Bounding Strategies for Scheduling on Identical Parallel Machines

Mohamed Haouari[1], **Anis Gharbi**[1,2], **Mahdi Jemmali**[1]

[1]CORG - ROI, Ecole Polytechnique de Tunisie, BP 743, 2078, La Marsa, Tunisia.
[2]Department of Applied Mathematics, Institut Supérieur d'Informatique, Ariana, Tunisia.
mohamed.haouari@ept.rnu.tn; anis.gharbi@ept.rnu.tn; mah_jem_2004@yahoo.fr

ABSTRACT

We address the problem of minimizing makespan on identical parallel machines. We propose new lower bounding strategies and heuristics for this fundamental scheduling problem. The lower bounds are based on the so-called lifting procedure. In addition, two optimization-based heuristics are proposed. These heuristics require iteratively solving a subset-sum problem. We present the results of computational experiments that provide strong evidence that the new proposed lower and upper bounds consistently outperform the best bounds from the literature.
Keywords: Scheduling, Identical parallel machines, Lower bounds, Heuristics.

1. INTRODUCTION

Given a set J of n jobs with corresponding processing times $p_1, p_2, ..., p_n$ and m identical machines $(1 < m < n$ and $p_1 \leq p_2 \leq ... \leq p_n)$, the problem is to assign each job to exactly one machine with the objective of minimizing the makespan. This problem, which is denoted by $P||C_{\max}$, is a very basic and fundamental scheduling problem that has been intensely studied since the late 1950s [14].

Despite its deceptive simplicity, the $P\,||\,C_{\max}$ is not easy to solve. Actually, even the two-machine special case (i.e. $P2\,||\,C_{\max}$) is $\mathcal{NP}$-hard. Hundreds of scheduling theory analysts have cumulatively devoted an impressive number of papers to the worst-case and probabilistic analysis of numerous approximation algorithms for this problem. Mokotoff [15] summarizes much of this research. In addition, several heuristics have been proposed [1,2,3,6,7,10, 11,12,20]. However, relatively little attention has been paid so far to the development of lower bounding strategies and exact optimization algorithms. Indeed, in addition to exponential-time dynamic programming algorithms that can only solve tiny instances [18,19], only two further exact optimization algorithms have been presented so far [4,16]. Actually, Dell'Amico and Martello [4] describe also effective lower bounds as well as an effective heuristic which is referred to as Multi-Subset (MS). Finally, Webster [21] presented a lower bound which is applicable also to a slightly more general version of $P\,||\,C_{\max}$ with unequal machine availability times.

Firstly, we describe a lifting procedure which aims at enhancing previously developed lower bounds. Secondly, we describe two heuristics which require iteratively solving a subset-sum problem. We carried out extensive computational experiments on a large set of randomly generated instances which provide empirical evidence of the tightness of the proposed lower and upper bounds.

2. LIFTED LOWER BOUNDS

Haouari and Gharbi [8] proved that in any feasible schedule with n jobs and m machines, there is at least a set of k machines $(1 \leq k \leq m)$ which must process at least $\lambda_k(n) = k\lfloor n/m \rfloor + \min(k, n - \lfloor n/m \rfloor m)$ jobs. Given an instance P defined on a jobset J, we can derive m instances P_k $(k = 1, ..., m)$, where P_k is a $P\,||\,C_{\max}$ instance defined on k machines and the jobset $J_n^k \subseteq J$ that is obtained by considering the $\lambda_k(n)$ jobs of J with smallest processing times. Let $L(J_n^k)$ denote a valid lower bound on the optimal makespan of P_k. Hence, a valid bound for instance P is $\bar{L}(J) = \max_{1\leq k\leq m} L(J_n^k)$. A stronger lifted lower bound can be derived by considering different subsets of J. Define $J_l \subseteq J$ as the subset of jobs that contains the l jobs of J with largest processing times, and $J_l^k \subseteq J_l$ as the subset of jobs that is obtained by considering the $\lambda_k(l)$ jobs of J_l with smallest processing times. Hence, a stronger lifted bound is $\tilde{L}(J) = \max_{1\leq k\leq m}\left\{\max_{1\leq l\leq n} L(J_l^k)\right\}$. It is shown, in the context of the bin packing problem [9], that, under very mild conditions, the only values of l that have to be considered in the computation of $\tilde{L}(.)$ are those such that $l = \alpha m + k$, for $\alpha = 1, ..., \lfloor (n-k)/m \rfloor$. Therefore, $\tilde{L}(.)$ can be obtained after $O(n)$ computations of $L(.)$.

