



START Project Y43-MAT
Combinatorial Approximation Algorithms
TU Graz, Austria

*Günter Rote
Gerhard J. Woeginger*

*Time Complexity and Linear-Time Approximation
of the Ancient Two Machine Flow Shop*

**Report Woe-14
November 1997**

Time complexity and linear-time approximation of the ancient two machine flow shop ^{*}

GÜNTER ROTE [†]

GERHARD J. WOEGINGER [‡]

Abstract

We consider the scheduling problems $F2 || C_{\max}$ and $F2 | no-wait | C_{\max}$, i.e. makespan minimization in a two machine flow shop, with and without no wait in process. For both problems solution algorithms based on sorting with $O(n \log n)$ running time are known, where n denotes the number of jobs [Johnson 1954, Gilmore & Gomory 1964].

We prove that no asymptotically faster algorithms can solve these problems. This is done by establishing $\Omega(n \log n)$ lower bounds in the algebraic computation tree model of computation. Moreover, we develop for every $\varepsilon > 0$ approximation algorithms with linear running time $O(n \log \frac{1}{\varepsilon})$ that deliver feasible schedules whose makespan is at most $1 + \varepsilon$ times the optimum makespan.

Keywords. Flow shop, no wait in progress, computational complexity, algebraic computation tree, lower bounds.

1 Introduction

In the two machine flow shop, there are two machines M_A and M_B , and n jobs J_1, \dots, J_n . Every job J_j ($j = 1, \dots, n$) needs to be processed first for a_j time units on machine M_A and afterwards for b_j time units on machine M_B . The goal is to minimize the makespan, i.e. the point in time at which the last job is completed. In the standard scheduling notation (cf. Lawler, Lenstra, Rinnooy Kan & Shmoys [7]) this problem is denoted by $F2 || C_{\max}$. In the closely related two machine *no wait flow shop*, $F2 | no-wait | C_{\max}$, each job, once started, has to be processed without interruption until it is completed.

The problems $F2 || C_{\max}$ and $F2 | no-wait | C_{\max}$ both are solvable in $O(n \log n)$ time. For $F2 || C_{\max}$, Johnson [6] shows that the following rule yields an optimum schedule: first process the jobs with $a_j \leq b_j$ in order of nondecreasing a_j , and then process the remaining jobs in order of nonincreasing b_j . $F2 | no-wait | C_{\max}$ can be formulated as a travelling salesman problem whose distance matrix assumes a special combinatorial structure. The results of Gilmore & Gomory [4] then yield an $O(n \log n)$ solution to this travelling salesman problem (cf. Reddi & Ramamoorthy [10], or Gilmore, Lawler & Shmoys [5]). The only expensive step of the

^{*}This research has been supported by the START program Y43-MAT of the Austrian Ministry of Science.

[†]`rrote@opt.math.tu-graz.ac.at`. Institut für Mathematik B, TU Graz, Steyrergasse 30, A-8010 Graz, Austria.

[‡]`gwoegi@opt.math.tu-graz.ac.at`. Institut für Mathematik B, TU Graz, Steyrergasse 30, A-8010 Graz, Austria.

Gilmore & Gomory algorithm is an $O(n \log n)$ sorting step, whereas the remaining steps take $O(n)$ overall time. Hence, the time complexity of both algorithms mainly comes from a sorting step. Naturally the question arises whether one can do faster, without using a sorting routine.

The main contribution of this paper is that every algorithm for problems $F2 || C_{\max}$ and $F2 | no-wait | C_{\max}$ must use $\Omega(n \log n)$ time in the worst case. This is proved by establishing $\Omega(n \log n)$ lower bounds in the algebraic computation tree model of computation, which is a standard model in theoretical computer science for studying the time complexity of computational problems. We also investigate the strength of *linear-time* algorithms for these problems. We show that for every $\varepsilon > 0$, the value of the optimum makespan in both problems can be approximated in linear time within an multiplicative error of $1 + \varepsilon$. These results are essentially based on replacing the sorting steps that cost $O(n \log n)$ time by partitioning steps that cost $O(n)$ time.

The paper is organized as follows. In Section 2 we briefly introduce the algebraic computation tree model, and we establish the $\Omega(n \log n)$ lower bounds on the time complexity of the scheduling problems. Section 3 gives the linear-time approximation algorithms for $F2 || C_{\max}$ and for $F2 | no-wait | C_{\max}$. Section 4 contains the conclusion.

