Makespan minimization in job shops: a polynomial time approximation scheme

Klaus Jansen *
IDSIA Lugano
Corso Elvezia 36
6900 Lugano
Switzerland
klaus@idsia.ch

Roberto Solis-Oba †
MPII Saarbrücken
Im Stadtwald
66123 Saarbrücken
Germany
solis@mpi-sb.mpg.de

Maxim Sviridenko
Sobolev Inst. of Mathematics
pr. Koptyuga 4
630090 Novosibirsk
Russia
svir@math.nsc.ru

April 5, 2001

Abstract

In this paper we present a polynomial time approximation scheme for the job shop scheduling problem with fixed number of machines and fixed number of operations per job. The polynomial time approximation scheme can be extended to the case of job shop problems with release and delivery times, multiprocessor job shops, and dag job shops.

1 Introduction

In the job shop scheduling problem there is a set $J = \{J_1, \ldots, J_n\}$ of n jobs that must be processed on a given set $M = \{M_1, \ldots, M_m\}$ of m machines. Each job J_j consists of a sequence of μ_j operations $O_{1j}, \ldots, O_{\mu_j j}$ that need to be processed in this order. Operation O_{ij} must be processed without interruption on machine $M_{t_{ij}}$, during p_{ij} time units. Each machine can process at most one operation at a time, and each job may be processed by at most one machine at any time. For a given schedule, let C_{ij} be the completion time of operation O_{ij}. The objective is to find a schedule that minimizes the maximum completion time, $C_{max} = \max_{ij} C_{ij}$. The value of C_{max} is also called the *makespan* or the *length* of the schedule.

For a given instance of the job shop scheduling problem, the value of the optimum makespan will be denoted as C^*_{max}. Let $P_t = \sum_{t_{ij}=t} p_{ij}$ be the total processing time of operations assigned to machine M_t. We call P_t the *load* of machine M_t. Let $P = \max\{P_1, \ldots, P_m\}$, be the maximum machine load. Clearly, $P \leq C^*_{max}$. Let $l_j = \sum_{i=1}^{\mu_j} p_{ij}$ be the length of job J_j and let $\mu = \max_j \mu_j$ be the maximum number of operations in any job. We define $p_{max} = \max_{i,j} p_{ij}$ to be the maximum operation length.

The job shop scheduling problem is considered to be one of the most difficult problems in combinatorial optimization, both from the theoretical and the practical points of view. Even very constrained versions of the problem are strongly NP-hard (see e.g. the survey paper by Lawler et al. [6]). Two widely studied shop scheduling problems are the flow shop and the open shop problems. In the flow shop problem every job has exactly one operation per machine, and the order of execution for the operations is the same for all jobs. In the open shop problem every job has also one operation per machine, but there is no specified order for the execution of the operations of a job. Williamson et al. [16] proved that for any $\rho < 5/4$, the existence of a ρ-approximation algorithm for any of the above shop scheduling problems when the number of machines is part of the input would imply that $P = NP$.

During the last few years, many papers have been written about shop scheduling problems. Many of these works are based on the seminal work by Leighton, Maggs and Rao [7], who studied the acyclic job shop problem, in which every job has exactly one unit length operation per machine. Their main result was to

*This work was supported in part by the Swiss Office Fédéral de l'éducation et de la Science project n 97.0315 titled "Platform"

†This work was supported in part by EU ESPRIT LTR Project No. 20244 (ALCOM-IT)

show that this problem always has a solution of length $O(P + l_{max})$, where $l_{max} = \max_j l_j$ is the maximum job length. This is not an algorithmic result since it relies on a non-constructive probabilistic argument (for a constructive version see [8]).

Shmoys, Stein and Wein [15] described an approximation algorithm for the general job shop problem. The performance guarantee of this algorithm is $O(\log^2(m\mu))$. This algorithm was later slightly improved by Goldberg, Paterson, Srinivasan and Sweedyk [1] who designed an approximation algorithm with performance guarantee of $O(\log(m\mu)\log(min(m\mu, p_{max}))/(\log\log(m\mu))^2)$. An approximation algorithm with performance ratio of $(2+\varepsilon)$, for any fixed value $\varepsilon > 0$, for the job shop problem with fixed m and μ was also proposed by Shmoys, Stein and Wein [15].

A polynomial time approximation scheme (PTAS) for a (minimization) optimization problem P is an algorithm that given any constant value $\varepsilon > 0$ finds in polynomial time a solution of value no larger than $1+\varepsilon$ times the value of an optimum solution. A fully polynomial time approximation scheme is an approximation scheme that runs in time polynomial in the size of the input and $1/\varepsilon$.

When the number m of machines is fixed, there exist polynomial time approximation schemes for the flow shop [2] and the open shop [14] problems. But the $2+\varepsilon$ approximation algorithm of Shmoys et al. [15] was the previously best known algorithm for the job shop problem with m and μ fixed.

