DEPARTEMENT TOEGEPASTE ECONOMISCHE WETENSCHAPPEN

ONDERZOEKSRAPPORT NR 9628

COMPUTATIONAL EXPERIENCE WITH A BRANCH-AND-BOUND PROCEDURE FOR THE RESOURCE-CONSTRAINED PROJECT SCHEDULING PROBLEM WITH GENERALIZED PRECEDENCE RELATIONS

by

B. De Reyck

W. Herroelen

Katholieke Universiteit Leuven

Naamsestraat 69, B-3000 Leuven

ONDERZOEKSRAPPORT NR 9628

COMPUTATIONAL EXPERIENCE WITH A BRANCH-AND-BOUND PROCEDURE FOR THE RESOURCE-CONSTRAINED PROJECT SCHEDULING PROBLEM WITH GENERALIZED PRECEDENCE RELATIONS

by

B. De Reyck

W. Herroelen

D/1996/2376/28

COMPUTATIONAL EXPERIENCE WITH A BRANCH-AND-BOUND PROCEDURE FOR THE RESOURCE-CONSTRAINED PROJECT SCHEDULING PROBLEM WITH GENERALIZED PRECEDENCE RELATIONS

Bert DE REYCK

Willy HERROELEN

June 1996

Operations Management Group
Department of Applied Economics
Katholieke Universiteit Leuven
Hogenheuvel College
Naamsestraat 69, B-3000 Leuven, Belgium
Phone: 32-16-32 69 66 or 32-16-32 69 70
Fax: 32-16-32 67 32
E-mail: Bert.DeReyck@econ.kuleuven.ac.be or Willy.Herroelen@econ.kuleuven.ac.be
WWW-page: http://econ.kuleuven.ac.be/tew/academic/om/people/bert
http://econ.kuleuven.ac.be/tew/academic/om/people/willy

COMPUTATIONAL EXPERIENCE WITH A BRANCH-AND-BOUND PROCEDURE FOR THE RESOURCE-CONSTRAINED PROJECT SCHEDULING PROBLEM WITH GENERALIZED PRECEDENCE RELATIONS

Bert De Reyck • Willy Herroelen

Department of Applied Economics, Katholieke Universiteit Leuven

ABSTRACT

In a previous paper (De Reyck and Herroelen, 1996a), we presented an optimal procedure for the resource-constrained project scheduling problem (RCPSP) with generalized precedence relations (further denoted as RCPSP-GPR) with the objective of minimizing the project makespan. The RCPSP-GPR extends the RCPSP to arbitrary minimal and maximal time lags between the starting and completion times of activities. The procedure is a depth-first branch-and-bound algorithm in which the nodes in the search tree represent the original project network extended with extra precedence relations, which resolve a resource conflict present in the project network of the parent node. Resource conflicts are resolved using the concept of minimal delaying alternatives, i.e. minimal sets of activities which, when delayed, release enough resources to resolve the conflict. Precedence- and resource-based lower bounds as well as dominance rules are used to fathom large portions of the search tree. In this paper we report new computational experience with the algorithm using a new RCPSP-GPR random problem generator developed by Schwindt (1995). A comparison with other computational results reported in the literature is included.

1. Introduction

Assume a project represented in activity-on-node (AoN) notation by a directed graph $G = \{V, E\}$ in which V is the set of vertices or activities, and E is the set of edges or generalized precedence relations (GPRs). The non-preemptable activities are numbered from 1 to n, where the dummy activities 1 and n mark the beginning and the end of the project. The duration of an activity is given by $d_i (1 \le i \le n)$, its starting time by $s_i (1 \le i \le n)$ and its finishing time by $f_i (1 \le i \le n)$. There are m *renewable* resource types, with r_{ikx} $(1 \le i \le n, 1 \le k \le m, 1 \le x \le d_i)$ the resource requirements of activity i with respect to resource type k in the x^{th} period it is in progress and $a_{kt} (1 \le k \le m; 1 \le t \le T)$ the availability of resource type k in time period $]t\text{-}1, t]$ (T is an upper bound on the project length). If the resource requirements and availabilities are not time-dependent, they are represented by r_{ik} $(1 \le i \le n, 1 \le k \le m)$ and $a_k (1 \le k \le m)$ respectively. The minimal and maximal time lags between two activities i and j have the form:

$$s_i + SS_{ij}^{\min} \le s_j \le s_i + SS_{ij}^{\max}$$

$$s_i + SF_{ij}^{\min} \le f_j \le s_i + SF_{ij}^{\max}$$

$$f_i + FS_{ij}^{\min} \le s_j \le f_i + FS_{ij}^{\max}$$

$$f_i + FF_{ij}^{\min} \le f_j \le f_i + FF_{ij}^{\max}$$

The different types of GPRs can be represented in a *standardized form* by reducing them to just one type, e.g. the minimal start-start precedence relations, using the following transformation rules (Bartusch et al., 1988):

$$\begin{array}{lcll}
s_i + SS_{ij}^{\min} \le s_j & \Rightarrow & s_i + l_{ij} \le s_j & \text{with} \quad l_{ij} = SS_{ij}^{\min} \\
s_i + SS_{ij}^{\max} \ge s_j & \Rightarrow & s_j + l_{ji} \le s_i & \text{with} \quad l_{ji} = -SS_{ij}^{\max} \\
s_i + SF_{ij}^{\min} \le f_j & \Rightarrow & s_i + l_{ij} \le s_j & \text{with} \quad l_{ij} = SF_{ij}^{\min} - d_j \\
s_i + SF_{ij}^{\max} \ge f_j & \Rightarrow & s_j + l_{ji} \le s_i & \text{with} \quad l_{ji} = d_j - SF_{ij}^{\max} \\
f_i + FS_{ij}^{\min} \le s_j & \Rightarrow & s_i + l_{ij} \le s_j & \text{with} \quad l_{ij} = d_i + FS_{ij}^{\min} \\
f_i + FS_{ij}^{\max} \ge s_j & \Rightarrow & s_j + l_{ji} \le s_i & \text{with} \quad l_{ji} = -d_i - FS_{ij}^{\max} \\
f_i + FF_{ij}^{\min} \le f_j & \Rightarrow & s_i + l_{ij} \le s_j & \text{with} \quad l_{ij} = d_i - d_j + FF_{ij}^{\min} \\
f_i + FF_{ij}^{\max} \ge f_j & \Rightarrow & s_j + l_{ji} \le s_i & \text{with} \quad l_{ji} = d_j - d_i - FF_{ij}^{\max}
\end{array}$$

Then, the resource-constrained project scheduling problem with generalized precedence relations (RCPSP-GPR) can be conceptually formulated as follows:

$$\text{Minimize } s_n \qquad [1]$$

Subject to

$$s_i + l_{ij} \le s_j \qquad \forall (i,j) \in E \qquad [2]$$

$$\sum_{i \in S(t)} r_{ik} \le a_{kt} \qquad k = 1,2,\ldots,m \qquad t = 1,2,\ldots,T \qquad [3]$$

$$s_1 = 0 \qquad [4]$$

$$s_i \in \mathrm{N} \qquad i = 1,2,\ldots,n \qquad [5]$$

where $S(t)$ is the set of activities in progress in time period $]t\text{-}1, t]$ and T is an upper bound on the project duration, for instance $T = \sum_{i \in V} \max\left\{d_i, \max_{j \in Q(i)}\{l_{ij}\}\right\}$, $Q(i) = \{j \mid (i,j) \in E\}$ being the set of all immediate successors of activity i. Note that it is not always possible to derive a feasible solution for a RCPSP-GPR instance. The upper bound T indicates the maximal value for the project makespan if a feasible solution exists. The objective function given in Eq. 1 minimizes the project duration, given by the starting time (or finishing time, since $d_n = 0$) of the dummy activity n. The precedence constraints are denoted in standardized form by Eqs. 2. Eqs. 3 represent the resource constraints. The resource requirements and availabilities are assumed to be constant over time, although this assumption can be relaxed using GPRs without having to change the solution procedures (Bartusch et al., 1988). Eq. 4 forces the dummy start activity to begin at time zero and Eqs. 5 ensure that the activity starting times assume nonnegative integer values. Once started, activities run to completion (no preemption).

The RCPSP-GPR is known to be strongly NP-hard, and even the decision problem of testing whether a RCPSP-GPR instance has a feasible solution is NP-complete (Bartusch et al., 1988).

2. Solution procedures

The first optimal solution procedure for the RCPSP-GPR presented in the literature is the procedure of Bartusch et al. (1988). The procedure is a depth-first type branch-and-bound procedure which is based on the concept of a *forbidden set*, i.e. a set of activities which may never be scheduled in parallel because a violation of the resource constraints would result. Such a set is called *minimal* if no subset of that set constitutes a forbidden set in itself. Moreover, a minimal forbidden set is labelled a *reduced* forbidden set if the activities belonging to that set can be scheduled in parallel without violating the (generalized) precedence constraints between them. The procedure starts with the earliest start schedule and consequently adds new precedence relations between activities in order to eliminate reduced forbidden sets until no such set is scheduled in parallel.

In De Reyck and Herroelen (1996a), we presented a new depth-first branch-and-bound procedure for the RCPSP-GPR based on the concept of *minimal delaying alternatives*, i.e. sets of activities, which, when delayed release enough resources to resolve a resource conflict. Similar to Bartusch et al. (1988), we start with the earliest start schedule, in which we determine the first resource conflict, caused by a set of activities denoted as the *conflict set*. If no such conflict exists, then the earliest start schedule constitutes the optimal solution. If, however, such a conflict exists, we determine a set of minimal delaying alternatives and minimal delaying modes (a minimal delaying activity being delayed by another activity also belonging to the conflict set). Each minimal delaying mode then gives rise to a new earliest start schedule to which we can apply the same reasoning until a feasible solution is obtained. The procedure then backtracks to a previous level in the search tree until the root node is reached. The procedure also backtracks to a previous level when no undominated nodes remain at a certain level. Several dominance rules and lower bounds are introduced which fathom large portions of the search tree.

Apart from these optimal solution procedures for the RCPSP-GPR, several heuristics have been presented. Neumann and Zhan (1996) developed a priority-rule-based heuristic which allows to solve RCPSP-GPR instances using a parallel search scheme (see also Zhan, 1994). Brinkmann and Neumann (1994) developed a serial heuristic for the RCPSP-GPR (called DIRECT) and a heuristic based on the (serial) scheduling of cycle structures and the subsequent (serial) scheduling of the (acyclic) contracted project network (called CONTRACT).

Franck and Neumann (1996) improved the approach of Neumann and Zhan (1996) and validated the performance of the heuristics described above. They conclude that the DIRECT method performs significantly better than the CONTRACT method, although the required computation time is higher due to the necessity of rescheduling steps caused by time-infeasibilities. No conclusion could be made whether the serial or parallel scheme was more efficient for the DIRECT method. For the contract method, the most effective approach is to use the parallel scheme for scheduling the cycle structures and the serial scheme for scheduling the (acyclic) contracted project network. The best performing priority rules are the latest start time (*LST*) heuristic, in which priorities are assigned to schedulable (eligible) activities in the order of nondecreasing latest start times, and the worst case slack (*WCS*) heuristic, which assigns priorities in the order of nondecreasing worst case slack, where the *WCS* of an activity i with respect to another activity j is defined as the difference between the latest start time of activity i and the earliest possible time instant at which activity i can be scheduled if another eligible activity j is scheduled instead. The *WCS* of an eligible activity is then defined as the minimal value of all *WCS* values with respect to all other eligible activities.

Schwindt and Neumann (1996) developed a branch-and-bound-based heuristic based on the optimal procedure for the RCPSP-GPR of De Reyck and Herroelen (1996a) and the optimal procedure of Demeulemeester and Herroelen (1992, 1995) for the RCPSP. First, the cycle structures are scheduled using the procedure of De Reyck and Herroelen (1996a), after which they are contracted into single activities, which allows the use of a modified version (including variable resource requirements) of the procedure of Demeulemeester and Herroelen (1992, 1995) to schedule the contracted (acyclic) project network as an RCPSP instance.

3. Benchmark problem sets

In this paper, we will perform extensive tests to determine the effectiveness and efficiency of our procedure for the RCPSP-GPR (De Reyck and Herroelen, 1996a), in relation to several complexity measures which have been developed for the RCPSP-GPR and to other procedures described in the literature.

The procedure has been programmed in Microsoft® Visual C++ 2.0 under Windows NT for use on a Digital Venturis Pentium-60 personal computer with 16Mb of internal memory. The code itself requires 109Kb of memory, whereas 10Mb are reserved for the storage of the search tree. In order to validate our branch-and-bound procedure, we generated 550 RCPSP-GPR instances based on the problem set for the RCPSP assembled by Patterson (1984). The results indicate the high efficiency of the branching scheme based on minimal delaying alternatives relative to a complete enumeration and show the significant speedup (up to a factor of 1410 for the problem instances with 20% maximal time lags) realized by adding dominance rules and lower bounds, and show the potential of the procedure to solve problems up to 30 activities within very small computation times. More information on this problem set and on the heuristic capabilities of a truncated version of our procedure can be found in De Reyck and Herroelen (1996a).

Schwindt (1995) developed a random problem generator ProGen/max for the RCPSP-GPR based on the problem generator ProGen for the RCPSP developed by Kolisch et al. (1995). ProGen/max can generate RCPSP instances, multiple-mode RCPSP (MRCPSP) instances, RCPSP-GPR instances as well as MRCPSP-GPR (a combination of multiple modes and generalized precedence relations) instances. In addition, instances of the resource levelling problem with generalized precedence relations (RLP-GPR) and the resource availability cost problem with generalized precedence relations (RACP-GPR) can be generated. Two methods are proposed: DIRECT, which directly generates entire projects, and CONTRACT, which first generates cycle structures, upon which the (acyclic) contracted project network is generated. Several control parameters can be specified, as indicated in Table I.