2 Lower bounds in the algebraic computation tree model

In this section we prove that any algorithm for $F2 || C_{\max}$ and $F2 | no-wait | C_{\max}$, respectively, must have an $\Omega(n \log n)$ worst case time complexity in the algebraic computation tree model of computation.

An *algebraic computation tree* (ACT, for short) models a deterministic program in which each step performs either a binary arithmetic operation $z := x \circ y$ with $\circ \in \{+, -, *, /\}$, a unary arithmetic operation $z := \sqrt{x}$, or a branching operation depending on whether $z < 0$, $z = 0$ or $z > 0$. These operations are the *atomic* operations that can be performed within one time-unit; note that computing the floor-function is not an atomic operation. The ACT is one of the standard models of computation in theoretical computer science; cf. e.g. Ben-Or [1], Ramanathan [9], Seiferas [11], and Yao [12].

Definition 2.1 *An instance of the UNIFORM GAP problem consists of n real numbers x_1, \dots, x_n and a real number $\varepsilon > 0$. The problem is to decide whether all gaps between consecutive numbers are equal to ε . (Two numbers x_i and x_j are consecutive if $x_i \leq x_j$ and if there exists no other number x_k , $i \neq k \neq j$, such that $x_i \leq x_k \leq x_j$).*

Proposition 2.2 *(Lee & Wu [8])*

The UNIFORM GAP problem requires $\Omega(n \log n)$ worst case time in the algebraic computation tree model. ■

The lower bounds for problems $F2 || C_{\max}$ and $F2 | no-wait | C_{\max}$ are proved by reducing the UNIFORM GAP problem in linear time to the scheduling problems. Indeed, let x_1, \dots, x_n and ε constitute an instance of UNIFORM GAP. Without loss of generality we may assume that the smallest of the numbers x_i is zero. We determine in linear time the sum X of all values x_i . If $X \neq \frac{1}{2}(n-1)n\varepsilon$, the answer to the UNIFORM GAP instance must be NO. Otherwise, we

construct in linear time $O(n)$ an instance of the two machine flow shop: For $j = 1, \dots, n$ we introduce a job J_j with processing times $a_j = x_j$ on machine A and processing time $b_j = x_j + \varepsilon$ on machine B . We claim that this scheduling instance for $F2 || C_{\max}$ and $F2 | no-wait | C_{\max}$ has a feasible schedule with makespan at most $\frac{1}{2}n(n+1)\varepsilon$ if and only if the UNIFORM GAP instance has answer YES.

(If). In case the UNIFORM GAP instance has answer YES, there exists a permutation $\pi \in S_n$ of the jobs such that $a_{\pi(i)} = (i-1)\varepsilon$ and $b_{\pi(i)} = i\varepsilon$ for all $1 \leq i \leq n$. Then the schedule $(J_{\pi(1)}, J_{\pi(2)}, \dots, J_{\pi(n)})$ where for $i = 2, \dots, n$ the operations $a_{\pi(i)}$ and $b_{\pi(i-1)}$ always are processed in parallel is a feasible permutation schedule for $F2 || C_{\max}$ and for $F2 | no-wait | C_{\max}$. Moreover, this schedule has makespan $\frac{1}{2}n(n+1)\varepsilon$.

(Only if). Assume that a schedule exists with makespan at most $\frac{1}{2}n(n+1)\varepsilon$. Without loss of generality we may assume that this schedule is a permutation schedule in which both machines process the jobs in the same order $\pi \in S_n$. Then for every $1 \leq k \leq n$, the inequality

$$\sum_{i=1}^k a_{\pi(i)} + \sum_{i=k}^n b_{\pi(i)} \leq \frac{1}{2}n(n+1)\varepsilon \quad (1)$$

holds. By rewriting the left-hand side of this inequality, one derives

$$\sum_{i=1}^k a_{\pi(i)} + \sum_{i=k}^n (a_{\pi(i)} + \varepsilon) = X + a_{\pi(k)} + (n-k+1)\varepsilon \leq \frac{1}{2}n(n+1)\varepsilon, \quad (2)$$

which is equivalent to $a_{\pi(k)} \leq (k-1)\varepsilon$. Summing this over all $1 \leq k \leq n$ yields

$$X = \sum_{k=1}^n a_{\pi(k)} \leq \sum_{k=1}^n (k-1)\varepsilon = \frac{1}{2}n(n-1)\varepsilon = X. \quad (3)$$

This implies that for all $1 \leq k \leq n$, $a_{\pi(k)} = (k-1)\varepsilon$ must hold. Hence, the considered instance of UNIFORM GAP has answer YES.