In this work we describe a polynomial time approximation scheme for the job shop scheduling problem with fixed m and μ. Our scheme can be used to design polynomial time approximation schemes for more general problems like the so-called dag shop problem [11, 12, 15], in which the order for processing the operations of a job must be consistent with a given partial order. This problem includes as a special case the open shop problem. Since the flow shop problem is a special case of the job shop problem, our result generalizes the results of Hall [2], Sevastianov and Woeginger [14] in the sense that we prove the existence of polynomial time approximation schemes for these two latter problems also. The running time of our algorithm is not polynomial in $1/\varepsilon$, but this is not surprising since the problem is strongly NP-hard [6] and therefore no fully polynomial time approximation scheme for the problem can exist unless P=NP.

Our work is strongly based on ideas contained in some of the aforementioned papers. We use the idea by Sevastianov and Woeginger [14] of dividing the set of jobs into three subsets of big, small and tiny jobs. Then we adopt an idea of Hall [2] of using a linear program for assigning tiny jobs to some predefined time intervals. To find a feasible schedule for these jobs we use an algorithm by Sevastianov [11], which runs in $O((\mu mn)^2)$ time and for any instance of the job shop problem it finds a schedule of length at most $P + \varphi(m,\mu)p_{max}$, where $\varphi(m,\mu) = (m\mu^2 + 2\mu - 1)(\mu - 1) = O(m\mu^3)$. (See also [12] for survey and historical overview of geometric methods used in the design and analysis of approximation algorithms with absolute performance guarantee for scheduling problems.)

Our polynomial time approximation scheme uses the notion of an *outline scheme* developed by Hall and Shmoys [3, 4, 5] and Hall [2]. Hall [2] gave the following definition of an outline scheme:

> An outline scheme is a partition of all feasible schedules into sets, whereby the schedules are grouped together into a set according to some characteristics that they share. The goal is that two things are simultaneously possible: first, that the partition has at most a polynomial number of sets in it; and second, that, given characteristics describing a set of the partition, it is possible to generate a schedule in polynomial time that is nearly as good as any schedule in that set. The outline scheme then suggests a natural way to obtain a good schedule: simply generate, for each possible set in the partition, a schedule that is "good" relative to that set. Since the optimal schedule must be contained in one of the sets, we are guaranteed to find a schedule that is "nearly" optimal, among all of the schedules generated.

2 Algorithm and its Analysis

2.1 Big, Small, and Tiny Jobs

Let ε be an arbitrary small rational number in the interval $(0, 1)$ such that $1/\varepsilon$ is an integer. Let $m \geq 2$ and $\mu \geq 1$ be the number of machines and maximum number of operations per job, respectively. We assume that the values of ε, μ and m are fixed and not part of the input. Following [14], we partition the set of jobs

into three subsets as follows. Let α be a real number such that

$$\varepsilon^{m/\varepsilon} \leq \alpha \leq \varepsilon.$$

We define three sets of jobs:

$$B = \{J_j | l_j \geq \alpha P\},$$
$$S = \{J_j | \alpha\varepsilon P < l_j < \alpha P\}, \text{ and}$$
$$T = \{J_j | l_j \leq \alpha\varepsilon P\}.$$

The jobs in B are called *big jobs* (or B-jobs), the jobs in S are called *small jobs* (or S-jobs) and the jobs in T are called the *tiny jobs* (or T-jobs). For the operations of big, small and tiny jobs we will use the same notation: the operations of big jobs will be called B-operations, while operations of small and tiny jobs will be called S-operations and T-operations, independently on their actual sizes. The number of big jobs is at most $mP/(\alpha P)$, and thus the size of B is bounded by a constant depending only on ε and m:

$$|B| \leq m/\alpha \leq m\varepsilon^{-m/\varepsilon}.$$

Sevastianov and Woeginger [14] showed that we can choose the number α so that

$$\sum_{J_j \in S} l_j \leq \varepsilon P. \tag{1}$$

This was done as follows. Define a sequence of numbers α_l, where l is a nonnegative integer, by $\alpha_l = \varepsilon^l$ and consider the sets S_l of small jobs with respect to α_l. Note that two sets S_i and S_j are disjoint for $i \neq j$. Since the total length of all jobs is at most mP, then there exists a value $k \leq m/\varepsilon$ for which S_k satisfies inequality (1). We set $\alpha = \alpha_k$.

