$ [%]	0.78	8.331	0.55	8.331	0.49	7.96	0.36	7.96
	CPU [sec]	41.23	206.26	72.29	411.88	135.19	813.63	303.21	2,077.42

Table 6: J60 - Average Deviation and CPU-Time versus Precision

larger the subprojects or the higher the precision of the solutions of the subproblems, the better the quality of the compound solutions. However, although the average CPU-time increases only sub-linearly with the precision, the increase is substantial.

To reduce the CPU-time required we have generated a certain number of sequences and selected only a portion of them to be decomposed. That is, by the selection of the best sequences we have intensified the search in the neighborhood of good schedules. Using 200 sequences (schedules) the best 5, 15, and 50 have been decomposed. We compare the results of the evaluation of a subset of the set of sequences with the evaluation of the entire set of sequences. The comparison is to be found in Table 7. We see, if, e.g., the best 5 instead of the first 10 sequences or the best 15 instead of the first 50 sequences are employed to form the subproblems, then the average CPU-time is substantially reduced without decrease of quality, i.e., increase of the average deviation. If all the sequences are used, i.e., 200, then the evaluation of all the sequences is better then the evaluation of best 50 sequences. However, approximately four times the CPU-time is required.

In the next experiment we have examined the influence of the number of initially generated sequences (schedules) on the quality of the solutions. The results are displayed in Table 8. We have generated 200, 400, and 1,000 sequences (schedules). The table shows the results after the evaluation of the best 20 sequences. Results from composing 5, 10, 15, and 50 sequences can be found in Table 13 of the Appendix. Beside the abbreviations previously introduced, we employ $MPM(BestSeq)$ to denote the average deviation of the best makespan found through the earliest start schedules of the initially generated sequences from the MPM-bound. This method coincides with the serial scheme with priority rule $MinLFT$ developed by Kolisch (cf.

Decomposition	(15, 15, 15, 15)		(20, 20, 20)		(30, 30)	
	$\delta = 3$		$\delta = 3$		$\delta = 3$	
Precision	100,000		100,000		100,000	
	first 10	best 5	first 10	best 5	first 10	best 5
Avg.$\Delta(MPM)$ [%]	14.23	13.28	13.47	12.96	13.16	12.90
Avg.$\Delta(hrs)$ [%]	2.19	1.38	1.61	1.14	1.22	1.04
Avg. CPU [sec]	3.45	1.99	5.73	3.16	7.95	4.31
	first 50	best 15	first 50	best 15	first 50	best 15
Avg.$\Delta(MPM)$[%]	13.25	13.00	12.69	12.04	12.48	12.54
Avg.$\Delta(hrs)$[%]	1.41	1.18	0.97	0.92	0.71	0.78
Avg. CPU [sec]	14.20	4.65	25.86	8.05	26.66	11.58
	first 200	best 50	first 200	best 50	first 200	best 50
Avg.$\Delta(MPM)$[%]	12.67	12.80	12.20	12.38	12.11	12.54
Avg.$\Delta(hrs)$[%]	0.95	1.04	0.60	0.72	0.43	0.78
Avg. CPU [sec]	54.40	14.07	101.36	25.48	143.86	36.87

Table 7: J60 - Average Deviation and CPU-Time versus Partial Selection and Complete Selection

	Decomposition	($\delta = 3; 30, 30$)							
	Precision[nps]	200,000						5,000,000	
	No. of Gen. Seqs	200		400		1,000		1,000	
	$MPM(BestSeq)$[%]	14.77		14.42		14.04		14.04	
NrOfSeq		avg.	max.	avg.	max.	avg.	max.	avg.	max.
best 20	$\Delta(MPM)$[%]	12.29	118.18	12.27	115.58	12.07	114.29	11.61	109.09
	$\Delta(hrs)$ [%]	0.60	7.96	0.59	8.33	0.43	7.92	0.14	4.42
	CPU [sec]	27.32	166.20	27.31	160.71	29.14	166.54	460.16	4,311.46

Table 8: J60 - Average Deviation and CPU-Time versus Number of Initially Generated Sequences