Summarizing, any algorithm for $F2 || C_{\max}$ or $F2 | no-wait | C_{\max}$ with worst case running time $O(f(n))$ can be used to construct an $O(n+f(n))$ algorithm for UNIFORM GAP. Now the lower bound result of Lee & Wu [8] (cf. Proposition 2.2) yields that $f(n)$ must be $\Omega(n \log n)$.

Theorem 2.3 *Any algorithm for $F2 || C_{\max}$ and any algorithm for $F2 | no-wait | C_{\max}$ must have a worst case running time $\Omega(n \log n)$ in the algebraic computation tree model of computation.* ■

3 A linear-time approximation algorithm

In this section, our goal is to approximate the optimum makespan of a two machine flow shop (with or without no wait in process) in linear time. Let $\varepsilon > 0$ be some small fixed real number and define $g = \lceil \log_2(\frac{4}{\varepsilon}) \rceil$. We consider an instance I of $F2 || C_{\max}$ or $F2 | no-wait | C_{\max}$ that is specified by the processing times a_j and b_j ($j = 1, \dots, n$) of the n jobs. Let $C_{\max}^*(I)$ denote the optimum makespan of I . Hence, our goal is to find in $O(n g) = O(n \log \frac{1}{\varepsilon})$ time a feasible schedule for I with makespan at most $(1 + \varepsilon)C_{\max}^*(I)$.

Observation 3.1 *Without loss of generality we may assume that the number n of jobs is of the form $n = 2^g(2^m - 1)$ for some integer $m \geq 1$.*

Proof. If $n < 2^g$ holds, then we can solve the instance to optimality in $O(n \log n) = O(ng)$ time. If $n \geq 2^g$ holds and n is not of the form $2^g(2^m - 1)$, then we add an appropriate set of dummy jobs of length 0. By at most tripling the number of jobs, we get a number of jobs of the desired form. \blacksquare

Consider the n processing times a_1, \dots, a_n on the first machine and define the following partition $A_{i,k}$, $0 \leq i \leq m-1$ and $1 \leq k \leq 2^g$, of the numbers a_j :

- For $0 \leq i \leq m-1$ and $1 \leq k \leq 2^g$, the set $A_{i,k}$ contains 2^i elements.
- For $0 \leq i \leq m-2$ and $1 \leq k, \ell \leq 2^g$, all elements in $A_{i+1,k}$ are less or equal to all elements in $A_{i,\ell}$.
- For $0 \leq i \leq m-1$ and $1 \leq k < \ell \leq 2^g - 1$, all elements in $A_{i,k+1}$ are less or equal to all elements in $A_{i,k}$.

Moreover, we define a similar partition $B_{i,k}$, $0 \leq i \leq m-1$ and $1 \leq k \leq 2^g$, of the numbers b_1, \dots, b_n . For $0 \leq i \leq m-1$ and $1 \leq k \leq 2^g$, denote by $\alpha_{i,k}$ the maximum value in $A_{i,k}$ and denote by $\beta_{i,k}$ the maximum value in $B_{i,k}$. Finally, we define a new instance I' of $F2 || C_{\max}$, respectively $F2 | no-wait | C_{\max}$, as follows: For every job J_j in I with processing times $a_j \in A_{i,k}$ and $b_j \in B_{r,s}$ the instance I' contains a corresponding job J'_j with processing times $a'_j = \alpha_{i,k}$ and $b'_j = \beta_{r,s}$.