Table I. The control parameters of ProGen/max (Schwindt, 1995)

Problem size-based	Resource-based	Acyclic network-based	Cyclic network-based
# activities (n)	# resource types (m)	# initial and terminal activities	% maximal time lags
	min. / max. number of resources used per activity	maximal # predecessors and successors	# cycle structures
	resource factor (RF) (Pascoe, 1966)	order strength (OS)[1] (Mastor, 1970)	min. / max. # nodes per cycle structure
	resource strength (RS) (Kolisch et al., 1995)		coefficient of cycle structure density (Schwindt, 1995)
			cycle structure tightness (Schwindt, 1995)

Two RCPSP-GPR problem sets have already been generated using ProGen/max. The first set (Schwindt, 1996) consists of 1080 instances, of which 540 are generated using the DIRECT method and 540 using the CONTRACT method. The second set (Franck and Neumann, 1996) consists of 1440 problem instances generated using the DIRECT method. We used the DIRECT method to generate a new set consisting of 7200 problem instances, which allows for a more extensive testing of the impact of several control parameters. We will use these three benchmark sets to test the effectiveness and efficiency of our branch-and-bound procedure, and validate it, where possible, with other results in the literature.

4. The problem set of Schwindt (1996) - 1080 instances

4.1. Description of the problem set

The problem set of Schwindt (1996) has been generated using the parameter settings described in Table II. The indication $[x,y]$ means that the value is randomly generated in the interval $[x,y]$, whereas x; y; z means that three settings for that parameter were used in a full factorial experiment. For each combination of control parameter values, 10 problem instances have been generated.

[1] Schwindt (1996) uses an estimator for the restrictiveness (Thesen, 1977) as a network complexity measure. However, De Reyck (1995) has shown that this measure is identical to the order strength (Mastor, 1970), the flexibility ratio (Dar-El, 1973) and the density (Kao and Queyranne, 1982). We will use *order strength* when referring to this measure.

Table II. The parameter settings of the problem set of Schwindt (1996)

Control parameter	Value
# activities	100
activity durations	[5,15]
# resource types	5
minimal / maximal # resources used per activity	1 / 5
activity resource demand	[1,3]
RF	0.50; 0.75; 1.00
RS	0.20; 0.50; 0.70
# initial and terminal activities	[3,7]
max. # initial / terminal activities for the cycle structures (only for CONTRACT method)	2 / 2
max. # predecessors / successors	5 / 5
max. # predecessors / successors for the cycle structures (only for CONTRACT method)	3 / 3
OS	0.35; 0.50; 0.65
OS for the cycle structures (only for CONTRACT method)	0.50
% maximal time lags	[5%,15%]
# cycle structures	[2,5]; [6,9]
minimal / maximal # nodes per cycle structure	2 / 15
coefficient of cycle structure density	0.3
cycle structure tightness	0.5

4.2. The results obtained by Franck and Neumann (1996)

Franck and Neumann (1996) have tested their improved versions of the serial and parallel heuristics for the DIRECT and CONTRACT approach on this problem set. The results given in Table III include the number of problems solved to optimality, the number of unsolved problems, the average deviation from a lower bound and the average and maximal deviation from the best known solution. A problem is considered to be unsolved when no feasible solution has been found. Two deviations with respect to a lower bound are given. The first lower bound (lb_1), used by Franck and Neumann (1996) to report deviations, is the maximum of a critical path-based lower bound and a resource-based lower bound (based on dividing the activity duration-resource requirement products by the resource availability). The second lower bound (lb_2) is the maximum of lb_1 and the lower bound lb_3 for the RCPSP-GPR (developed by De Reyck and Herroelen, 1996a) computed in the root node of the search tree after preprocessing. The values for

lb_1 as reported by Schwindt (1996) for this problem set contained some errors, so that we recalculated lb_1 (and the deviations reported by Schwindt, 1996). The best known solution referred to in Table III is the best solution obtained with various versions of our branch-and-bound algorithm running for 1 hour per problem and the heuristic solutions, and can therefore be considered as near-optimal. Note that, when it is indicated that the heuristic finds an optimal solution, this means that the obtained solution is equal to the lower bound.

The heuristics are coded in Smalltalk® for use on a personal computer (Franck and Neumann, 1996). The total computation time for the heuristics to produce the results given in Table III is not known exactly. Franck and Neumann (1996) report that about three seconds on a Pentium-90 PC are needed to solve one heuristic procedure based on the CONTRACT technique. Since in total 44 different CONTRACT-based heuristics are used, this accounts for about 132 seconds. However, also 11 DIRECT-based heuristics are used, which consume much more computation time due to a possibly large number of rescheduling steps (Franck and Neumann, 1996). Therefore, the total time needed to produce the results given in Table III is not known exactly, but runs in the hundreds of seconds.

Table III. The results of Franck and Neumann (1996)

	DIRECT	CONTRACT
Problems solved to optimality	136 (>25%)	60 (>11%)
Unsolved problems	8 (<2%)	13 (<3%)
Average deviation from lb_1	14.65%	20.59%
Average deviation from lb_2	13.51%	20.53%
Average deviation from best known solution	5.32%	8.37%
Maximal deviation from best known solution	59.85%	91.70%

Notice, on the one hand, the rather high average (and maximal) deviations from the near-optimal solutions, but, on the other hand, the small number of unsolved problems. Remember that these problems may not have a feasible solution at all. A heuristic procedure is, however, not capable of determining whether an RCPSP-GPR instance is infeasible.

4.3. Truncated branch-and-bound

Solving all 100 activity-problem instances of the Schwindt (1995) problem set to optimality is probably beyond the capabilities of current branch-and-bound procedures. Even for the classic RCPSP, problem instances with 100 activities are not amenable to optimal solution

within acceptable computational effort. As an example, the RCPSP problem set of Kolisch et al. (1995), which consists of 480 instances with 30 activities has only recently been solved to optimality by Demeulemeester and Herroelen (1995). Therefore, given the higher complexity of the RCPSP-GPR, it is to be expected that a similar set consisting of 30-activity RCPSP-GPR instances will not be solved to optimality within acceptable computation times. Computational experience with a truncated branch-and-bound algorithm on 100-activity problem instances have not yet been reported in the literature, not even for the classic RCPSP case.

Inspired by the excellent results obtained by Demeulemeester and Herroelen (1995) with a truncated branch-and-bound procedure for the RCPSP, we report in Table IV the results of our procedure, when truncated after some seconds of running time. As was the case for the results given in Table III, the deviations are calculated only for the problem instances for which a feasible solution has been found. This leads to a deflation of the reported deviations when a large number of instances have not yet reached feasibility. The procedure is truncated if the elapsed computation time exceeds the given limit. However, to avoid excessive time checking (a very time-consuming procedure in itself), we only check the elapsed time when the procedure backtracks to a previous level. Although this reduces the number of time checks substantially, the procedure just barely runs over the given time limit for some problem instances. However, there are some exceptions. If a time limit of 1 or 2 seconds is set, it often happens that the procedure is still going down the search tree when the time limit is reached, and therefore, has not yet backtracked to a previous level in the search tree. Therefore, it sometimes happens that the given time limit is exceeded. This, of course, is due to the fact that a search time of 1 second (or 2) for such large problem instances is very small. For each class of results, we therefore also report an average computation time. This average time will generally be much smaller than the given time limit, except for the case where the time limit is 1 or 2 seconds, in which case the time limit can be exceeded, leading to a higher average computation time. In general, however, the computation times do not heavily exceed the time limit of 1 second (2 seconds) very much, as can be seen from the average computation times.

For some problem instances, the procedure runs out of addressable memory, which results in a truncation of the algorithm even if the time limit is not exceeded. The fact that sometimes, the memory requirements exceed the available memory is due to the limits that we have set on the size of the search tree, in the form of a maximal number of levels in the search tree and a maximal number of delaying modes per level. Actually, when such a memory overflow occurs, it does not mean that more than 10Mb of storage have been used, but rather that one of these limits has been exceeded.

Table IV. The results with a truncated version of our branch-and-bound procedure

	1 second		2 seconds		10 seconds		100 seconds	
	DIRECT	CONTRACT	DIRECT	CONTRACT	DIRECT	CONTRACT	DIRECT	CONTRACT
Problems solved to optimality	260 (>48%)	283 (>52%)	287 (>53%)	290 (>53%)	293 (>54%)	298 (>55%)	303 (>56%)	303 (>56%)
Unsolved problems	97 (<18%)	108 (20%)	56 (<11%)	81 (15%)	32 (<6%)	55 (<11%)	26 (<5%)	45 (<9%)
Average deviation from lb_1	5.66%	7.09%	7.92%	9.14%	9.48%	11.51%	9.57%	12.04%
Average deviation from lb_2	5.23%	6.76%	7.17%	8.67%	8.60%	10.93%	8.56%	11.44%
Average deviation from best known solution	1.54%	2.29%	1.67%	2.48%	1.53%	2.86%	1.24%	2.66%
Maximum deviation from best known solution	38.11%	48.67%	37.74%	48.67%	22.07%	72.02%	22.07%	72.02%
Average computation time (seconds)	1.08	1.02	1.37	1.37	5.07	4.94	45.29	44.60

One remarkable conclusion we can draw from Table IV is that, despite the problem size, more than 50% (543 out of 1080) of the problems can be solved to optimality with an average of somewhat more than 1 second of computation time (16 problems are proven to be infeasible). It should be observed that this set does not contain any 'easy' instances in contrast to the RCPSP problem set with 480 instances of Kolisch et al. (1995), which contains 120 (25%) instances with a resource strength (*RS*) of 1, meaning that the problems are not resource-constrained and therefore can be solved by simply calculating the earliest start schedule. Solving such problem instances using our branch-and-bound procedure would require no branching at all.

Therefore, we can conclude that it is often advantageous to try and solve such problems, despite their size and complexity, using truncated optimal solution procedures before resorting to other types of heuristic procedures. Note that Franck and Neumann report that about 18% (199) problems were solved to optimality using the set of heuristics (a solution equal to the lower bound), at the price, however, of very large computation times.

The average deviations from the lower bounds and the best known solutions are very promising. For instance, the average deviation from the best solution known, which we can regard as being near-optimal since many different procedures and a lot of CPU time were used to obtain these solutions, varies between 1% and 2% for the DIRECT set and between 2% and 3% for the CONTRACT set. The heuristics resulted in an average deviation of 5.32% for the DIRECT set and 8.37% for the CONTRACT set (see Table III). Note, however, that these deviations are only computed for the instances for which a feasible solution has been found. This explains the increased deviations when more CPU time is allotted.

In conclusion, we can say that the number of problem instances solved to optimality and the deviations from the optimum are very promising. Less reassuring, however, is that, especially for small time limits, a relatively large number of problems remains unsolved. The set of heuristics does a better job on this issue. This inspired us to another approach (cf. infra) which is based on finding a feasible solution first, rather than going immediately for the optimal solution. This will allow us to find feasible solutions much more quickly at the expense of the quality of the solutions found when a certain time limit is imposed.

4.4. Truncated branch-and-bound using the time window slack branching scheme

In our branch-and-bound algorithm (De Reyck and Herroelen, 1996a), nodes are branched from in nondecreasing order of a critical path-based lower bound. The rationale behind this (often applied) branching criterion is that the nodes which entail a high chance of finding a very good solution are chosen first, in the hope that the other nodes will be dominated by the obtained upper

bound. However, when solving the RCPSP-GPR, two criteria, which may be in conflict, should be examined simultaneously. Each node in the search tree does not only contain information on the effect of added precedence constraints on the best solution that can ever be obtained by branching from that node (indicated by the lower bound), but also on the effect of added precedence constraints on the probability that a *feasible* solution can be obtained by branching from that node. The branching scheme described below also incorporates the latter information.

In the branch-and-bound procedure of De Reyck and Herroelen (1996a), a node (with a corresponding delaying activity k and delaying alternative D_d) is eliminated (because it does not contain a feasible solution) if $\exists l \in D_d : k \prec l$ is infeasible, i.e. if $d_k > -d[l][k]$ for some $l \in D_d$, where $d[l][k]$ denotes the maximal distance between activities l and k. Thus, if $d_k + d[l][k] \leq 0$ for each $l \in D_d$ (no positive cycle in the project network), the delaying mode is considered for further branching, and the selection of the delaying mode to branch from is derived from the lower bound.

Consider two delaying modes M_1 and M_2, each with one activity in the corresponding delaying alternative, with $d_{k_1} + d[l_1][k_1] = -1$ for M_1 and $d_{k_2} + d[l_2][k_2] = -20$ for M_2. Even if the lower bound of M_1 is smaller than the lower bound of M_2, branching from M_2 may be the smartest thing to do since there is a high probability that branching from M_1 will not lead to any feasible solution. The fact that $d_k + d[l][k] = -1$ means that activity k, which was delayed by activity l, only has 1 time unit of slack within its time window with respect to activity l. Thus, when activity k has to be delayed later on in the project, a positive cycle will probably result, leading to time-infeasibility of the corresponding project network. Therefore, if we want to find a feasible solution, it may be better to branch from the node for which the delayed activities have a relatively high 'slack' in the time windows in which they can be scheduled. This leads to our new branching strategy, namely branching from the node with the highest 'slack' with respect to the maximal time lags, i.e. in which the cycles created by delaying activities, if any, are as negative as possible. This slack value will be referred to as *time window slack* (*TWS*).

If multiple activities are delayed, the minimal *TWS* value over all the delayed activities is used as the 'slack' of the node, since this is probably the cycle that is going to create feasibility problems if additional activities are to be delayed. Suppose, for instance, that there are two possible delaying modes M_1 and M_2 to branch from. M_1 consists of delaying activities 2 and 3 by activity 1, and M_2 consists of delaying activities 1 and 3 by activity 2. If for M_1, $d_1 + d[1][2] = -4$

Table V. The results with the *TWS* version of our branch-and-bound procedure

	1 second		2 seconds		10 seconds		100 seconds	
	DIRECT	CONTRACT	DIRECT	CONTRACT	DIRECT	CONTRACT	DIRECT	CONTRACT
Problems solved to optimality	261 (>48%)	276 (>51%)	289 (>53%)	294 (>54%)	294 (>54%)	303 (>56%)	302 (>55%)	306 (>56%)
Unsolved problems	12 (<3%)	15 (<3%)	8 (<2%)	7 (<2%)	5 (<1%)	5 (<1%)	4 (<1%)	5 (<1%)
Average deviation from lb_1	12.77%	17.51%	12.57%	18.18%	12.20%	17.35%	11.72%	16.83%
Average deviation from lb_2	11.67%	16.77%	11.43%	16.77%	11.02%	16.60%	10.59%	16.09%
Average deviation from best known solution	3.26%	4.99%	2.82%	4.92%	2.49%	4.06%	2.11%	3.67%
Maximum deviation from best known solution	38.11%	51.17%	37.84%	59.89%	26.99%	52.85%	26.99%	50.41%
Average computation time (seconds)	1.16	1.24	1.38	1.38	5.02	4.86	44.99	43.42

and $d_1 + d[1][3] = -10$, and for M_2, $d_2 + d[2][1] = -6$ and $d_2 + d[2][3] = -12$, the values 4 and 6 are chosen as the respective *TWS* values. Therefore, if we want to find a feasible solution, branching from M_2 (with the highest time window slack) is preferred.