[8]). We see: First, the larger the initial sample from which the best sequences are selected, the better are the results. The decomposition and concatenation algorithm substantially improves the makespans obtained through the serial scheme. Third, the last two columns show the fundamental influence of the precision. That is, the better the subproblems are solved the better the quality of the compound solution. Table 9 summarizes the results for instance set J90 and J120. The results for the analysis of the best 5, 10, 15, 20, 25 sequences can be found in Table 14 of the Appendix

Table 10 compares different heuristics for the single-mode resource-constrained project scheduling problem: The truncated version of the general sequencing algorithm GSA ([14]), the developed decomposition and concatenation algorithm (DACA), and the genetic algorithm presented by Hartmann (GA) ([6]). The genetic algorithm is currently the best heuristic available to solve the RCPSP (cf. [7]). The algorithms are compared on the instance set J30, J60, J90, and J120. The sets consist of 480 and 600 instances, respectively. The fourth and fifth column of the table display the average CPU-time $\overline{CPU}$ and the maximum CPU-time $Max(CPU)$. Note, for the GSA the maximum CPU-time is specified by the time allotted for solving a

		J90		J120	
		($\delta = 3; 30, 30, 30$)		($\delta = 3; 30, 30, 30, 30$)	
	Precision[nps]	500,000		500,000	
	No. of Gen. Seqs	1,000		1,000	
	$MPM(BestSeq)$[%]	14.13		43.13	
NrOfSeq		avg.	max.	avg.	max.
best 20	$\Delta(MPM)$[%]	12.45	118.64	39.35	214.14
	$\Delta(hrs)$ [%]	1.09	8.22	3.52	9.90
	CPU [sec]	110.16	673.14	369.38	1,209.36

Table 9: Average Deviation and CPU-Time for Instance Set J90 and J120

Set	NrOfInst	Algorithm	$\overline{CPU}$ [sec.]	$Max(CPU)$ [sec.]	$\overline{\Delta}$ [%]	$Max(\Delta)$ [%]
J30	480	GSA	1.42	60.00	0.02	4.69
		DACA, $(2, \delta = 3; 15, 15)$,	2.75	39.70	0.12	4.41
		200 of 200 seqs, 10,000 nodes				
		GA, 1000 It.	0.54	-	0.54	-
		GA, 5000 It.	2.70	-	0.25	-
J60	480	GSA	88.07	300.00	13.60	131.17
		GSA	472.69	1,800.00	13.04	128.57
		DACA, $(2, \delta = 3; 30, 30)$,	605.94	4,161.94	11.78	111.69
		200 of 200 seqs, 500,000 nodes				
		DACA, $(2, \delta = 3; 30, 30)$,	116.49	1,079.87	11.97	112.99
		best 5 of 1000 seqs, 5,000,000 nodes				
		DACA, $(2, \delta = 3; 30, 30)$,	460.16	4,311.46	11.61	109.09
		best 20 of 1000 seqs, 5,000,000 nodes				
		GA, 1000 It.	1.15	-	12.68	-
		GA, 5000 It.	5.75	-	11.89	-
J90	480	GSA	120.26	300.00	15.73	123.75
		DACA, $(3, \delta = 3; 30, 30, 30)$,	58.07	325.83	12.59	118.64
		best 10 of 1000 seqs, 500,000 nodes				
J120	600	GSA	497.03	600.00	44.46	242.42
		DACA , $(4, \delta = 3; 30, 30, 30, 30)$,	458.53	1,511.33	39.29	214.14
		best 25 of 1000 seqs, 500,000 nodes				
		GA, 1000 It.	3.04	-	39.37	-
		GA, 5000 It.	15.20	-	36.74	-

Table 10: Comparison of Different Algorithms

problem. The maximum CPU-time of the GA should be approximately equal to its average CPU-time. The GSA and the DACA have been executed on a Pentium with 166 MHz and the GA on a Pentium with 133 MHz. The sixth and seventh column display the average and maximum percentage deviation of the best makespan found from the optimal makespan (J30) or the MPM-bound (J60, J90, J120). On instance set J30

the GSA outperforms the DACA and the GA. The DACA produces a lower average deviation than the GA within nearly identical average CPU-time. On the instance sets J60, J90 and J120 the DACA determines in comparable average CPU-time solutions of higher quality than the GSA. The GA outperforms both the GSA and the DACA with respect to average CPU-time and average deviation.