Since for all j ($j = 1, \dots, n$) $a_j \leq a'_j$ and $b_j \leq b'_j$ holds, the optimum makespan $C_{\max}^*(I')$ of instance I' must be greater or equal to $C_{\max}^*(I)$. Conversely, consider an optimum schedule for I in which each machine processes the jobs in the same order $\pi \in S_n$. If we use π to process the corresponding jobs in I' , the increase in the makespan is upper bounded by the total increase in the processing times. Summarizing, we have

$$C_{\max}^*(I) \leq C_{\max}^*(I') \leq C_{\max}^*(I) + \sum_{j=1}^n (a'_j - a_j) + \sum_{j=1}^n (b'_j - b_j). \quad (4)$$

Observe the following statements for the smallest value in set $A_{i,k}$.

- For $1 \leq k \leq 2^g$, the smallest value in $A_{0,k}$ is equal to its only element $\alpha_{0,k}$.
- For $1 \leq i \leq m-1$ and $1 \leq k \leq 2^g - 1$, the smallest value in $A_{i,k}$ is at least $\alpha_{i,k+1}$.
- If $1 \leq i \leq m-1$, then the smallest value in $A_{i,2^g}$ is at least $\alpha_{i+1,1}$. (Define $\alpha_{m,1} = 0$).
- In $A_{m-1,2^g}$, the smallest value is at least 0.

By using these bounds on a_j and by summing over all sets $A_{i,k}$, one gets that

$$\sum_{j=1}^n a_j \geq \sum_{k=1}^{2^g} \alpha_{0,k} + \sum_{i=1}^{m-1} \sum_{k=1}^{2^g-1} 2^i \alpha_{i,k+1} + \sum_{i=1}^{m-1} 2^i \alpha_{i+1,1}. \quad (5)$$

By using $\alpha_{0,k} \geq \alpha_{1,1}$ together with $\alpha_{i,k+1} \geq \alpha_{i+1,1}$ for $1 \leq i \leq m-2$ and $1 \leq k \leq 2^g - 1$, and together with $\alpha_{m-1,k+1} \geq 0$ for $1 \leq k \leq 2^g - 1$, inequality (5) leads to

$$\sum_{j=1}^n a_j \geq 2^g \alpha_{1,1} + \sum_{i=1}^{m-2} \sum_{k=1}^{2^g-1} 2^i \alpha_{i+1,1} + \sum_{i=1}^{m-2} 2^i \alpha_{i+1,1} = 2^g \sum_{i=1}^{m-1} 2^{i-1} \alpha_{i,1}. \quad (6)$$

Moreover,

$$\sum_{j=1}^n a'_j = \sum_{k=1}^{2^g} \alpha_{0,k} + \sum_{i=1}^{m-1} \sum_{k=1}^{2^g-1} 2^i \alpha_{i,k} + \sum_{i=1}^{m-1} 2^i \alpha_{i,2^g}. \quad (7)$$

By subtracting (5) from (7) and by telescoping several sums, we get

$$\begin{aligned} \sum_{j=1}^n (a'_j - a_j) &\leq \sum_{i=1}^{m-1} \sum_{k=1}^{2^g-1} 2^i (\alpha_{i,k} - \alpha_{i,k+1}) + \sum_{i=1}^{m-1} 2^i (\alpha_{i,2^g} - \alpha_{i+1,1}) \\ &= \sum_{i=1}^{m-1} 2^i (\alpha_{i,1} - \alpha_{i,2^g}) + \sum_{i=1}^{m-1} 2^i (\alpha_{i,2^g} - \alpha_{i+1,1}) \\ &= \sum_{i=1}^{m-1} 2^i (\alpha_{i,1} - \alpha_{i+1,1}) \\ &= \alpha_{1,1} + \sum_{i=1}^{m-1} 2^{i-1} \alpha_{i,1}. \end{aligned} \quad (8)$$

Finally, since the optimum makespan is greater or equal to the total processing time on the first machine, we get from (6) and (8) that

$$\begin{aligned} C_{\max}^*(I) &\geq \sum_{j=1}^n a_j \geq 2^g \sum_{i=1}^{m-1} 2^{i-1} \alpha_{i,1} \\ &\geq 2^{g-1} (\alpha_{1,1} + \sum_{i=1}^{m-1} 2^{i-1} \alpha_{i,1}) \geq 2^{g-1} \sum_{j=1}^n (a'_j - a_j) \end{aligned} \quad (9)$$

Now (9) together with the definition of g yields that

$$\sum_{j=1}^n (a'_j - a_j) \leq \frac{\varepsilon}{2} \cdot C_{\max}^*(I). \quad (10)$$