We used this approach in a new version of our branch-and-bound algorithm (further denoted as the *TWS-approach*, in contrast to the *LB-approach*). When no feasible solution has been found yet, the procedure branches from the node with the highest *TWS* value with respect to the maximal time lags. Upon finding a feasible solution, the branching criterion switches to the lower bound criterion as before. The results with this approach are given in Table V.

The number of unsolved problems decreases dramatically, even for small computation times. Even with a time limit of 2 seconds (1.38 seconds on average), less unsolved problems remain than found by Franck and Neumann (1996). However, as was to be expected, the deviations from the lower bounds and the best known solutions increase. Part of this increase in reported deviations, however, is due to the fact that now much less unsolved problem instances remain, which results in less deflation in the reported deviations. The deviations are still much smaller than when using the entire set of heuristic procedures, as is represented in Fig. 1.

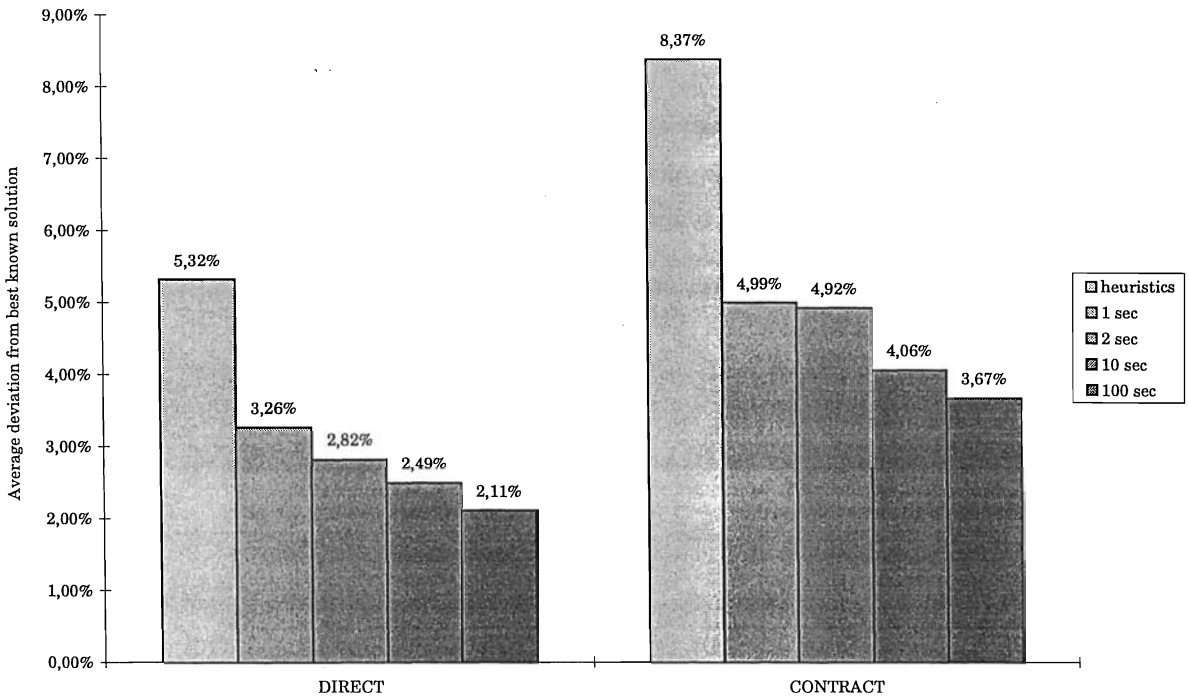

Fig. 1. Average deviations from best known solutions: *TWS-approach* vs. heuristics

5. The problem set of Franck and Neumann (1996) - 1440 instances

The problem set of Franck and Neumann (1996) has been generated using the DIRECT method of ProGen/max using the parameter settings described in Table VI. For each combination of control parameter values, 10 problem instances have been generated.

Table VI. The parameter settings of the problem set of Franck and Neumann (1996)

Control parameter	Value
# activities	100
activity durations	[5,15]
# resource types	[5,8]
minimal / maximal # resources used per activity	1 / 8
activity resource demand	[1,3]
RF	0.25; 0.50; 0.75; 1.00
RS	0.20; 0.50; 0.75
# initial and terminal activities	[3,7]
maximal # predecessors / successors	5 / 5
OS	0.35; 0.50; 0.65
% maximal time lags	[5%,15%]; [15%,25%]
# cycle structures	[2,7]; [8,13]
minimum / maximal # nodes per cycle structure	2 / 15
coefficient of cycle structure density	0.3
cycle structure tightness	0.5

Franck and Neumann (1996) report that, using the CONTRACT approach, a feasible solution was obtained for 1401 of the 1440 problem instances (1333 with the serial search scheme for the cycle structures and 1395 with the parallel search scheme for the cycle structures). The authors advise to use the *serial* search scheme for the (acyclic) contracted project network and the *parallel* scheme for the cycle structures (*SP scheme*). The best performing priority rule is the latest start time (*LST*) rule. The average deviation from lb_1 obtained with these rules equals 16.6% (± 3 seconds of CPU time on a Pentium-90 PC). The average deviation from lb_1 obtained with a multiple-pass approach consisting of all 11 examined priority rules and all four search schemes (SS, PS, SP and PP) equals 14.3% (± 132 seconds of CPU time). The authors state that better results can be obtained with the DIRECT approach, however, at the expense of increased

Table VII. The results with the truncated version of our branch-and-bound procedure

	1 second		2 seconds		10 seconds		100 seconds	
	LB approach	*TWS* approach	*LB* approach	*TWS* approach	*LB* approach	*TWS* approach	*LB* approach	*TWS* approach
Problems solved to optimality	766 (>53%)	766 (>53%)	869 (>60%)	870 (>60%)	891 (>61%)	895 (>62%)	910 (>63%)	916 (>63%)
Unsolved problems	267 (<19%)	39 (<3%)	149 (<11%)	23 (<2%)	66 (<5%)	16 (<2%)	43 (<3%)	16 (<2%)
Average deviation from lb_1	5.12%	12.36%	7.70%	12.28%	9.94%	11.60%	10.04%	11.14%
Average deviation from lb_2	4.61%	10.91%	6.83%	10.69%	8.52%	9.99%	8.50%	9.54%
Average deviation from best known solution	0.93%	2.38%	1.07%	2.04%	0.82%	1.36%	0.51%	1.01%
Maximum deviation from best known solution	24.09%	44.67%	26.55%	38.33%	20.00%	28.27%	14.44%	22.18%
Average computation time (seconds)	1.02	1.10	1.29	1.31	4.35	4.32	37.64	37.45

computation times due to a possibly large amount of rescheduling steps needed to resolve time-infeasibilities. Table VII indicates the results obtained with the truncated version of our branch-and-bound procedure, both for the *LB-approach* and the *TWS-approach*. The so-called best known solutions used to calculate the average and maximal deviations are obtained using the *LB*-approach and the *TWS*-approach running for 1 hour each.

Again, with a time limit of only 1 second (resulting in an average computation time of just over 1 second), more than 50% (766 out of 1440) of the problem instances can be solved to optimality. The deviations from lb_1 remain under 10% for the *LB*-approach and under 12.5% for the *TWS*-approach. Remember that the average deviations reported by Franck and Neumann (1996) were 16.6% for the best performing heuristic and 14.3% for the entire set of heuristics. With, on the average, only 1 second of computation time, the *TWS*-approach can solve as many problems (namely 1441; including the 12 problems proven to be infeasible) as do the entire set of heuristic procedures presented by Franck and Neumann (1996). Although the average deviations are significantly better for the *LB*-approach, the number of unsolved problems remains rather high (except for a time limit of 100 seconds).

6. A new problem set - 7200 instances

In order to examine the impact of several types of problem characteristics on the complexity of the RCPSP-GPR, we generated a new problem set consisting of 7200 instances using the DIRECT method of ProGen/max using the parameter settings described in Table VIII. For each combination of control parameter values, 10 problem instances have been generated.

The results with the truncated version of our branch-and-bound procedure using the *LB* and *TWS*-approach are given in Table IX. The best known solutions used to calculate the average and maximal deviations are obtained using the *LB*-approach and the *TWS*-approach running for 100 seconds. The effects of the various control parameters are represented in Fig. 2 through 23.

Table VIII. The parameter settings of the new problem set

Control parameter	Value
# activities	10; 20; 30; 50; 100
activity durations	[2,10]
# resource types	[1,5]
minimal / maximal # resources used per activity	1 / 5
activity resource demand	[1,10]
RF	0.25; 0.50; 0.75; 1.00
RS	0.25; 0.50; 0.75
# initial and terminal activities	[2,4]
maximal # predecessors and successors	3
OS	0.25; 0.50; 0.75
% maximal time lags	0%; 10%; 20%; 30%
# cycle structures	[0,10]
minimal / maximal # nodes per cycle structure	2 / 100
coefficient of cycle structure density	0.3
cycle structure tightness	0.5

From Table IX, we can observe that over 77% of the problems can be solved to optimality within 1 second of computation time. If 100 seconds of CPU time are allowed, this percentage increases to 86%. However, as Fig. 2 clearly displays, the number of problems solved to optimality heavily depends on the problem size. For 1 second of computation time, the percentage of problems solved to optimality decreases from 100% for the 10-activity problem instances to 58% for the 100-activity problem instances. Nevertheless, the relatively high number of problems solved to optimality (even for the 100-activity set) seems very promising, and indicates that, even for large problem instances, the use of (truncated) branch-and-bound procedures should not be discarded. Using the *LB*-approach, a relatively high number of problem instances remains unsolved, especially for the problem instances with 100 activities, as can be seen from Fig. 3. The *TWS*-approach, however, can solve all but ten instances (see Fig. 4).

Fig. 5 displays the average deviation from lb_2 obtained with the *LB*-approach. The counterintuitive effect, namely a decreasing deviation from the lower bound when the problem size increases, is probably due to two main reasons. First, the quality of lb_2 (measured by the average deviation from the optimal solution) increases when the problem size goes up. For

Table IX. The results with the truncated version of our branch-and-bound procedure

	1 second		2 seconds		10 seconds		100 seconds	
	LB approach	*TWS* approach	*LB* approach	*TWS* approach	*LB* approach	*TWS* approach	*LB* approach	*TWS* approach
Problems solved to optimality	5602 (>77%)	5575 (>77%)	5832 (81%)	5828 (>80%)	6017 (>83%)	6020 (>83%)	6210 (>86%)	6215 (>86%)
Unsolved problems	141 (<2%)	15 (<1%)	95 (<2%)	12 (<1%)	66 (<1%)	11 (<1%)	55 (<1%)	10 (<1%)
Average deviation from lb_1	8.08%	8.60%	8.06%	8.40%	7.95%	8.16%	7.77%	7.94%
Average deviation from lb_2	4.69%	5.40%	4.67%	5.20%	4.57%	4.97%	4.39%	4.75%
Average deviation from best known solution	0.44%	0.79%	0.38%	0.61%	0.29%	0.39%	0.06%	0.20%
Maximum deviation from best known solution	50.77%	46.21%	50.77%	42.66%	36.73%	40.27%	14.05%	31.06%
Average computation time (seconds)	0.36	0.39	0.54	0.54	1.92	1.92	14.93	14.87

instance, for the 10-activity set, the average deviation of the solutions obtained with our procedure from lb_2 is 9.13%, whereas all problems were solved to optimality (an actual average deviation from the optimum of 0%). Therefore, we can conclude that lb_2 itself is, on average, 9.13% from the optimum. When the problem size increases, the average deviation of lb_2 from the optimum decreases, leading to the lower average deviations from lb_2 obtained by our procedure. Second, with the *LB*-approach, the number of unsolved problems increases when the problem size goes up. Therefore, these problems cannot be included in the calculations, leading to a deflation in reported average deviations. This will not be the case for the *TWS*-approach, for which a similar graph is shown in Fig. 6. From Fig. 6 we can observe a similar effect when the problem size increases from 10 to 50 activities, beyond which the average deviations increase again. Nevertheless, it is more interesting to look at the average deviations from the best known solutions instead of the deviations with respect to a lower bound, since then, no such counterintuitive effects will occur. However, because of the size of the problem set, we were not able to run several versions of our procedure for 1 hour, as we did for the previous problem sets. Therefore, we took as the baseline the best results obtained with the *LB*-approach and the *TWS*-approach with a time limit of 100 seconds each. The average deviations with respect to these (best known) solutions are given in Fig. 7. Clearly, the effect of the problem size on the average deviation from the best known solution is not counterintuitive.