This procedure has been used for lifting the following lower bounds:

$i)$ The trivial bound defined by

$$L_{TV} = \max(p_n, p_{n-m} + p_{n-m+1}, \left\lceil \textstyle\sum_{j=1}^{n} p_j/m \right\rceil)$$

$ii)$ The general bound [21]: Firstly, starting from the original $P\,||\,C_{\max}$ instance, an auxiliary instance is defined. Then, an *optimal* schedule for the auxiliary instance is obtained using the longest processing time rule (LPT) which iteratively schedules a job with the longest processing time on the first available machine. More precisely, the auxiliary instance is defined on m machines and the jobset J' which is constructed in the following way. Let n' $(n' \leq n)$ and ρ denote two positive integers, and τ be equal to $\gcd(\ \rho,\ (p_1 - \lfloor p_1/\rho \rfloor \rho), ..., (p_n - \lfloor p_n/\rho \rfloor \rho))$. Then, $J' = J_1 \cup J_2 \cup J_3$ such that:

- J_1 is a set of n' jobs with respective processing times $\lfloor p_n/\rho \rfloor \rho$, $\lfloor p_{n-1}/\rho \rfloor \rho$, ..., $\lfloor p_{n-n'+1}/\rho \rfloor \rho$,

1-4244-0451-7/06/$20.00 ©2006 IEEE

- J_2 is a set of $\lfloor p_{n-n'}/\rho \rfloor + ... + \lfloor p_1/\rho \rfloor$ jobs with processing time ρ,
- J_3 is a set of $(p_1 - \lfloor p_1/\rho \rfloor \rho)/\tau + ... + (p_n - \lfloor p_n/\rho \rfloor \rho)/\tau$ jobs with processing time τ.

Two values of ρ are used in turn. These values are $\rho_1 = \gcd(p_1, p_2, ..., p_n)$ and $\rho_2 = p_{j_0}$ where j_0 is the last job to finish processing in the LPT schedule constructed for J. For a given ρ, the number n' is the largest integer for which at least one among the following three conditions holds:

(C1) $\lfloor p_{n-j+1}/\rho \rfloor \,/\, \lfloor p_{n-j}/\rho \rfloor$ is integer for $j = 1, ..., n'-1$,

(C2) $\lfloor p_{n-m+2}/\rho \rfloor \geq \lfloor p_{n-m+1}/\rho \rfloor + ... + \lfloor p_{n-n'+3}/\rho \rfloor$,

(C3) $n' \leq 2m$ and $\lfloor p_{n+n'-2m}/\rho \rfloor \leq 2 \lfloor p_1/\rho \rfloor$.

Each value of ρ yields a lower bound. We denote by L_{GB} the maximum of these two values.

$iii)$ Bin Packing-based bounds: Given an instance of $P||C_{\max}$ and the associated decision problem which consists in checking whether the n jobs could be processed on the m machines such that the makespan does not exceed a trial value C. A "no" answer to this decision problem is provided if a lower bound on the minimal number of bins (machines) that are required for packing the n items (jobs) exceeds m. Consequently, a valid lower bound for $P \,||\, C_{\max}$ is $C^* + 1$ where C^* is the largest value of C that yields a "no" answer. For deriving lower bounds for $P||C_{\max}$, we successively considered three BPP lower bounds

• The bound L_{MT}^{BPP} of Martello and Toth [13]: Let V be the set of all distinct values $p_j \leq \frac{C}{2}$. For each integer $p \in V$, let $J_1(p) = \{j \in J : C - p < p_j\}$, $J_2(p) = \{j \in J : \frac{C}{2} < p_j \leq C - p\}$ and $J_3(p) = \{j \in J : p \leq p_j \leq \frac{C}{2}\}$. Then L_{MT}^{BPP} is defined by

$$L_{MT}^{BPP} = \max_{p\in V} \{|J_1(p)| + |J_2(p)| + \max(0, \left\lceil (\textstyle\sum_{j\in J_3(p)} p_j - |J_2(p)|\, C + \sum_{j\in J_2(p)} p_j)/C \right\rceil)\}$$