Analogous arguments yield

$$\sum_{j=1}^n (b'_j - b_j) \leq \frac{\varepsilon}{2} \cdot C_{\max}^*(I). \quad (11)$$

Finally, we combine (10) and (11) with (4), and thus get

$$C_{\max}^*(I') \leq (1 + \varepsilon) C_{\max}^*(I). \quad (12)$$

Lemma 3.2 *From instance I , one can compute instance I' in O/ng time.*

Proof. Blum, Floyd, Pratt, Rivest & Tarjan [2] show that the r -largest value among s numbers can be determined in $O(s)$ time. We use this algorithm to compute all sets $A_{i,k}$ and all sets $B_{i,k}$ in O/ng time.

In a first step, we compute m auxiliary sets $A_{m-1}, A_{m-2}, \dots, A_0$. For $0 \leq i \leq m-1$, $|A_i| = 2^{g+i}$ holds; all elements in A_{i+1} are less or equal to all elements in A_i . Set A_{m-1} is computed as follows: Determine the median (i.e. the $\frac{n}{2}$ -largest element) of the values a_1, \dots, a_n . Remove $\frac{n}{2} = 2^{g+m-1}$ elements that are less or equal to the median, and put them into A_{m-1} . From the remaining elements, successively compute A_{m-2} , A_{m-3} , and so on. Since the computation of A_i costs $O(2^{g+i})$ time, this yields an overall time complexity of $O(n)$ for the first step.

In the second phase, every set A_i is split into the sets $A_{i,k}$ with $1 \leq k \leq 2^g$. We can split A_i by repeatedly halving it into smaller and smaller pieces. In the ℓ -th iteration, we call the algorithm of Blum & al. for $2^{\ell-1}$ sets with $2^{g+i-\ell+1}$ elements each, which takes $O(|A_i|)$ time. After g iterations, we are done. Hence, the overall time complexity for splitting A_i is $O(g|A_i|)$, and the overall time complexity for splitting all sets is O/ng .

Analogously, we compute all sets $B_{i,k}$ in O/ng time. It is straightforward to determine the values $\alpha_{i,k}$ and $\beta_{i,k}$, and the values a'_j and b'_j in linear time. Hence, instance I' can be computed in O/ng time from instance I . ■

Lemma 3.3 *For any instance I of $F2 || C_{\max}$ or $F2 | no-wait | C_{\max}$, one can compute an optimum solution to the corresponding instance I' in $O(n)$ time.*

Proof. First consider the case where I is an instance of $F2 || C_{\max}$. Johnson's algorithm [6] mainly relies on sorting the jobs according to their processing times. Since we already know the partitions $A_{i,k}$ and $B_{i,k}$, the sorting of the jobs in I' can easily be done in $O(n)$ time.

For the case where I is an instance of $F2 | no-wait | C_{\max}$, we refer the reader to the work of Gilmore & Gomory [4] (see also Gilmore, Lawler & Shmoys [5]). The first step of the Gilmore & Gomory algorithm consists in sorting the values a'_j and in sorting the values b'_j . Similarly as above, we observe that this sorting can be done in $O(n)$ time. The remaining steps of the Gilmore & Gomory algorithm – i.e. computing an optimum assignment, finding the subtours, computing a patching tree of minimum cost for the subtours, and performing the subtour patching – all can be done in $O(n)$ time. We stress the fact that in the Gilmore & Gomory case *any* spanning tree for the subtours is an optimum patching tree (cf. Section 5.3 in Burkard, Deineko, van Dal, van der Veen & Woeginger [3]). ■

Theorem 3.4 *For each real $\varepsilon > 0$ and for each n job instance I of $F2 || C_{\max}$, respectively $F2 | no-wait | C_{\max}$, one can compute in $O(n \log \frac{1}{\varepsilon})$ time a feasible schedule whose makespan is at most $1 + \varepsilon$ times the optimum makespan.*

Proof. Compute instance I' as described in Lemma 3.2, and solve it as described in Lemma 3.3. Let $J'_{\pi(1)}, \dots, J'_{\pi(n)}$ denote the job ordering in the optimum schedule for I' . Then by (12), the job ordering $J_{\pi(1)}, \dots, J_{\pi(n)}$ gives a schedule for instance I with makespan at most $(1 + \varepsilon)C_{\max}^*(I)$. ■