Fig. 8 through 11 represent the effect of the *OS* on the RCPSP-GPR complexity. De Reyck (1995) has shown that *OS* is a relatively good measure of network complexity for the RCPSP, in that it explains a lot of the variation in the required computation time of optimal branch-and-bound procedures to solve RCPSP instances, next to problem size and resource-based complexity measures. More specifically, *OS* has a negative impact on the computational complexity of the RCPSP, implying the higher *OS*, the easier the corresponding RCPSP. Schwindt and Neumann (1996) use *OS* as a network-based parameter for randomly generating (M)RCPSP(-GPR), (M)RLP(-GPR) or (M)RACP(-GPR) instances. In this respect, they differ from ProGen (Kolisch et al., 1995) which generates (M)RCPSP instances using *CNC* (arcs / nodes) as a network complexity measure. De Reyck (1995) and De Reyck and Herroelen (1996b) have shown that *CNC* does not perform very well as a network complexity measure for the RCPSP. The complexity index *CI* (De Reyck and Herroelen, 1996b) and the order strength *OS* can explain a much higher portion of the variability in CPU times needed to solve RCPSP instances. Fig. 8 through 11 also indicate a negative impact of *OS* on the computational complexity of the RCPSP-GPR. When *OS* increases, the number of problems solved to optimality generally increases, the number of unsolved problems decreases and the average deviations from the best known solutions also decrease.

The effect of the percentage of maximal time lags (see Fig. 12 through Fig. 15) on the computational complexity of the RCPSP-GPR is not monotonously increasing or decreasing. On the contrary, a kind of bell-shaped curve seems to result. When maximal time lags are introduced, the number of problems solved to optimality increases (Fig. 12), up to a certain point, at which the number of problems solved to optimality again decreases. The initial rise in performance can easily be explained if we remember that, in the branch-and-bound procedure, several dominance rules and lower bounds are used which require the existence of maximal time lags in order to be applicable. This makes the procedure more effective and efficient when such time lags are introduced. However, when many maximal time lags are introduced, the increased problem complexity (there are less feasible solutions, making it harder to find good ones which can be used to dominate other nodes using lower bound arguments) leads to a decrease in efficiency and consequently, a decrease in the number of problems solved to optimality within the given time limit. The non-linear effect of the percentage of maximal time lags on the number of unsolved problems (Fig. 13 and 14) can be explained in a similar manner. The introduction of maximal time lags leads to more unsolved problems, since maximal time lags can heavily reduce the number of feasible solutions for certain problem instances (even to zero), hence making it more difficult to find one. However, at the same time, the inclusion of additional maximal time lags make it possible for the lower bounds and the dominance rules to kick in (especially the preprocessing rules), leading to less unsolved problems. Although the number of infeasible problems will generally increase as the number of maximal time lags increases, the number of unsolved problems given a certain time limit may decrease again. The effect of the percentage of maximal time lags on the average deviation from the optimal solution (Fig. 15) seems to indicate that the inclusion of more maximal time lags makes the problem more difficult in that it is harder for truncated branch-and-bound procedures to obtain near-optimal solutions.

The effect of *RF*, which can be seen from Fig. 16 through 19 is similar to the effect of *RF* on the computational complexity of the RCPSP as was reported by Kolisch et al. (1995). That is, the higher *RF*, the more difficult it is to solve the corresponding RCPSP(-GPR). The number of problems solved to optimality decreases significantly (Fig. 16), the number of unsolved problems increases substantially (Fig. 17 and 18), as does the average deviation from the best known solution, although a *RF* of 0.75 does not yield very different results from a *RF* of 1.0 (Fig. 19). An opposite effect can be observed for *RS*, as was also observed by Kolisch et al. (1995) and De Reyck and Herroelen (1996b). When *RS* increases, the number of problems solved to optimality increases dramatically (Fig. 20), the number of unsolved problems decreases even more dramatically (Fig. 21 and 22) as does the average deviation from the near-optimal solution (Fig. 23). The strong effects of *RF* and *RS*, and even more pronounced for *RS* than for *RF*, lead us to believe that the effect of resource-based measures on the computational complexity of the RCPSP-GPR is much

larger than the effect of network-based measures. A similar observation for the RCPSP has been made by De Reyck and Herroelen (1996b).

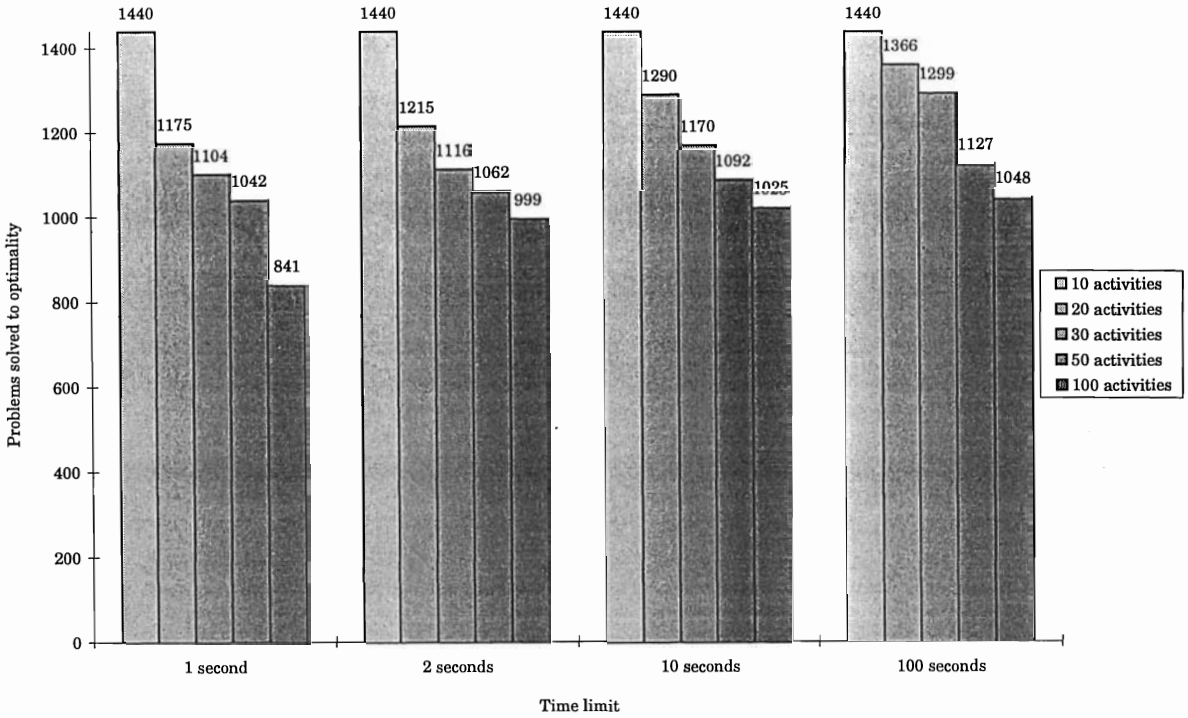

Fig. 2. The effect of problem size on the number of problems solved to optimality (*LB-approach*)

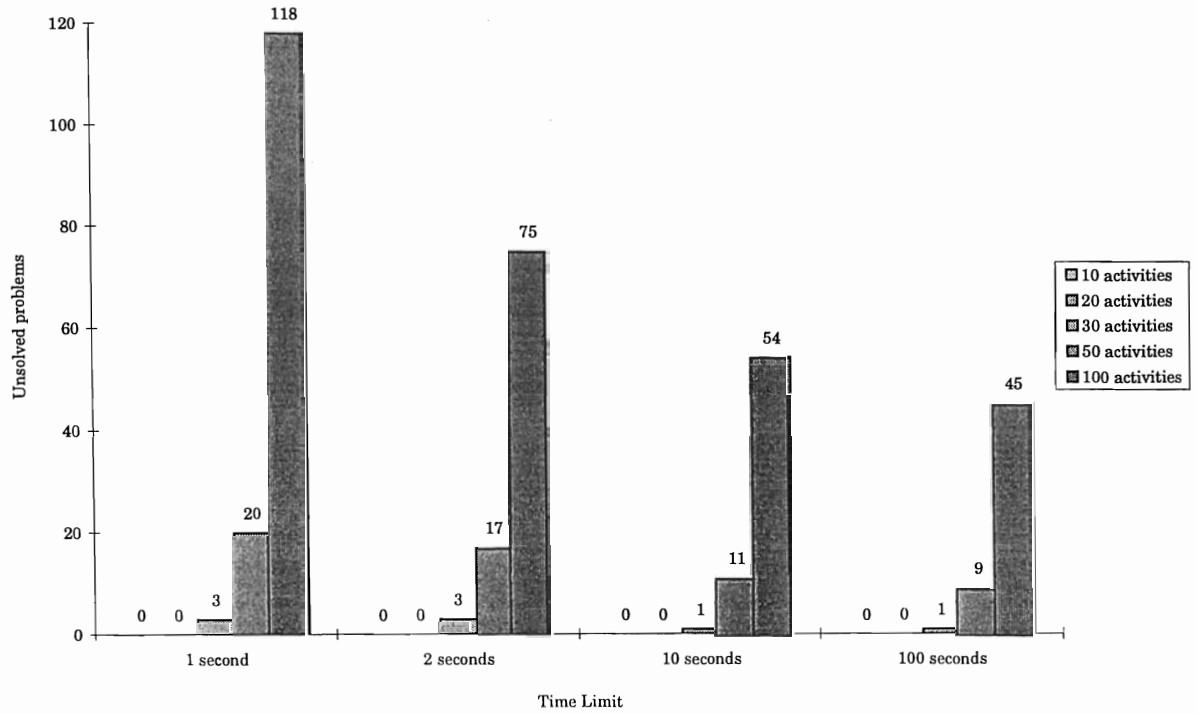

Fig. 3. The effect of problem size on the number of unsolved problems (*LB-approach*)

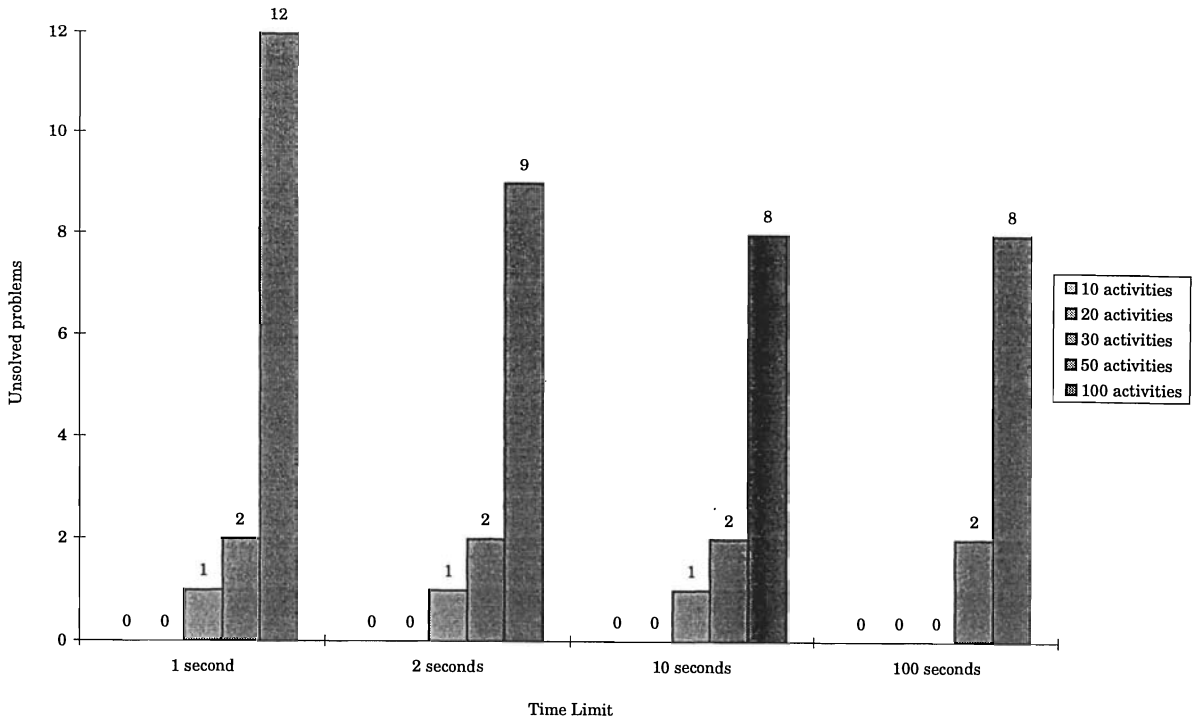

Fig. 4. The effect of problem size on the number of unsolved problems (*TWS-approach*)

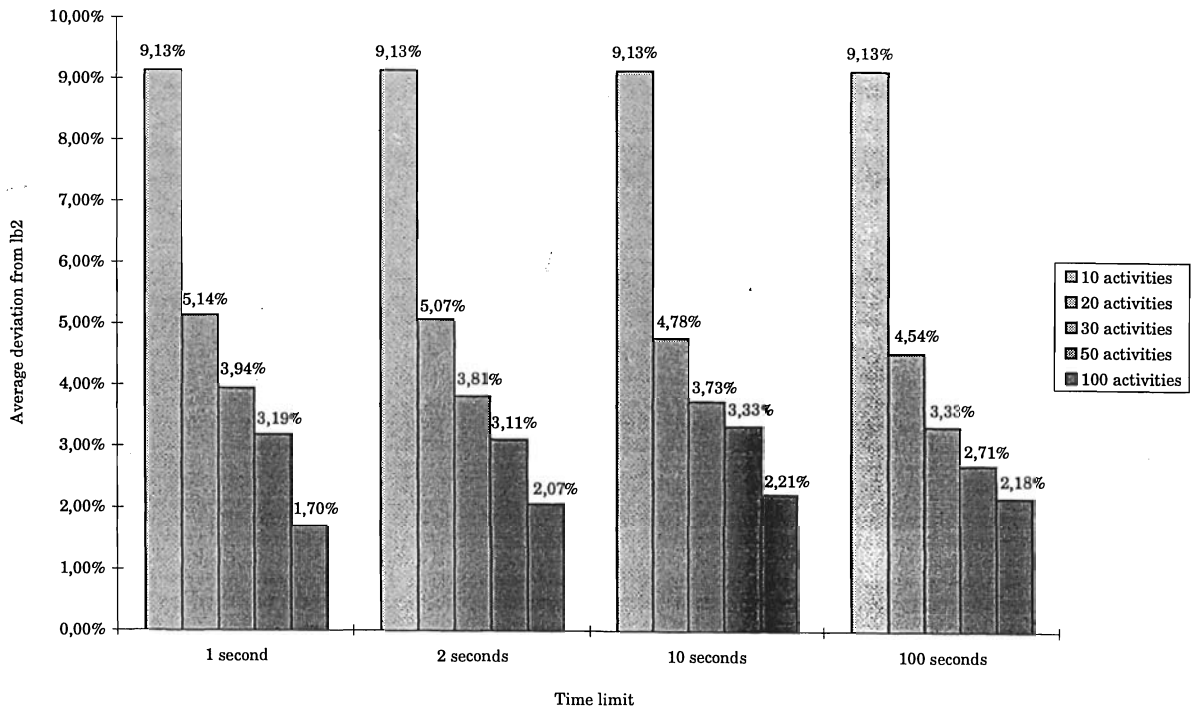

Fig. 5. The effect of problem size on the average deviation from lb_2 (*LB-approach*)