• The bound L_{DM}^{BPP} of Dell'Amico and Martello [4]: In the same vein as for the computation of L_{MT}^{BPP}, the bound L_{DM}^{BPP} is defined by

$$L_{DM}^{BPP} = \max_{p\in V} \{|J_1(p)| + |J_2(p)| + \max(0, \left\lceil (|J_3(p)| - \textstyle\sum_{j\in J_2(p)} \lfloor (C-p_j)/p \rfloor) / \lfloor C/p \rfloor \right\rceil)\}$$

• The class of bounds of Fekete and Schepers [5]: After normalizing the bin capacity to 1 and the weight of each item j to $x_j = p_j/C$, let:

$\diamond\, u^{(h)}(x) = x$ for $x(h+1) \in I\!N$ and $u^{(h)}(x) = \lfloor (h+1)x \rfloor \frac{1}{h}$ otherwise,

$\diamond\, U^{(\varepsilon)}(x) = 1$ for $x > 1 - \varepsilon$, $U^{(\varepsilon)}(x) = x$ for $\varepsilon \leq x \leq 1 - \varepsilon$, and $U^{(\varepsilon)}(x) = 0$ for $x < \varepsilon$.

A class of lower bounds $L_{FS}^{(q)}$ ($q \geq 2$) is defined in the following way:

$$L_{FS}^{(q)}(J) = \max\{L_{MT}^{BPP}(J), \max_{2\leq h\leq q}\{\max_{\varepsilon\in[0,\frac{1}{2}]} \left\lceil \textstyle\sum_{j=1}^{n} u^h \circ U^{(\varepsilon)}(x_j) \right\rceil\}\}$$

In the sequel, we define L_{MT}, L_{DM}, and L_{FS} as the $P \,||\, C_{\max}$ lower bounds that are yielded by L_{MT}^{BPP}, L_{DM}^{BPP}, and $L_{FS}^{(20)}$ respectively. For each lower bound L_ξ ($\xi \in \{TV, GB, MT, DM, FS\}$) we denote by $\tilde{L}_\xi$ the corresponding lifted bound.

Example : Consider the 8 jobs-3 machines instance with the following processing times {55,56,83,96,97,98,98,99}. We have $L_{TV} = \max(99, 98 + 97, \lceil 682/3 \rceil) = 228$. After applying the lifting procedure to the bound L_{TV}, we obtain $\tilde{L}_{TV} = 243$. This value corresponds to $k = 2$ and $l = 8$, which yield $\lambda_k(l) = 2 \lfloor 8/3 \rfloor + \min(2, 8 - 3 \lfloor 8/3 \rfloor) = 6$ and

$$L_{TV}(J_l^k) = \max(98, 97 + 96, \lceil (55 + 56 + 83 + 96 + 97 + 98)/2 \rceil) = 243.$$

Moreover, $L_{DM} = L_{MT} = L_{GB} = 228$ and $\tilde{L}_{DM} = \tilde{L}_{MT} = \tilde{L}_{GB} = 243$. Now, consider the trial value $C = 240$. For $\varepsilon = \frac{55}{240}$ and $h = 4$, we get

$$\left\lceil \sum_{j=1}^{n} u^h \circ U^{(\varepsilon)}(x_j) \right\rceil = \left\lceil \frac{1}{4} + \frac{1}{4} + \frac{1}{4} + \frac{2}{5} + \frac{1}{2} + \frac{1}{2} + \frac{1}{2} + \frac{1}{2} \right\rceil = 4 > m$$

Therefore, $L_{FS} = 241$ is a valid lower bound. Lifting this bound yields $\tilde{L}_{FS} = 247$. This value is obtained by setting $C = 246$, $k = 2$, $l = 8$, $\varepsilon = \frac{55}{246}$ and $h = 17$. We get

$$\left\lceil \sum_{j=1}^{\lambda_k(l)} u^h \circ U^{(\varepsilon)}(x_j) \right\rceil = \left\lceil \frac{4}{17} + \frac{4}{17} + \frac{6}{17} + \frac{7}{17} + \frac{7}{17} + \frac{7}{17} \right\rceil = 3 > k$$