4 Conclusion

In this paper we have shown that any algorithm for the two machine flow shop problems $F2||C_{\max}$ and $F2|no-wait|C_{\max}$ needs $\Omega(n \log n)$ time in the algebraic computation tree model of computation. We remark that this does not exclude the existence of linear-time algorithms which use the floor-function, indirect addressing, or similar tricks (In fact, the UNIFORM GAP problem can be solved in linear time by bucket sort, using the floor-function). Moreover, we have shown that the optimum solution of these problems can be approximated in linear time with arbitrarily good precision.

Finally, we remark that also the problem $1||L_{\max}$ of minimizing the maximum lateness, the problem $1||T_{\max}$ of minimizing the maximum tardiness, and the problem $1||\sum U_j$ of minimizing the number of tardy jobs have $\Omega(n \log n)$ lower bounds in the algebraic computation tree model of computation. These three problems possess well-known $O(n \log n)$ algorithms (cf. Lawler, Lenstra, Rinnooy Kan & Shmoys [7]).

Acknowledgement. We thank Walter Bucher, Bettina Klinz, and Ulrich Pferschy for valuable discussions.

References

- [1] M. BEN-OR, Lower bounds for algebraic computation trees, *Proceedings of the 15th Annual ACM Symposium on Theory of Computing*, 1983, 80–86.
- [2] M. BLUM, R.W. FLOYD, V.R. PRATT, R.L. RIVEST, AND R.E. TARJAN, Time bounds for selection, *J. Computer and System Sciences* **7**, 1972, 448–461.
- [3] R.E. BURKARD, V.G. DEINEKO, R. VAN DAL, J.A.A. VAN DER VEEN, AND G.J. WOEINGER, Well-solvable special cases of the TSP: a survey, *SIAM Reviews*, to appear. (Also available as: SFB Report 52, 1995, Institut für Mathematik, TU Graz, Austria, at ftp.tu-graz.ac.at/pub/papers/math/).
- [4] P.C. GILMORE AND R.E. GOMORY, Sequencing a one state-variable machine: a solvable case of the travelling salesman problem, *Operations Research* **12**, 1964, 655–679.
- [5] P.C. GILMORE, E.L. LAWLER, AND D.B. SHMOYS, Well-solved special cases, in: E.L. Lawler, J.K. Lenstra, A.H.G. Rinnooy Kan, and D.B. Shmoys (eds.) *The Traveling Salesman Problem*, John Wiley, Chichester, 1985, 87–143.
- [6] S.M. JOHNSON, Optimal two- and three-stage production schedules with setup times included, *Naval Res. Logist. Quart.* **1**, 1954, 61–68.
- [7] E.L. LAWLER, J.K. LENSTRA, A.H.G. RINNOOY KAN, AND D.B. SHMOYS, Sequencing and scheduling: Algorithms and complexity, in: S.C. Graves, A.H.G. Rinnooy Kan, and P.H. Zipkin (eds.) *Logistics of Production and Inventory*, Handbooks in Operations Research and Management Science 4, North-Holland, Amsterdam, 1993, 445–522.

-
- [8] D.T. LEE AND Y.F. WU, Geometric complexity of some location problems, *Algorithmica* **1**, 1986, 193–211.
 - [9] P. RAMANAN, Obtaining lower bounds using artificial components, *Information Processing Letters* **24**, 1987, 243–246.
 - [10] S.S. REDDI AND C.V. RAMAMOORTHY, On the flow-shop sequencing problem with no wait in process, *Operations Research Quarterly* **23**, 1972, 323–331.
 - [11] J. SEIFERAS, A variant of Ben-Or’s lower bound for algebraic decision trees, *Information Processing Letters* **26**, 1988, 273–276.
 - [12] A.C.C. YAO, Lower bounds for algebraic computation trees with integer inputs, *SIAM J. Computing* **20**, 1991, 655–668.

PostScript and Dvi-files of the listed reports can be downloaded by anonymous ftp from `ftp.tu-graz.ac.at`, directory `/pub/papers/math`.