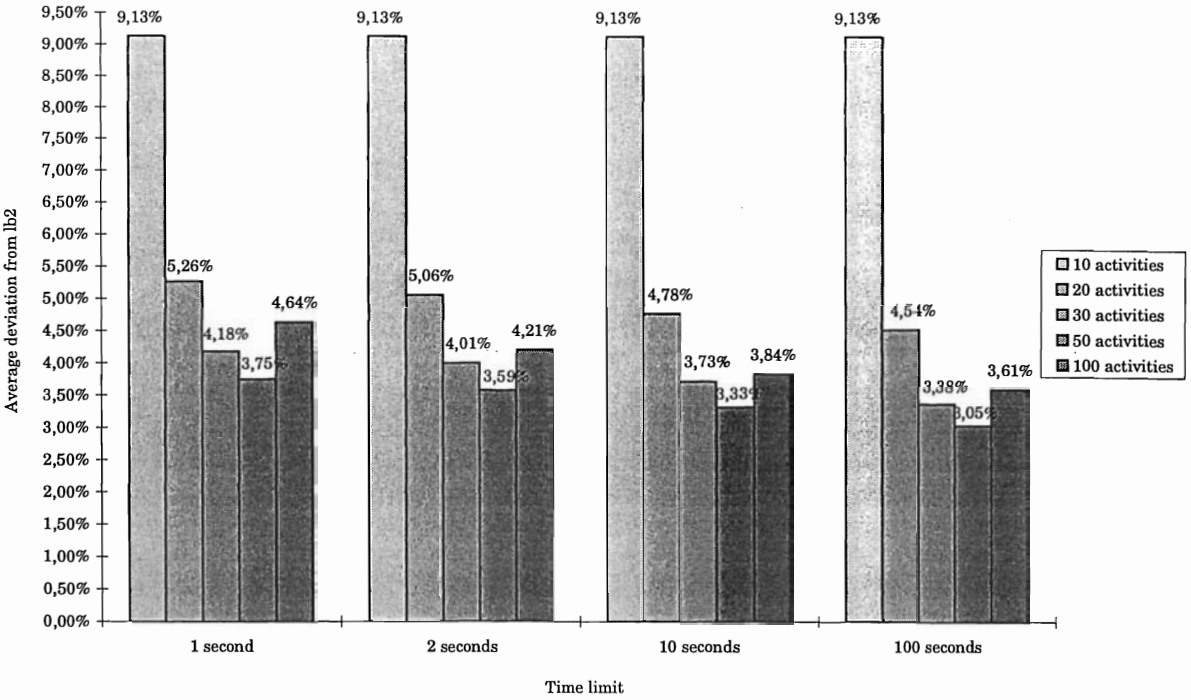

Fig. 6. The effect of problem size on the average deviation from lb_2 (*TWS-approach*)

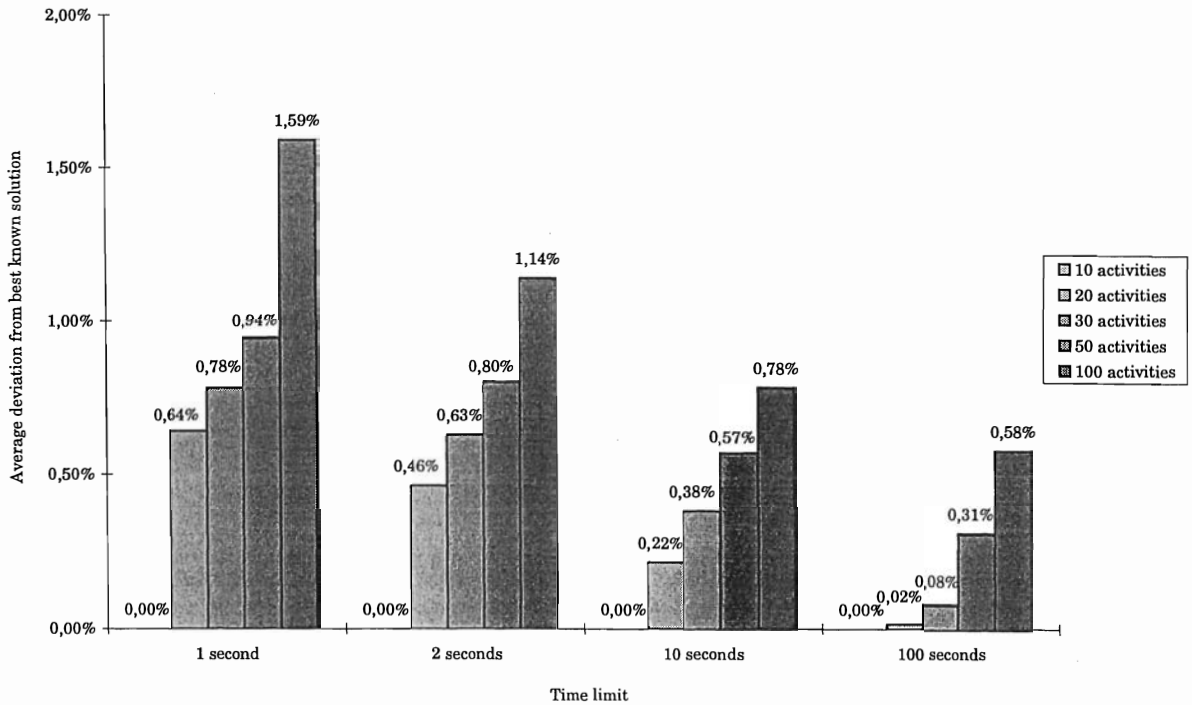

Fig. 7. The effect of problem size on the average deviation from the best known solution (*TWS-approach*)

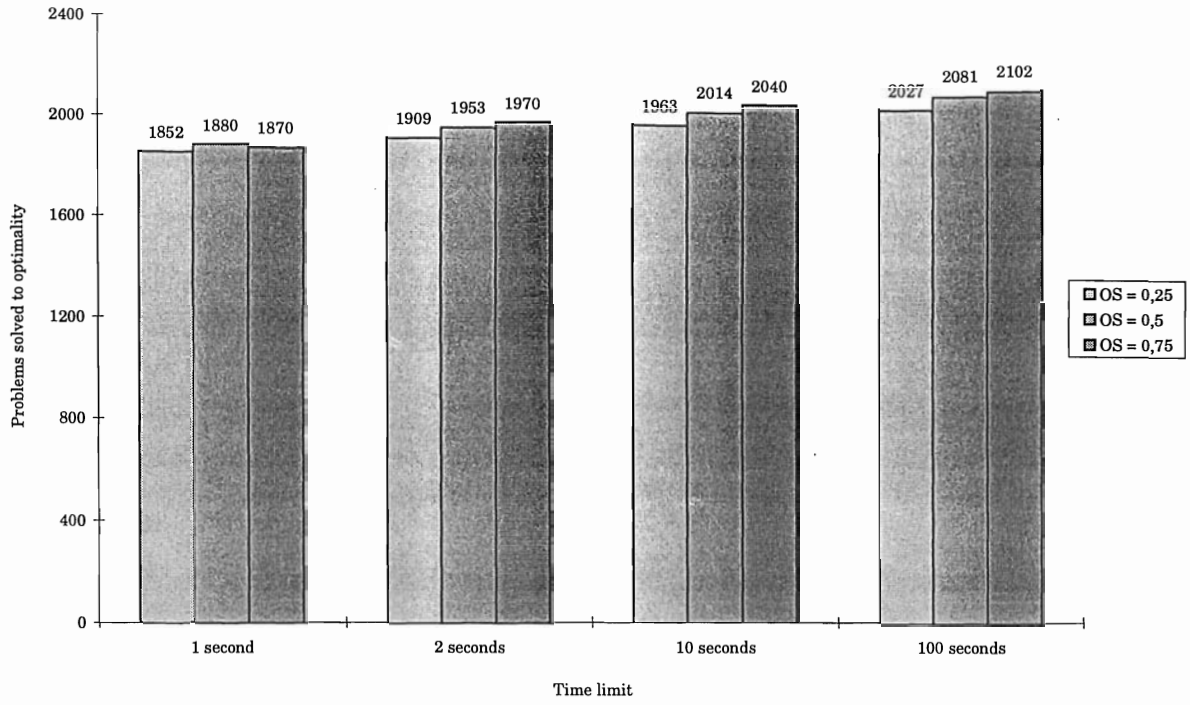

Fig. 8. The effect of *OS* on the number of problems solved to optimality (*LB-approach*)

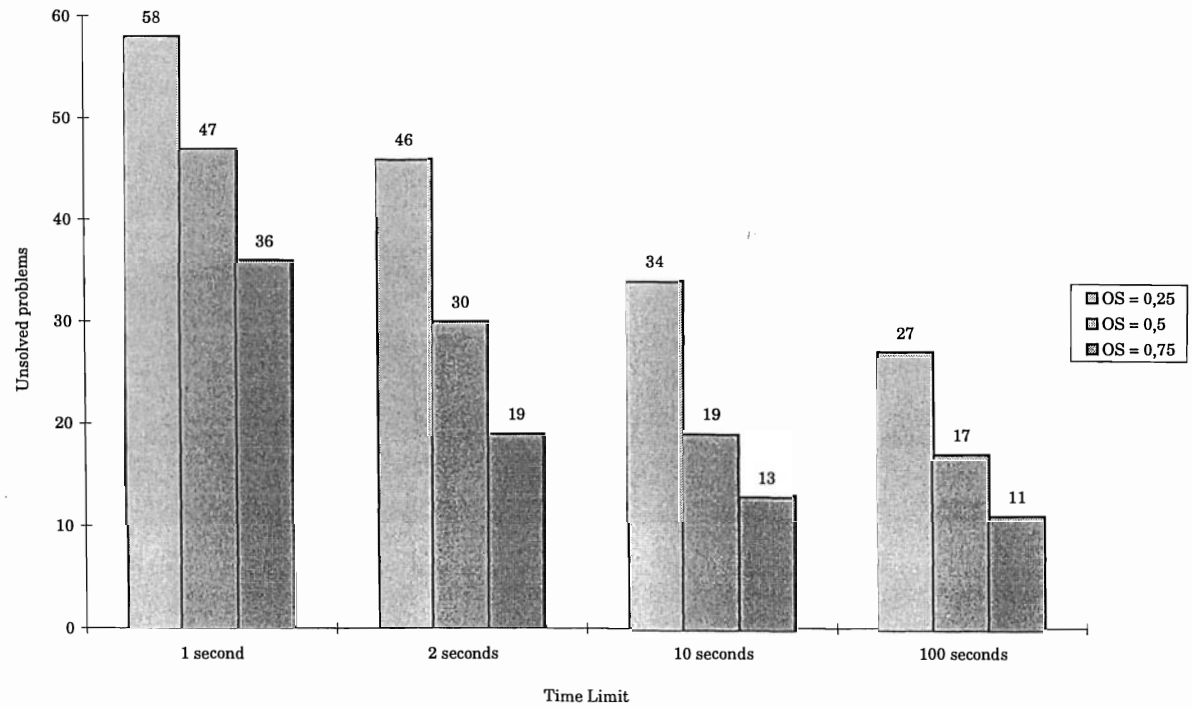

Fig. 9. The effect of *OS* on the number of unsolved problems (*LB-approach*)

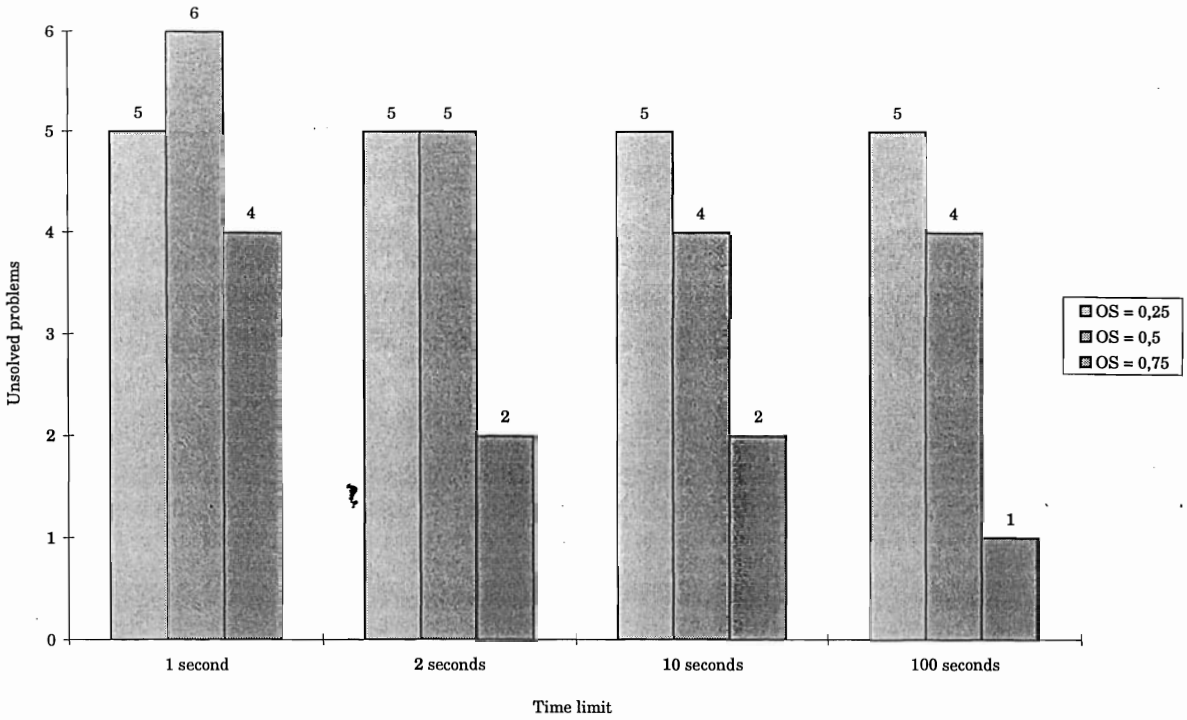

Fig. 10. The effect of *OS* on the number of unsolved problems (*TWS-approach*)