3. HEURISTICS

3.1. A subset sum problem based heuristic

This heuristic builds a feasible solution in a greedy way by iteratively solving a family of subset sum problems P_k ($k = 1, ..., m-1$) defined by

$$(P_k) : \text{Minimize } \textstyle\sum_{j\in S_k} p_j x_j$$

subject to:

$$\textstyle\sum_{j\in S_k} p_j x_j \geq L(S_k, m-k+1)$$

$$x_j \in \{0, 1\} \quad \forall j \in S_k$$

where

- $S_1 = J$, and $S_{k+1} = S_k \setminus J_k$ where J_k is an optimal subset for P_k ($k = 1, ..., m-1$),
- $L(S, k)$ denotes a valid lower bound on the makespan of a reduced instance defined on $k \leq m$ machines and a subset of jobs $S \subseteq J$.

Hence, the algorithm starts by solving P_1 in order to find a subset of jobs J_1 whose total processing time is minimal but not less than the value of a lower bound. These jobs will be assigned to a first machine. Then, the algorithm computes a lower bound on the optimal makespan of the instance which is defined by $m-1$ machines and the jobset $J \setminus J_1$. Again, a subset sum problem is solved in order to determine an optimal subset of jobs which will be assigned to a second machine, and so on. Obviously, the jobs that belong to $J_m = S_{m-1} \setminus J_{m-1}$ are assigned to the m^{th} machine. We refer to the value of the approximate solution that is obtained with this heuristic as U_{SS}. In our implementation, we used the lower bound $\tilde{L}_{TV}$ in the computation of U_{SS}.

It is worth noting that although the SSP is $\mathcal{NP}$-hard, it can be efficiently solved in a pseudo-polynomial time using the dynamic programming algorithm developed by Pisinger [17].

3.2. A multi-start subset-sum based improvement heuristic

Prior to describing the second heuristic, we introduce a reformulation of $P2 \,||\, C_{\max}$ as the following subset sum problem:

$$\text{Minimize } \textstyle\sum_{j \in J} p_j x_j$$

subject to:

$$\textstyle\sum_{j \in J} p_j x_j \geq \left\lceil \sum_{j \in J} p_j / 2 \right\rceil$$

$$x_j \in \{0,1\} \quad \forall j \in J$$

Based on this reformulation, we implemented a multi-start local search method which requires iteratively solving a sequence of two-machine problems. More precisely, given a feasible schedule σ, let J_k $(k = 1, ..., m)$ denote the subset of jobs that are assigned to machine M_k, and $C_k = \sum_{j \in J_k} p_j$. Assume that the machines are indexed such that $C_1 \leq C_2 \leq ... \leq C_m$. The improvement phase selects the pair of machines (M_1, M_m) and solves the resulting $P2||C_{\max}$ instance which is defined on the jobset $J_1 \cup J_m$. Let $C^1_{\max}$ denote the optimal makespan that is obtained after solving the corresponding subset sum problem. If $C^1_{\max} < C_m$, then the schedules of machines M_1 and M_m are replaced by the new ones, the values C_k $(k = 1, ..., m)$ are updated and the procedure is reiterated. Otherwise, the pair of machines (M_2, M_m) is selected and the corresponding $P2||C_{\max}$ instance is solved as a subset sum problem and so on. The procedure is stopped if all of the machine pairs (M_k, M_m) $(k = 1, ..., m-1)$ are considered in turn with no improvement.

The heuristic is repeated 100 times, starting at each time from a different initial solution. In our implementation, the starting solutions are obtained with a randomized LPT rule. This procedure iteratively selects two unscheduled jobs with the longest processing times and then randomly assigns one out of this pair to the first available machine.

It is worth noting that Ho and Wong [10] were the first to propose to solve the $P||C_{\max}$ by solving a sequence of $P2||C_{\max}$ instances. However, they devised a branch-and-bound algorithm for solving the two-machine problems and they did not embed it within a multi-start local search framework.