- SFB90 R.E. BURKARD AND J. KRARUP, A Linear Algorithm for the Pos/Neg-Weighted 1-Median Problem on a Cactus (Dec 96)
- SFB91 T. DUDÁS AND R. RUDOLF, Spanning Trees and Shortest Paths in Monge Graphs (Dec 96)
- SFB97 B. KLINZ AND G.J. WOEGINGER, A Note on the Bottleneck Graph Partition Problem (Dec 96)
- SFB100 V.G. DEĚNEKO, On the TSP with a Relaxed General Distribution Matrix (Jan 97)
- SFB105 R.E. BURKARD, Efficiently Solvable Special Cases of Hard Combinatorial Optimization Problems (Feb 97)
- SFB106 T. DUDÁS, B. KLINZ AND G.J. WOEGINGER, The Computational Complexity of the k -Minimum Spanning Tree Problem in Graded Matrices (Feb 97)
- Woe-01 P. SCHURMAN AND G.J. WOEGINGER, A Polynomial Time Approximation Scheme for the Two-Stage Multiprocessor Flow Shop Problem (Mar 97)
- Woe-02 N. ALON, Y. AZAR, G.J. WOEGINGER AND T. YADID, Approximation Schemes for Scheduling (Apr 97)
- Woe-03 L. BABEL AND G.J. WOEGINGER, Pseudo-Hamiltonian Graphs (May 97)
- Woe-04 A. FIAT AND G.J. WOEGINGER, On-line Scheduling on a Single Machine: Minimizing the Total Completion Time (Jun 97)
- SFB113 R.E. BURKARD, V.M. DEMIDENKO AND R. RUDOLF, A General Approach for Identifying Special Cases of the Travelling Salesman Problem with a Fixed Optimal Tour (Jun 97)
- Woe-05 V.G. DEĚNEKO AND G.J. WOEGINGER, A Study of Exponential Neighborhoods for the Traveling Salesman Problem and for the Quadratic Assignment Problem (Jul 97)
- SFB115 R.E. BURKARD AND V.G. DEĚNEKO, On the Traveling Salesman Problem with a Relaxed Monge Matrix (Jul 97)
- Woe-06 G.J. WOEGINGER, There is no Asymptotic PTAS for Two-Dimensional Vector Packing (Aug 97)
- Woe-07 G.J. WOEGINGER, A Comment on a Minmax Location Problem (Aug 97)
- SFB116 R.E. BURKARD AND Y. HE, A Note on Multifit Scheduling for Uniform Machines (Aug 97)
- SFB118 R.E. BURKARD, Y. HE AND H. KELLERER, A Linear Compound Algorithm for Uniform Machine Scheduling (Sep 97)
- Woe-08 S.S. SEIDEN, Randomized Preemptive Online Interval Scheduling (Sep 97)
- Woe-09 V.G. DEĚNEKO AND G.J. WOEGINGER, The Maximum Travelling Salesman Problem on Demidenko Matrices (Sep 97)
- Woe-10 G.J. WOEGINGER, A Note on the Depth Function of Combinatorial Optimization Problems (Sep 97)
- Woe-11 S.S. SEIDEN, Randomized Online Multi-threaded Paging (Oct 97)
- Woe-12 P. SCHURMAN AND G.J. WOEGINGER, A Polynomial Time Approximation Scheme for Preemptive Scheduling with Setup Times (Oct 97)
- Woe-13 P. SCHURMAN AND G.J. WOEGINGER, Approximation Algorithms for the Multiprocessor Open Shop Problem (Oct 97)
- SFB119 R.E. BURKARD, M. KOCHER AND R. RUDOLF, Rounding Strategies for Mixed Integer Programs Arising from Chemical Production Planning (Nov 97)
- SFB120 S.E. KARISCH, E. ÇELA, J. CLAUSEN AND T. ESPERSEN, A Dual Framework for Lower Bounds of the Quadratic Assignment Problem Based on Linearization (Nov 97)
- Woe-14 G. ROTE AND G.J. WOEGINGER, Time Complexity and Linear-Time Approximation of the Ancient Two Machine Flow Shop (Nov 97)
- Woe-15 H. HOOGEVEEN, P. SCHURMAN AND G.J. WOEGINGER, Non-Approximability Results for Scheduling Problems with Minsum Criteria (Nov 97)