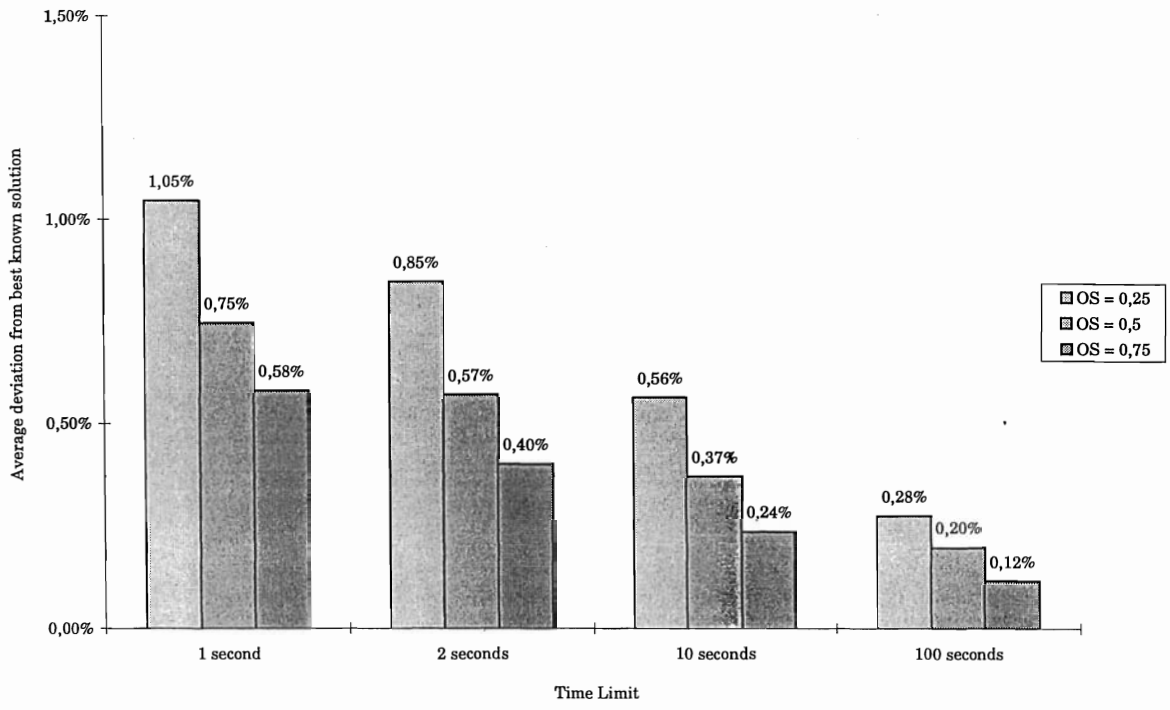

Fig. 11. The effect of *OS* on the average deviation from the best known solution (*TWS-approach*)

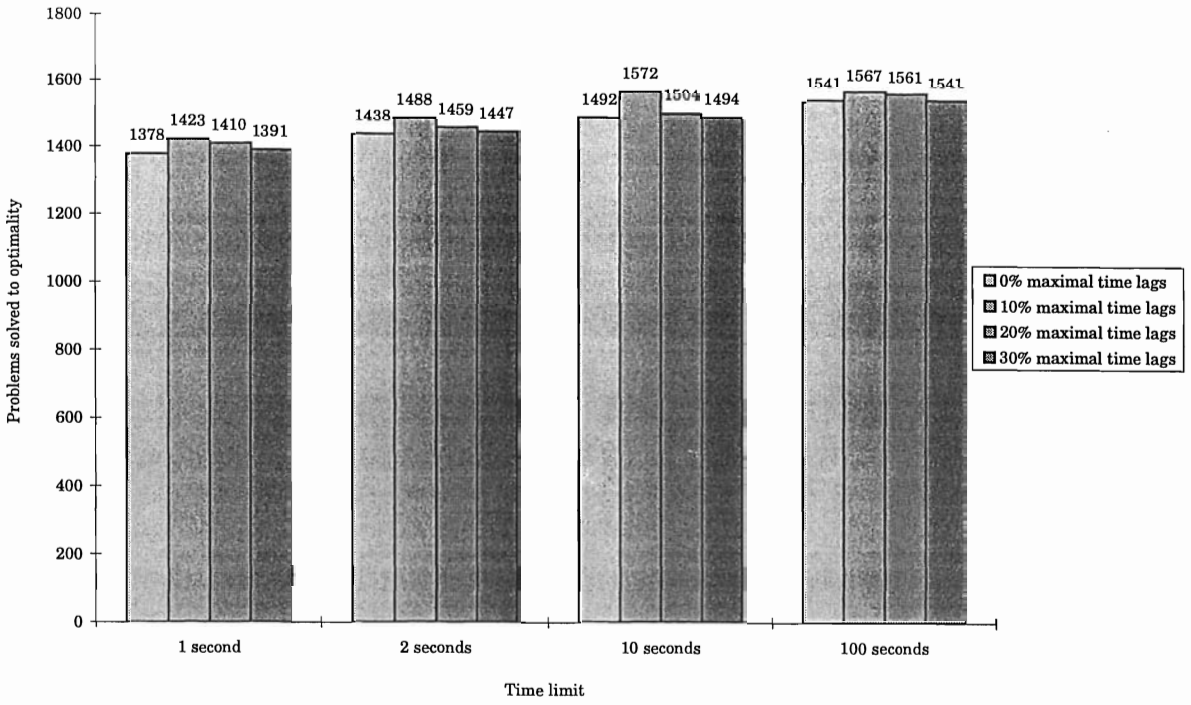

Fig. 12. The effect of % maximal time lags on the number of problems solved to optimality (*LB*)

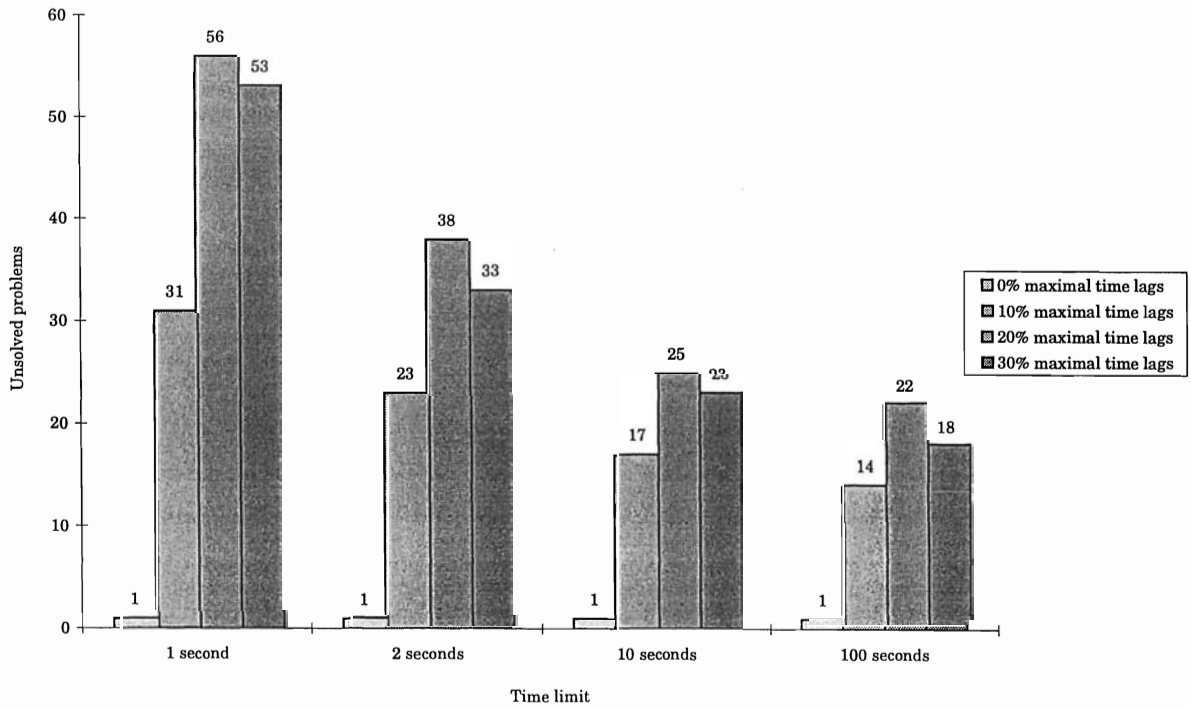

Fig. 13. The effect of % maximal time lags on the number of unsolved problems (*LB-approach*)

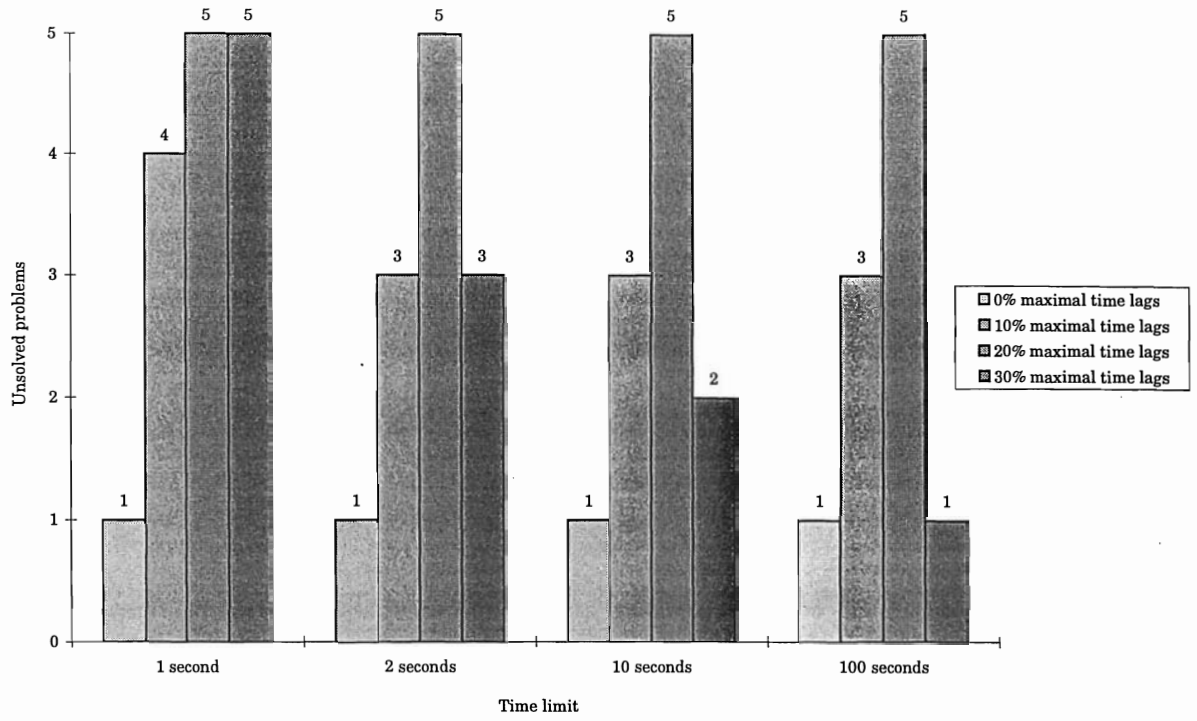

Fig. 14. The effect of % maximal time lags on the number of unsolved problems (*TWS-approach*)

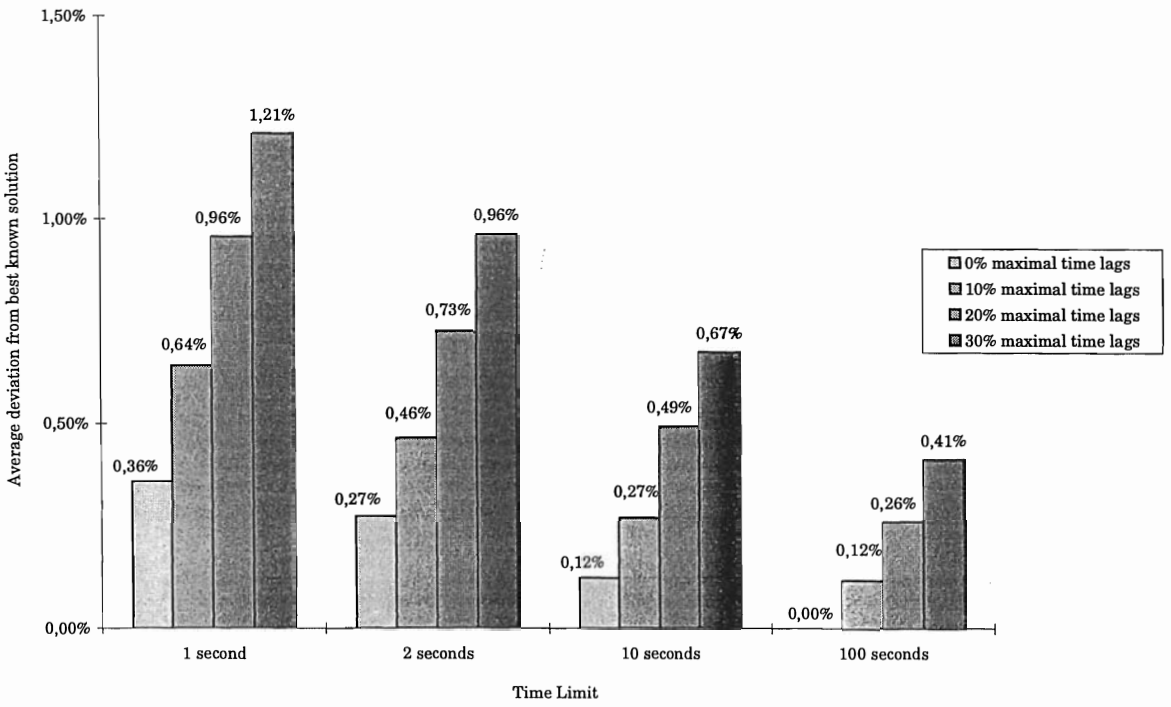

Fig. 15. The effect of % maximal time lags on the avg. dev. from the best known solution (*TWS*)