4. COMPUTATIONAL RESULTS

The performance of the proposed bounds was evaluated on a large test-bed of 4150 randomly generated instances (including large-sized instances with up to 10000 jobs and 15 machines) and 780 benchmark ones provided by Alvim and Ribeiro [1]. All our experiments were obtained on a Pentium IV 3.2 GHz Personal Computer with 1.5 GB RAM.

4.1. Performance on randomly generated instances

The test-bed we have used consists of instances that were randomly generated. Depending upon the distribution of the processing times, two different classes of instances were obtained:

- **Class 1:** the processing times are drawn from the discrete uniform distribution on [50,100]. This problem set contains 360 instances corresponding to 20 instances for different combinations of n $(10 \leq n \leq 100)$ and m $(3 \leq m \leq 15)$.
- **Class 2:** For each value of n, the number of machines m is set equal to $\frac{2n}{5}$ and the processing times are drawn from the discrete uniform distribution on $[\frac{n}{5}, \frac{n}{2}]$. This class contains 220 instances corresponding to 20 instances for each n $(10 \leq n \leq 200)$.

We observed that the lifting procedure is very effective since all of the lifted bounds consistently outperform the bounds from the literature. Interestingly, we found that even the weakest lifted bound (namely, the lifted trivial bound $\tilde{L}_{TV}$), despite its simplicity, performs quite well and clearly outperforms the best bounds from the literature. Indeed, $\tilde{L}_{TV}$ yielded the maximal value over all of the bounds in 329 out of the 360 instances of Class 1, and in 210 out of the 220 instances of Class 2, whereas the best bound from the literature (namely L_{FS}) yielded the maximal value in 309 instances of Class 1 and in 191 instances of Class 2. The lifted bound $\tilde{L}_{FS}$ provided the best lower bound for all of the instances, but requires (relatively) long CPU times. On the other hand, $\tilde{L}_{MT}$, $\tilde{L}_{DM}$, and $\tilde{L}_{GB}$ have very good performance and are very fast.

We compared the performance of our approximate algorithms with respect to the multi-subset algorithm (MS) of Dell'Amico and Martello [4] as well as to two well-known simple heuristics: the longest-processing time algorithm (LPT), and the multifit algorithm (MF). We observed that MSS performs extremely well since it yields the best upper bound for all of the instances but one, while requiring modest computing time. This heuristic provided proven optimal solutions for 73.10% of the instances (424 out of 580). Moreover, the second proposed heuristic U_{SS} outperforms U_{MS}.

In order to get a better picture of the actual performance of our best lower and upper bounds (i.e. $\tilde{L}_{FS}$ and U_{MSS}), we tested them on a second set of instances that was generated as described in Dell'Amico and Martello [4]. The processing times were generated according to the following distributions:

- **Type 1:** discrete uniform distribution on [1,100]
- **Type 2:** discrete uniform distribution on [20,100]
- **Type 3:** discrete uniform distribution on [50,100]
- **Type 4:** normal distribution with mean 100 and standard deviation 50
- **Type 5:** normal distribution with mean 100 and standard deviation 20
- **Perfect packing:** the processing times are generated from an interval $[1, Q]$ in such a way that the optimal schedule has equal completion time on each processor.

For each type and for each combination of n $(10 \leq n \leq 10000)$ and m $(2 \leq m \leq 15)$, 10 instances were generated. We observed that the values of our best proposed bounds coincide for 97.28% of the instances (3473 out of 3570). Interestingly, we found that, except in very few cases, those instances whose lower and upper bounds are not equal have less than 25 jobs. At this point, it is worth noting that we performed further experiments with the proposed heuristic U_{SS} on the set of perfect packing. We observed that all the instances were solved to optimality within a maximum CPU time of 0.34 seconds.