6 Conclusion

We have presented a new heuristic solution strategy for the resource-constrained project scheduling problem. The strategy combines elements of exact and heuristic solution procedures. It relies on decomposition of a problem into subproblems, near optimal solution of the subproblems, and concatenation of the subproblem solutions. The algorithm significantly outperforms the truncated exact branch-and-bound algorithm on larger instances. The computational results have shown that the quality of the overall solutions depend on the size of the subproblems and the quality of the subproblem solutions. Consequently the algorithm will benefit from the progress made in the development of exact solution procedures.

Acknowledgments: I am grateful to Sönke Hartmann for helpful comments and suggestions. Moreover, I would like to thank Andreas Drexl for his continuous support.

References

[1] BAAR, T.; P. BRUCKER; S. KNUST (1998): Tabu-search algorithms and lower bounds for the resource-constrained project scheduling problem. In: ; S. Voss; S. Martello; I. Osman; C. Roucairol (Eds.): Meta-heuristics: Advances and Trends in Local Search Paradigms for Optimization, Kluwer, pp. 1-18.

[2] BRUCKER, P.; A. DREXL; R. MÖHRING; K. NEUMANN AND E. PESCH (1999): Resource-constrained project scheduling: Notation, classification, model, and methods. European Journal of Operational Research, Vol. 112, pp. 3-41.

[3] BRUCKER, P.; S. KNUST; A. SCHOO AND O. THIELE (1997): A branch & bound algorithm for the resource-constrained project scheduling problem. European Journal of Operational Research, Vol. 107, pp. 272-288.

[4] DEMEULEMEESTER, E. AND W. HERROELEN (1997): New benchmark results for the resource-constrained project scheduling problem. Management Science, Vol. 43, pp. 1485-1492

[5] GAREY, M.R. AND D.S. JOHNSON (1979): Computers and intractability: A Guide to the Theory of NP-Completeness. Freeman, San Francisco, CA.

[6] HARTMANN, S. (1998): A competitive genetic algorithm for resource-constrained project scheduling. Naval Research Logistics, Vol. 45, pp. 733 - 750.

[7] KOLISCH, R. AND S. HARTMANN (1998): Heuristic algorithms for solving the resource-constrained project scheduling problem: Classification and computational analysis. Manuskripte aus den Instituten für Betriebswirtschaftslehre, Research Report No. 469, Christian-Albrechts-Universität Kiel, Germany.

[8] KOLISCH, R. (1996): Serial and parallel resource-constrained project scheduling methods revisited: Theory and computation. European Journal of Operational Research, Vol. 90, pp. 320-333.

[9] KOLISCH, R. AND A. DREXL (1996): Adaptive search for solving hard project scheduling problems. Naval Research Logistics, Vol. 43, pp. 23-40.

[10] KOLISCH, R. AND A. SPRECHER (1996): PSPLIB – A project scheduling problem library. European Journal of Operational Research, Vol. 96, S. 205-216.

[11] KOLISCH, R.; A. SPRECHER AND A. DREXL (1995): Characterization and generation of a general class of resource-constrained project scheduling problems. Management Science, Vol. 41, pp. 1693-1703.

[12] MINGOZZI, A.; V. MANIEZZO; S. RICCIARDELLI AND L. BIANCO (1998): An exact algorithm for the resource-constrained project scheduling problem based on a new mathematical formulation. Management Science, Vol. 44, pp. 714-729.

[13] PATTERSON, J.H.; R. SLOWINSKI; F.B. TALBOT AND J. WEGLARZ (1989): An algorithm for a general class of precedence and resource constrained scheduling problems. In: Slowinski, R. and J. Weglarz (Eds.): Advances in project scheduling. Elsevier, Amsterdam, pp. 3-28.

[14] SPRECHER, A. (1997): Scheduling resource-constrained projects competitively at modest memory requirements. Manuskripte aus den Instituten für Betriebswirtschaftslehre, Research Report No. 449 (Revised Version), Christian-Albrechts-Universität Kiel, Germany.

[15] SPRECHER, A.; R. KOLISCH AND A. DREXL (1995): Semi-active, active and non-delay schedules for the resource-constrained project scheduling problem. European Journal of Operational Research, Vol. 80, pp. 94-102.