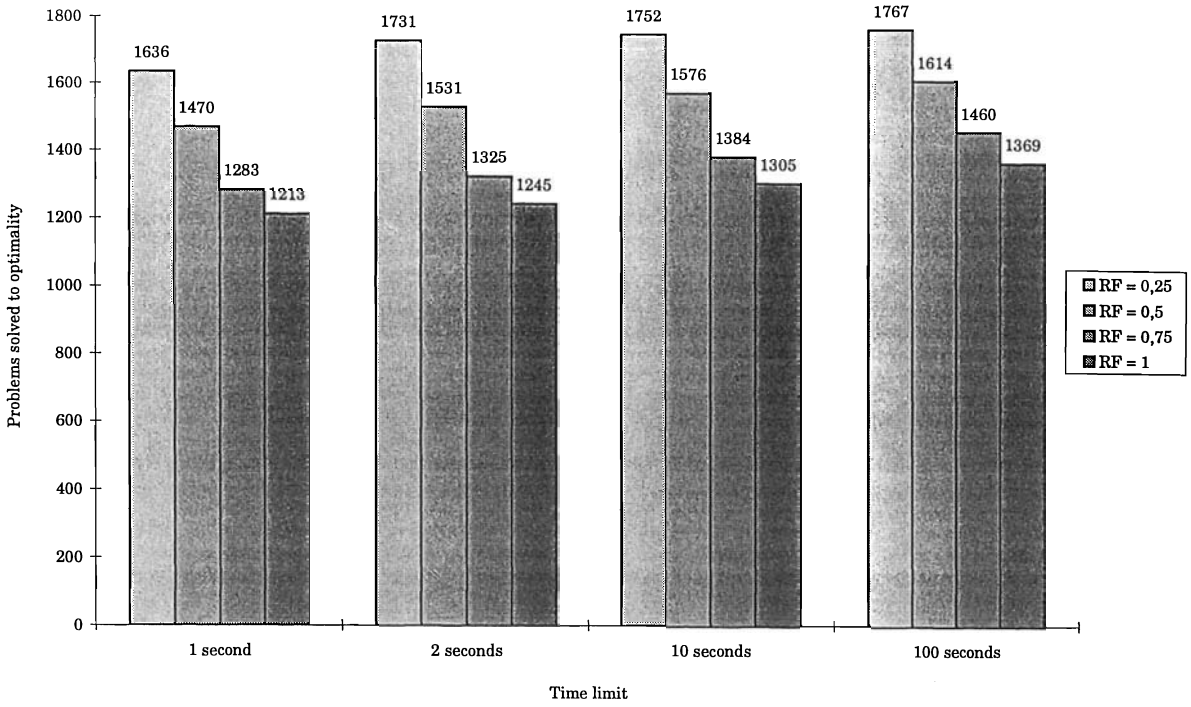

Fig. 16. The effect of *RF* on the number of problems solved to optimality (*LB-approach*)

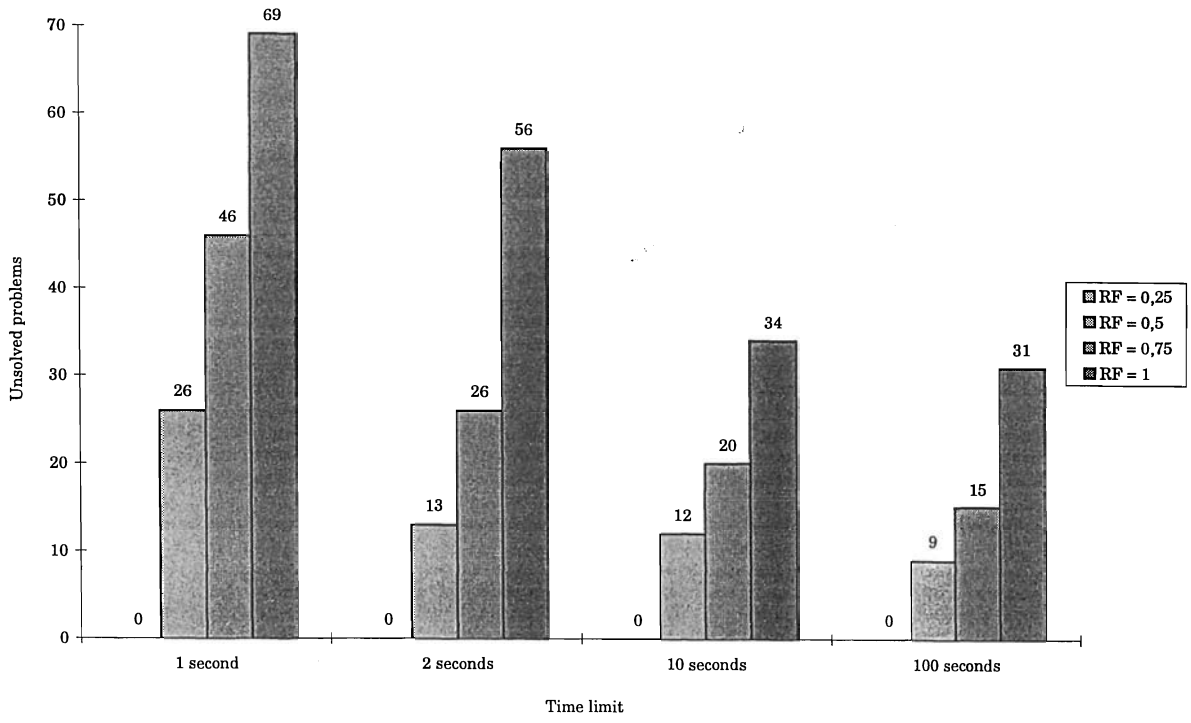

Fig. 17. The effect of *RF* on the number of unsolved problems (*LB-approach*)

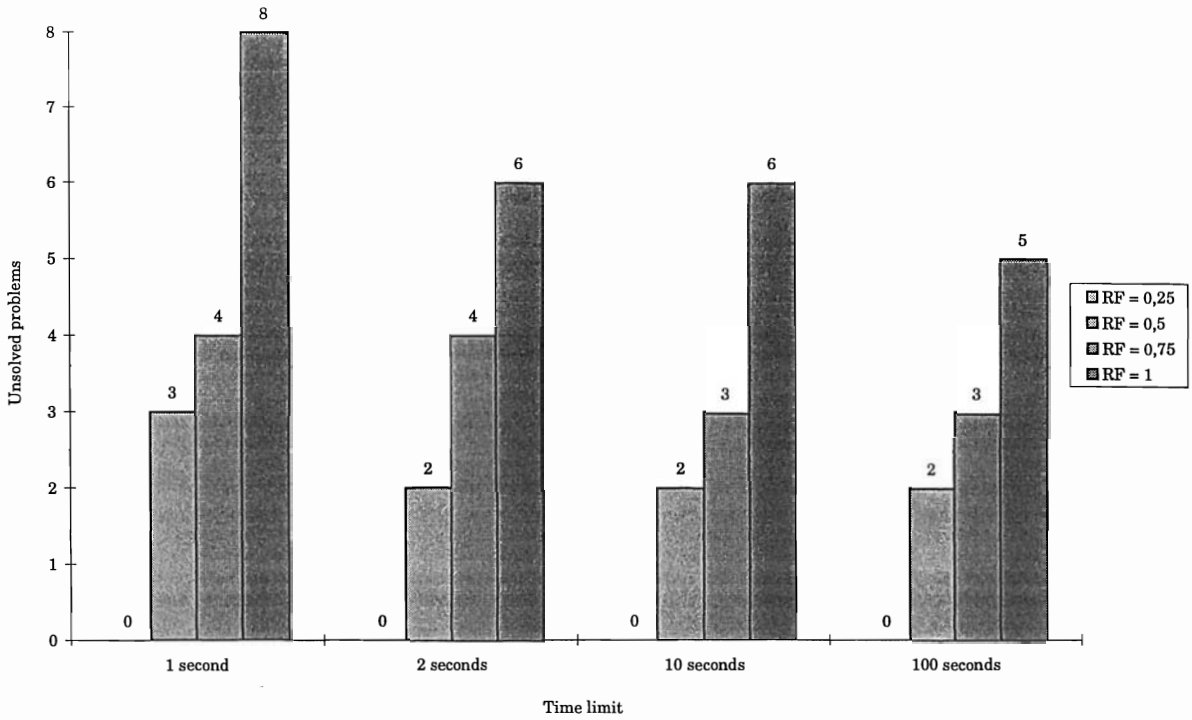

Fig. 18. The effect of *RF* on the number of unsolved problems (*TWS-approach*)

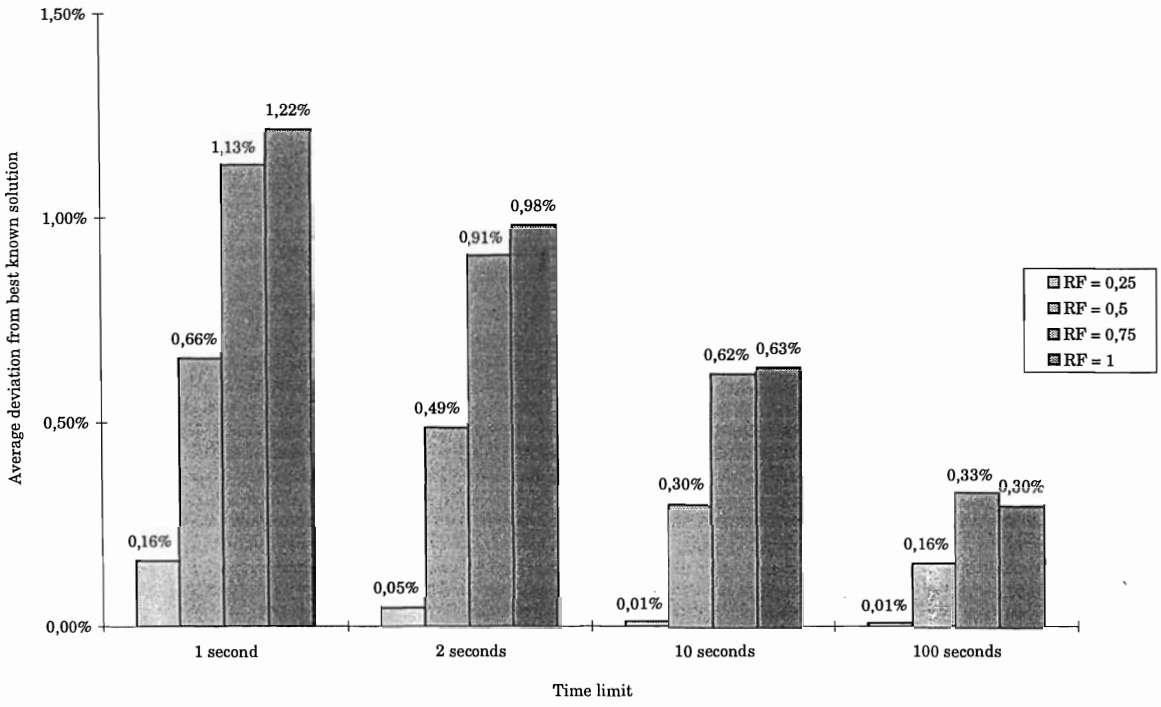

Fig. 19. The effect of *RF* on the average deviation from the best known solution (*TWS-approach*)

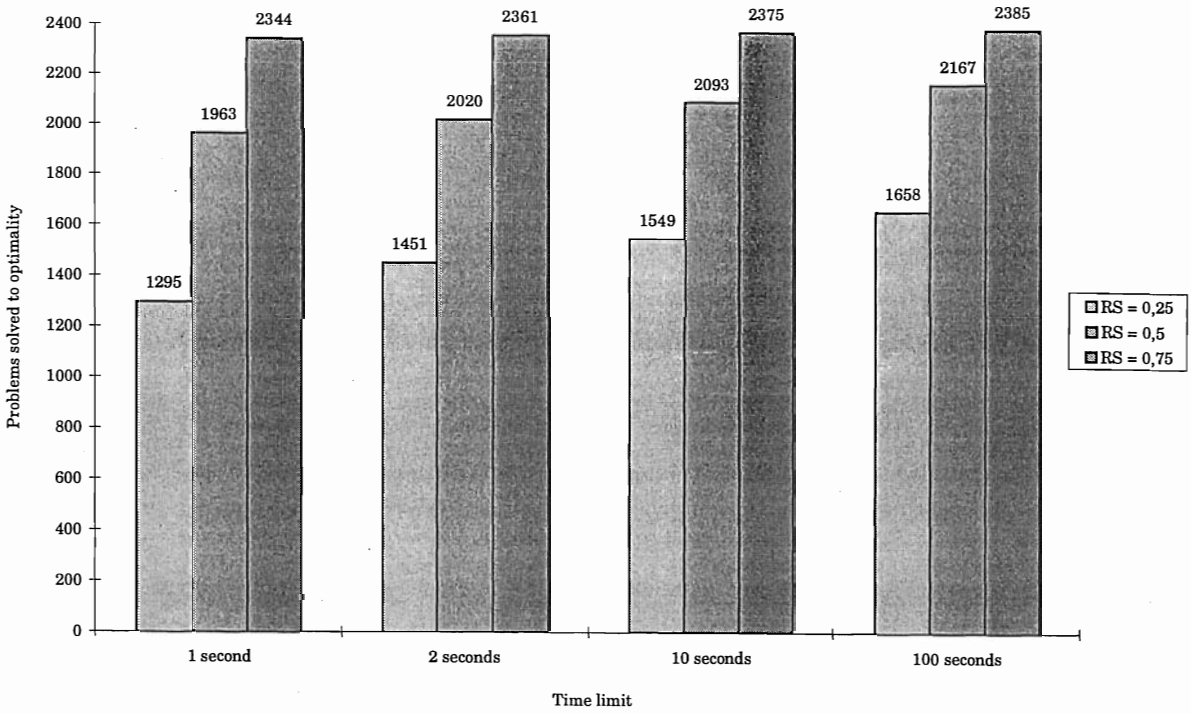

Fig. 20. The effect of *RS* on the number of problems solved to optimality (*LB-approach*)