4.2. Performance on benchmark instances

Pushing our investigation a step further, we evaluated the performance of $\tilde{L}_{FS}$ and U_{MSS} on the benchmark instances proposed by França et al. [6], which include 390 uniform distribution instances, and those proposed by Frangioni et al. [7], which includes 390 non uniform distribution instances. For these instances, the best bounds which have been obtained so far are those provided by Alvim and Ribeiro [1]. We found that $\tilde{L}_{FS}$ outperforms L_{AR} for 14 out of 780 instances, and is equal to L_{AR} for the remaining instances. Also, U_{MSS} provides the best upper bound for 757 out of 780 instances, and improves U_{AR} for 15 instances. Moreover, if we restrict our attention to the 45 hardest benchmark instances (i.e. those for which $L_{AR} < U_{AR}$), we observe that the proposed procedures were able to reduce the gap for 29 instances. Furthermore, proven optimal solutions were delivered for 18 instances. In Tables 1 and 2, we provide the values of the newly improved bounds. Each instance is identified by its distribution (uniform U, or non-uniform N), its interval range $[1, k]$ $(k \in \{100, 1000, 10000\})$, its number of jobs n, its number of machines m, and its instance number. For example, the fourth 50 job-10 machine instance which data are non-uniformly distributed on the interval [1,100] is identified by $(N, 100, 50, 10, 4)$.

In summary, we run our best lower and upper bounds on a total of 4930 instances, and we found that for 4630 instances (93.91%), these two bounds yield proven optimal values. Table 3 provides, for each problem class, the percentage of instances for which $\tilde{L}_{FS}$ and U_{MSS} are equal.

Tab. 1: New improved lower bounds of benchmark instances

Instance	L_{AR}	$\tilde{L}_{FS}$
$(U, 10000, 10, 5, 5)$	10788	10789
$(U, 10000, 50, 25, 2)$	9517	9659
$(N, 100, 50, 10, 2)$	466	472^*
$(N, 100, 50, 10, 4)$	468	475^*
$(N, 100, 50, 10, 5)$	466	471^*
$(N, 100, 50, 10, 7)$	470	476^*
$(N, 100, 50, 10, 8)$	466	472^*
$(N, 100, 50, 10, 9)$	465	471^*
$(N, 100, 100, 25, 5)$	375	378^*
$(N, 100, 100, 25, 6)$	373	375^*
$(N, 1000, 100, 25, 5)$	3750	3775^*
$(N, 1000, 100, 25, 6)$	3726	3751^*
$(N, 10000, 100, 25, 5)$	37501	37754^*
$(N, 10000, 100, 25, 6)$	37264	37511^*

(*) indicates that the bound is proven optimal

Tab. 2: New improved upper bounds of benchmark instances

Instance	U_{AR}	U_{MSS}
$(U, 1000, 50, 10, 6)$	2405	2404^*
$(U, 1000, 50, 10, 8)$	2797	2796^*
$(U, 1000, 100, 25, 1)$	2093	2092^*
$(U, 10000, 50, 10, 1)$	26665	26663
$(U, 10000, 50, 10, 5)$	27208	27207
$(U, 10000, 50, 10, 8)$	25484	25481
$(U, 10000, 50, 10, 9)$	32271	32270
$(U, 10000, 50, 10, 10)$	26710	26709
$(U, 10000, 100, 25, 3)$	21579	21578
$(U, 10000, 100, 25, 5)$	20580	20576
$(U, 10000, 100, 25, 6)$	20705	20704
$(U, 10000, 100, 25, 9)$	20606	20604
$(N, 100, 100, 10, 5)$	942	941^*
$(N, 100, 100, 10, 9)$	940	939^*
$(N, 10000, 50, 10, 7)$	47545	47544^*

(*) indicates that the bound is proven optimal

Tab. 3: Percentage of instances for which $\tilde{L}_{FS}$ and U_{MSS} are equal

Class 1		81.39
Class 2		59.55
Dell'Amico and Martello's instances	Type 1	97.07
	Type 2	94.39
	Type 3	97.80
	Type 4	93.66
	Type 5	96.59
	Perfect packing	99.14
Benchmark instances	Uniform	91.29
	Non-uniform	96.67

5. CONCLUSION

In this paper, we have shown that the lifting procedure makes it feasible to derive several new tight lower bounds for the $P||C_{\max}$. Our computational experiments provide strong evidence that the new lifted bounds consistently outperform the best bounds from the literature. Moreover, we have shown that the $P2||C_{\max}$ can be reformulated and efficiently solved as a subset-sum problem. This led to an approximate algorithm which was found to perform extremely well on a wide range of instances.