[16] STINSON, J.P.; E.W. DAVIS AND B.M. KHUMAWALA (1978): Multiple resource-constrained scheduling using branch and bound. AIIE Transactions, Vol. 10, pp. 252-259.

A Further Computational Results

	Decomposition	$(\delta = 3; 15, 15, 15, 15)$				$(\delta = 3; 20, 20, 20)$			
	Precision [nps]	25,000		100,000		25,000		100,000	
NrOfSeq		avg.	max	avg.	max.	avg.	max.	avg.	max.
10	$\Delta(MPM)$[%]	14.28	116.88	14.23	114.29	13.76	112.99	13.47	112.99
	$\Delta(hrs)$ [%]	2.22	18.18	2.19	18.18	1.79	12.73	1.61	12.73
	CPU [sec]	1.84	11.44	3.45	34.15	2.43	12.69	5.73	45.92
20	$\Delta(MPM)$[%]	13.79	114.29	13.72	111.69	13.35	112.99	13.11	112.99
	$\Delta(hrs)$ [%]	1.82	10.91	1.78	10.91	1.47	12.73	1.32	12.73
	CPU [sec]	2.95	20.96	6.18	68.03	4.15	24.20	10.81	90.80
50	$\Delta(MPM)$ [%]	13.35	112.99	13.25	111.69	12.92	112.99	12.69	112.99
	$\Delta(hrs)$ [%]	1.47	10.91	1.41	10.91	1.13	12.73	0.97	9.09
	CPU [sec]	6.30	50.45	14.20	168.31	9.28	58.79	25.86	225.60
100	$\Delta(MPM)$[%]	13.04	112.99	12.92	110.39	12.70	112.99	12.45	110.39
	$\Delta(hrs)$ [%]	1.23	10.91	1.15	10.91	0.95	9.09	0.79	9.09
	CPU [sec]	11.88	98.02	27.35	325.92	17.87	114.75	51.02	447.69
200	$\Delta(MPM)$ [%]	12.75	112.99	12.67	110.39	12.48	112.99	12.20	110.39
	$\Delta(hrs)$ [%]	1.00	9.09	0.95	9.09	0.78	7.27	0.60	7.27
	CPU [sec]	23.05	193.21	54.40	638.37	35.08	231.08	101.36	882.11
	NOS	758.71	800.00	777.57	800.00	526.14	600.00	548.43	600.00

Table 11: J60 – Average Deviation and CPU-Time versus Decomposition and Number of Decomposed Sequences

	Decomposition	$(\delta = 3; 30, 30)$							
	Precision [nps]	50,000		100,000		200,000		500,000	
NrOfSeq		avg.	max.	avg.	max.	avg.	max.	avg.	max.
10	$\Delta(MPM)$[%]	13.47	119.48	13.16	119.48	13.03	118.18	12.81	116.88
	$\Delta(hrs)$ [%]	1.46	11.11	1.22	11.11	1.16	11.11	1.02	11.11
	CPU [sec]	4.80	21.01	7.95	51.43	14.23	79.08	31.03	206.90
20	$\Delta(MPM)$[%]	13.04	119.48	12.81	119.48	12.69	118.18	12.48	115.58
	$\Delta(hrs)$ [%]	1.15	9.09	0.96	9.09	0.90	9.09	0.76	9.09
	CPU [sec]	8.89	41.90	15.13	97.29	27.67	162.12	61.43	420.40
50	$\Delta(MPM)$ [%]	12.74	116.88	12.48	115.58	12.32	112.99	12.13	111.69
	$\Delta(hrs)$ [%]	0.91	9.09	0.71	9.09	0.63	9.09	0.50	9.09
	CPU [sec]	21.08	104.00	36.66	216.33	68.04	407.55	151.82	1,046.17
100	$\Delta(MPM)$[%]	12.56	116.88	12.27	115.58	12.14	112.99	11.94	111.69
	$\Delta(hrs)$ [%]	0.78	8.331	0.55	8.331	0.49	7.96	0.36	7.96
	CPU [sec]	41.23	206.26	72.29	411.88	135.19	813.63	303.21	2,077.42
200	$\Delta(MPM)$ [%]	12.35	116.88	12.11	115.58	11.98	112.99	11.78	111.69
	$\Delta(hrs)$ [%]	0.61	7.96	0.43	7.96	0.37	7.96	0.23	6.19
	CPU [sec]	81.65	412.21	143.86	821.55	268.86	1,604.02	605.94	4,161.94
	NOS	312.47	400.00	321.64	400.00	329.29	400.00	338.43	400.00