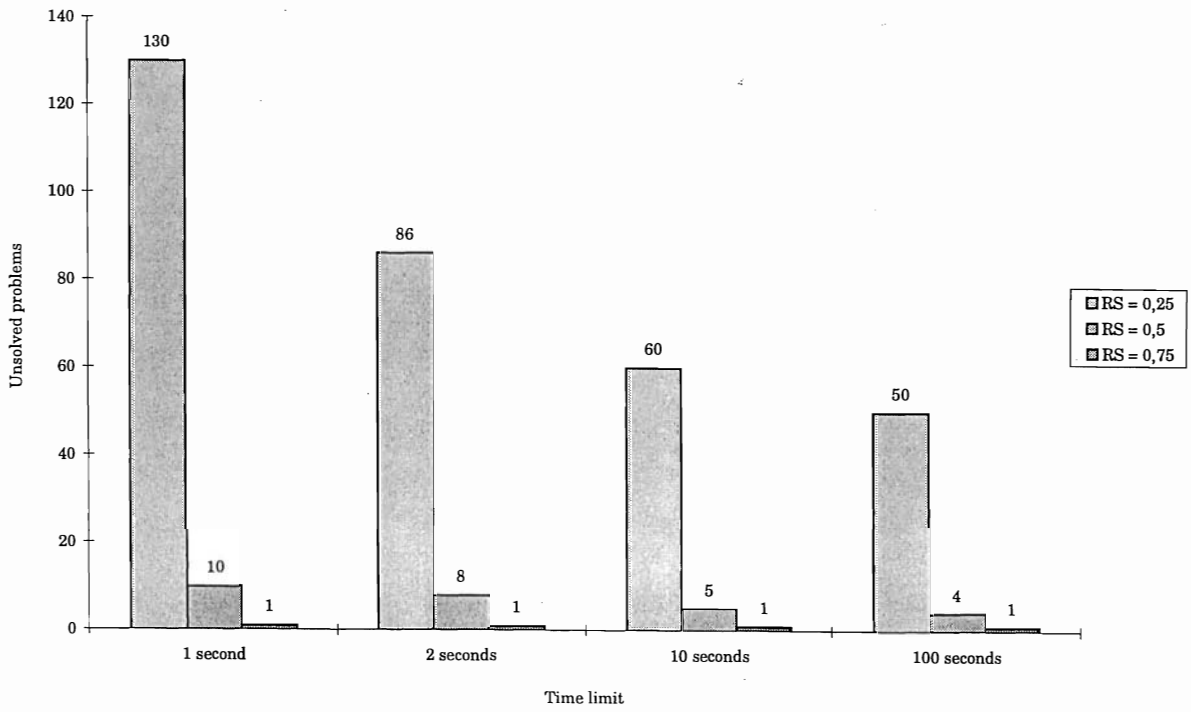

Fig. 21. The effect of *RS* on the number of unsolved problems (*LB-approach*)

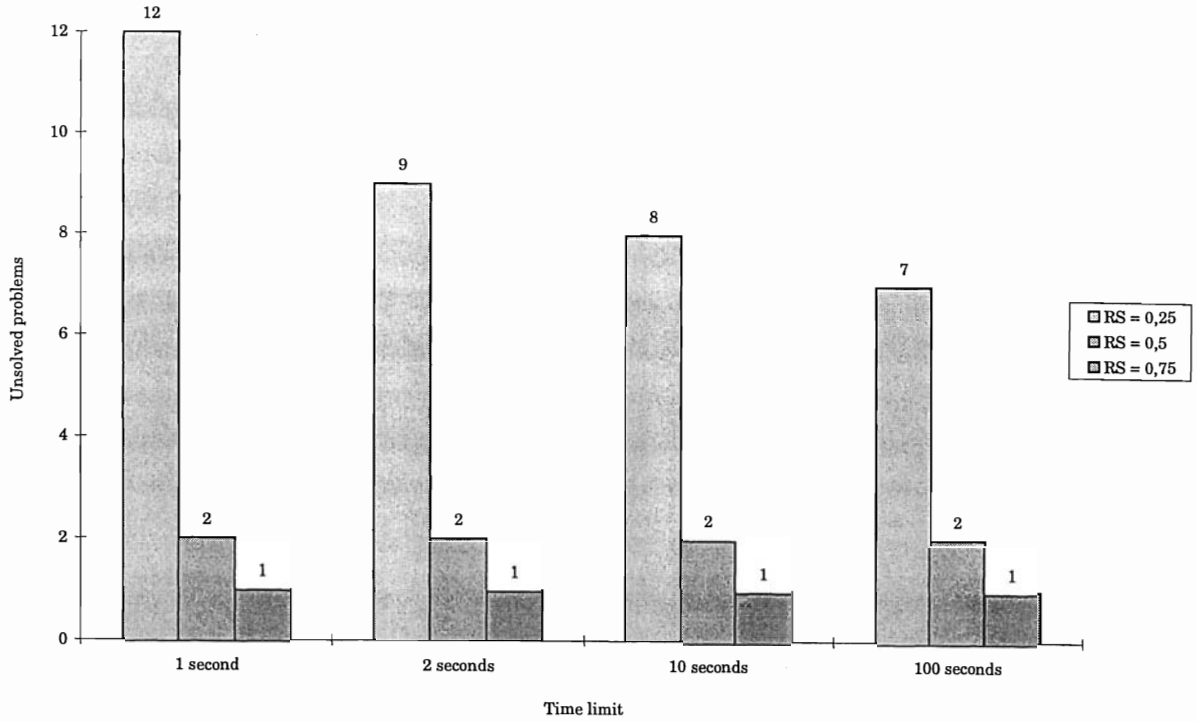

Fig. 22. The effect of *RS* on the number of unsolved problems (*TWS-approach*)

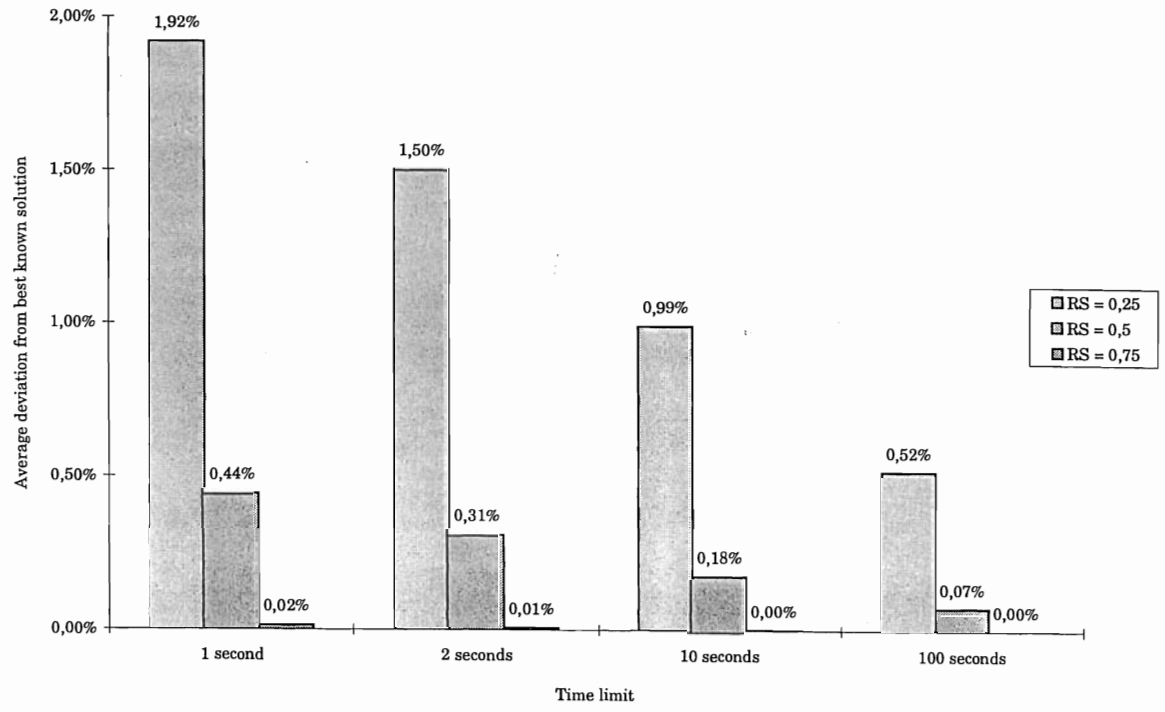

Fig. 23. The effect of *RS* on the average deviation from the best known solution (*TWS-approach*)

7. Conclusions

In this paper we present new computational experience with the branch-and-bound procedure presented in De Reyck and Herroelen (1996a). Results are reported on three problem sets consisting of 1080, 1440 and 7200 problem instances, the first two of which have been previously presented in the literature (Schwindt, 1996; Franck and Neumann, 1996). These problem sets have been generated with ProGen/max, the new random problem generator of Schwindt (1995) which can generate RCPSP-GPR instances as well as other types of generalized resource-constrained project scheduling problems. The results concern the effect of several types of problem characteristics on the complexity of the RCPSP-GPR and include a comparison with other computational results reported in the literature.

Demeulemeester and Herroelen (1995) have shown that a truncated version of their optimal procedure for the RCPSP is capable of outperforming the best heuristic procedures available, based on the problem set of Patterson (1984) and the problem set of Kolisch et al. (1995). These problem sets contain up to 50 activities. However, it is also generally accepted that for larger problem instances, heuristic procedures will have competitive advantages. Nevertheless, using the problem sets described above, we have shown that even for large problem instances (100 activities is very large for the RCPSP-GPR) and even a more complex problem type, a truncated branch-and-bound procedure can outperform a combination of the best heuristic procedures available in less than, on the average, 2 seconds of computation time, whereas the heuristics themselves have much higher CPU time requirements.

References

Bartusch, M., Möhring, R.H. and Radermacher, F.J., 1988, "Scheduling project networks with resource constraints and time windows", *Annals of Operations Research*, 16, 201-240.

Brinkmann, K. and Neumann, K., 1994, "Heuristic procedures for resource-constrained project scheduling with minimal and maximal time lags: the minimum project-duration and resource-levelling problem", Technical Report WIOR-443, Institut für Wirtschaftstheorie und Operations Research, Universität Karlsruhe.

Dar-El, E. M., 1973, "MALB - A heuristic technique for balancing large single-model assembly lines", *AIIE Transactions*, 5, 343-356.

Demeulemeester, E. and Herroelen, W., 1992, "A branch-and-bound procedure for the multiple resource-constrained project scheduling problem", *Management Science*, 38, 1803-1818.

Demeulemeester, E. and Herroelen, W., 1995, "New benchmark results for the resource-constrained project scheduling problem", Research Report 9521, Department of Applied Economics, Katholieke Universiteit Leuven.

De Reyck, B., 1995, "On the use of the restrictiveness as a measure of complexity for resource-constrained project scheduling", Research Report 9535, Department of Applied Economics, Katholieke Universiteit Leuven.

De Reyck, B. and Herroelen, W., 1996a, "A branch-and-bound procedure for the resource-constrained project scheduling problem with generalized precedence relations", Research Report 9613, Department of Applied Economics, Katholieke Universiteit Leuven.

De Reyck, B. and Herroelen, W., 1996b, "On the use of the complexity index as a measure of complexity in activity networks", *European Journal of Operational Research*, to appear.

Franck, B. and Neumann, K., 1996, "Priority-rule methods for the resource-constrained project scheduling problem with minimal and maximal time lags - an emprical analysis", *Fifth International Workshop on Project Management and Scheduling*, 11 - 13 april, Poznan.

Kao, E. P. C. and Queyranne, M., 1982, "On dynamic programming methods for assembly line balancing", *Operations Research*, 30, 375-390.

Kolisch, R., Sprecher, A. and Drexl, A., 1995, "Characterization and generation of a general class of resource-constrained project scheduling problems", *Management Science*, 41 (10), 1693-1703.

Mastor, A. A., 1970, "An experimental and comparative evaluation of production line balancing techniques", *Management Science*, 16 (11), 728-746.

Neumann, K. and Schwindt, C., 1995, "Projects with minimal and maximal time lags: construction of activity-on-node networks and applications", Technical Report WIOR-447, Institut für Wirtschaftstheorie und Operations Research, Universität Karlsruhe.

Neumann, K. and Zhan, J., 1996, "Heuristics for the minimum project-duration problem with minimal and maximal time lags under fixed resource constraints", *Journal of Intelligent Manufacturing*, to appear.

Pascoe, T.L., 1966, "Allocation of resources - CPM", *Revue Française de Recherche Opérationelle*, 38, 31-38.

Patterson, J.H., 1984, "A comparison of exact procedures for solving the multiple constrained resource project scheduling problem", *Management Science*, 30, 854-867.

Schwindt, C., 1995, "ProGen/max: a new problem generator for different resource-constrained project scheduling problems with minimal and maximal time lags", Technical Report WIOR-449, Institut für Wirtschaftstheorie und Operations Research, Universität Karlsruhe.

Schwindt, C., 1996, private communication.

Schwindt, C. and Neumann, K., 1996, "A new branch-and-bound-based heuristic for resource-constrained project scheduling with minimal and maximal time lags", *Fifth International Workshop on Project Management and Scheduling*, 11 - 13 April, Poznan.

Thesen, A., 1977, "Measures of the restrictiveness of project networks", *Networks*, 7, 193- 208.

Zhan, J., 1994, "Heuristics for scheduling resource-constrained projects in MPM networks", *European Journal of Operational Research*, 76, 192-205.

Figure Captions

Table Captions