We expect that the new derived bounds would prove useful for the exact solution of hard $P||C_{\max}$ instances but this would require further investigation.

REFERENCES

[1] Alvim A.C.F., C.C. Ribeiro, ”A hybrid bin packing heuristic to multiprocessor scheduling”, In C.C. Ribeiro and S.L. Martins (editors), *Lecture Notes in Computer Science*, Vol 3059, pp1-13. Springer-Verlag, Berlin, 2004.

[2] Brucker P., J. Hurink, F. Werner, ”Improving local search heuristics for some scheduling problems. Part I”, *Discrete Applied Mathematics,* Vol 65, pp97-122, 1996.

[3] Brucker P., J. Hurink, F. Werner, ”Improving local search heuristics for some scheduling problems. Part II”, *Discrete Applied Mathematics,* Vol 72, pp47-69, 1997.

[4] Dell’Amico M., S. Martello, ”Optimal scheduling of tasks on identical parallel processors”, *ORSA Journal on Computing,* Vol 7, pp191-200, 1995.

[5] Fekete S., J. Schepers, ”New classes of fast lower bounds for bin packing problems”, *Mathematical Programming,* Vol 91, pp11-31, 2001.

[6] França P.M., M. Gendreau, G. Laporte, F.M. Müller, ”A composite heuristic for the identical parallel machine scheduling problem with minimum makespan objective”, *Computers & Operations Research,* Vol 21, pp205-210, 1994.

[7] Frangioni A., E. Necciari, M. G. Scutellà, ”A multi-exchange neighborhood for minimum makespan machine scheduling problems”, *Journal of Combinatorial Optimization,* Vol 8, pp195-220, 2004.

[8] Haouari M., A. Gharbi, ”Lower Bounds for Scheduling on Identical Parallel Machines with Heads and Tails”, *Annals of Operations Research,* Vol 129, pp187-204, 2004.

[9] Haouari M., A. Gharbi, ”Fast Lifting Procedures for the Bin Packing Problem”, *Discrete Optimization,* Vol 2, pp201-218, 2005.

[10] Ho J.C., J.S. Wong, ”Makespan minimization for m parallel identical processors”, *Naval Research Logistics,* Vol 42, pp935-948, 1995.

[11] Hübscher R., F. Glover, ”Applying Tabu Search with influential diversification to multiprocessor scheduling”, *Computers Operations Research,* Vol 21, pp877-884, 1994.

[12] Jozefowska J., M. Milka, R. Rozycki, G. Waligora, J. Weglarz, ”Local search metaheuristics for discrete-continuous problems”, *European Journal of Operational Research,* Vol 107, pp354-370, 1998.

[13] Martello S., P. Toth, ”Knapsack problems: Algorithms and computer implementations”, *John Wiley & Sons*, 1990.

[14] McNaughton R., ”Scheduling with deadlines and loss function”, *Management Science,* Vol 6, pp1-12, 1959.

[15] Mokotoff E., ”Parallel Machine Scheduling Problems: A Survey”, *Asia-Pacific Journal of Operations Research,* Vol 18, pp193-243, 2001.

[16] Mokotoff E., ”An Exact Algorithm for the Identical Parallel Machine Scheduling Problem”, *European Journal of Operational Research,* Vol 152, pp758-769, 2004.

[17] Pisinger D., ”Dynamic programming on the word RAM”, *Algorithmica,* Vol 35, pp437-459, 2003.

[18] Rothkopf M.H., ”Scheduling independent tasks on parallel processors”, *Management Science,* Vol 12, pp437-447, 1966.

[19] Sahni S., ”Algorithms for scheduling independents tasks”, *Journal of the Association for Computing Machinery,* Vol 23, pp116-127, 1976.

[20] Thesen A., ”Design and evaluation of tabu search algorithms for multiprocessor scheduling”, *Journal of Heuristics,* Vol 4, pp141-160, 1998.

[21] Webster S.T., ”A general lower bound for the makespan problem”, *European Journal of Operational Research,* Vol 89, pp516-524, 1996.