Table 12: J60 - Average Deviation and CPU-Time versus Precision and Number of Decomposed Sequences

	Decomposition	($\delta = 3;30,30$)							
	Precision[nps]	200,000						5,000,000	
	No. of Gen. Seqs	200		400		1,000		1,000	
	$MPM(BestSeq)$[%]	14.77		14.42		14.04		14.04	
NrOfSeq		avg.	max.	avg.	max.	avg.	max.	avg.	max.
best 5	$\Delta(MPM)$[%]	12.75	119.48	12.66	118.18	12.41	114.29	11.97	112.99
	$\Delta(hrs)$ [%]	0.94	9.09	0.87	9.73	0.72	8.33	0.40	7.68
	CPU [sec]	7.28	43.10	7.68	41.89	9.84	45.98	116.49	1,079.87
best 10	$\Delta(MPM)$[%]	12.50	119.48	12.43	118.18	12.25	114.29	11.70	109.09
	$\Delta(hrs)$ [%]	0.76	9.09	0.70	9.73	0.56	7.92	0.22	6.19
	CPU [sec]	14.02	85.05	14.24	81.35	16.34	86.50	232.03	2,133.39
best 15	$\Delta(MPM)$ [%]	12.35	118.18	12.34	118.18	12.13	114.29	11.66	109.09
	$\Delta(hrs)$ [%]	0.65	9.09	0.63	8.33	0.47	7.92	0.19	6.19
	CPU [sec]	20.68	127.08	20.84	120.93	22.80	124.33	345.44	3,233.66
best 20	$\Delta(MPM)$[%]	12.29	118.18	12.27	115.58	12.07	114.29	11.61	109.09
	$\Delta(hrs)$ [%]	0.60	7.96	0.59	8.33	0.43	7.92	0.14	4.42
	CPU [sec]	27.32	166.20	27.31	160.71	29.14	166.54	460.16	4,311.46
best 50	$\Delta(MPM)$ [%]	12.12	112.99	12.04	112.99	11.92	112.99	–	–
	$\Delta(hrs)$ [%]	0.47	7.08	0.42	7.96	0.33	6.19	–	–
	CPU [sec]	67.26	415.75	66.65	410.34	68.12	400.92	–	–
	NOS	82.33	100.00	82.63	100.00	82.83	100.00	35.85	40.00

Table 13: J60 – Average Deviation and CPU-Time versus Selection of a Subset of Generated Sequences

		J90 ($\delta = 3;30,30,30$)		J120 ($\delta = 3;30,30,30,30$)	
	Precision[nps]	500,000		500,000	
	No. of Gen. Seqs	1,000		1,000	
	$MPM(BestSeq)$[%]	14.13		43.13	
NrOfSeq		avg.	max.	avg.	max.
best 5	$\Delta(MPM)$[%]	12.76	120.34	40.18	215.49
	$\Delta(hrs)$ [%]	1.32	9.59	4.07	13.39
	CPU [sec]	31.94	165.38	101.48	311.73
best 10	$\Delta(MPM)$[%]	12.59	118.64	39.65	215.15
	$\Delta(hrs)$ [%]	1.19	9.09	3.72	9.90
	CPU [sec]	58.07	325.83	191.12	610.12
best 15	$\Delta(MPM)$ [%]	12.49	118.64	39.46	215.15
	$\Delta(hrs)$ [%]	1.13	9.09	3.60	9.90
	CPU [sec]	84.22	487.85	279.37	899.56
best 20	$\Delta(MPM)$[%]	12.45	118.64	39.35	214.14
	$\Delta(hrs)$ [%]	1.09	8.22	3.52	9.90
	CPU [sec]	110.16	673.14	369.38	1,209.36
best 25	$\Delta(MPM)$[%]	12.40	118.64	39.29	214.14
	$\Delta(hrs)$ [%]	1.06	8.22	3.46	9.90
	CPU [sec]	136.46	853.57	458.53	1,511.33
	NOS	66.44	75.00	53.20	100.00

Table 14: Average Deviation and CPU-Time versus Selection of a Subset of Generated Sequences