With this choice of α we can bound the number of small jobs by a constant depending only on m and ε:

$$|S| \leq \frac{\varepsilon P}{\varepsilon\varepsilon^k P} = \varepsilon^{-k} \leq \varepsilon^{-m/\varepsilon}.$$

It is not difficult to see that mP is an upper bound on the length of an optimal schedule. We partition the time interval from 0 to $m(1+\mu\varepsilon)P$ into $\lceil m(1+\mu\varepsilon)/(\alpha\varepsilon)\rceil$ equal intervals of length at most $\alpha\varepsilon P$. These intervals are called *intervals of the first type.*

2.2 Restricted Job Shop Problem

We will consider only such schedules in which every B - operation starts its processing at the beginning of some interval of the first type. It is easy to see that such a restriction does not increase the length of the optimal schedule considerably. Indeed, let us consider the first B-operation in an optimal schedule that does not start at the beginning of some interval of the first type. We can simply shift this B-operation to the right, so that it starts at the beginning of the next interval. All operations starting after this B-operation are shifted to the right by the same length. The overall increase in the length of the optimal schedule is bounded by $\mu|B|\alpha\varepsilon P \leq \mu m\varepsilon P \leq \mu m\varepsilon C^*_{max}$. Let $\widetilde{C}^*_{max}$ be the length of an optimal schedule in which the B-operations start at the beginning of some interval of the first type. As was noted above $\widetilde{C}^*_{max} \leq (1+\mu m\varepsilon)C^*_{max}$.

In the rest of this section we consider only the restricted job shop scheduling problem in which every B-operation must start at the beginning of some interval of the first type.

Let us fix some feasible schedule for the B-operations within the time interval $[0, m(1+\mu\varepsilon)P]$. Let S_{ij} and C_{ij} be the starting and finishing times of B-operation O_{ij}, respectively. Let $A = \{a_k | a_k = S_{ij} \text{ or } a_k = C_{ij} \text{ for some B-operation } O_{ij}\}$, be the set of starting and completion times of B-operations. Notice that $|A| \leq 2\mu|B| \leq 2\mu m/\alpha$. Assume that the elements in A are indexed so that $a_1 \leq a_2 \leq \ldots \leq a_{|A|}$. We define two elements $a_0 = 0$ and $a_{|A|+1} = C \in [a_{|A|}, (1+2\mu\varepsilon)mP]$ (the exact value of C will be specified latter), and partition the time interval from 0 to C into $|A|+1$ intervals $[a_k, a_{k+1}), k = 0, \ldots, |A|$. We will call this intervals *the intervals of the second type.* Let Δ_k be the length of the kth interval, i.e. $\Delta_k = a_{k+1} - a_k$.

For each S-operation we fix the interval of the second type in which that operation starts its processing. By doing this we have effectively defined an outline scheme. Since the number of big and small operations is

constant and since the number of intervals of both types is constant, we deduce that the number of possible groups of schedules (outlines) with fixed starting times of B-operations and fixed starting intervals for S-operations is constant, too. For each such outline we assign the starting times for the T-operations within some interval of the second type by solving a linear program as described below.

2.3 Scheduling the T-Operations

Let S^{tk} be the sets of S-operations with starting times scheduled on the kth interval (of the second type) on machine M_t. For every $J_j \in T$ let

$$\begin{aligned} K_j &= \{K = (k_1, k_2, \ldots, k_{\mu_j}) \in Z^{\mu_j} \\ &\qquad \mid 0 \le k_1 \le k_2 \le \ldots \le k_{\mu_j} \le |A|\} \end{aligned}$$

be the set of all feasible assignments of starting times of operations of job J_j to intervals of the second type. A tuple $(k_1, k_2, \ldots, k_{\mu_j}) \in K_j$ means that the ith operation of job J_j, $1 \le i \le \mu_j$, is started within interval (of second type) k_i.

We note that for each tuple $(k_1, k_2, \ldots, k_{\mu_j}) \in K_j$ no B-operation is scheduled on any machine $M_{t_{ij}}$ needed by some operation O_{ij}, $1 \le i \le \mu_j$, of job $J_j \in T$. In other words, tuples K_j allow schedules for the operations of T-job J_j only on those intervals in which the machines needed by those operations do not process any B-operation.

Now we construct a linear program with decision variables

$$x_{j,K}, \quad K \in K_j, \quad J_j \in T$$

where $x_{j,K} = 1$ for $K = (k_1, k_2, \ldots, k_{\mu_j})$ means that the first operation of job J_j is started in interval k_1 on machine $M_{t_{1j}}$, the second operation is started in interval k_2 on $M_{t_{2j}}$ and so on. Let

$$\Delta'_{tk} = \Delta_k - \sum_{O_{ij} \in S^{tk}} p_{ij} + d(k)$$

where

$$d(k) = \max\{\alpha\varepsilon P, \max_{O_{ij} \in \cup_{t=1}^{m} S^{tk}} p_{ij}\}.$$

So Δ'_{tk} is an upper bound on the free time of machine M_t in interval k that remains after scheduling S-operations. Note that we have expanded the length of the kth interval by $d(k)$ to ensure that all S-operations and T-operations assigned to interval k fit in it (observe that some small or tiny operation may start processing right at the very end of the interval). To find a schedule for the jobs $J_j \in T$ we first find a basic feasible solution to the following linear program (we assume that we have already chosen the value of the length C of the schedule, so we are interested only in knowing whether the jobs can be scheduled within the time interval $[0, C]$; we show below how to choose C):

$$\sum_{K \in K_j} x_{j,K} = 1, \quad J_j \in T, \tag{2}$$

$$\begin{aligned} \sum_{J_j \in T} \sum_{K \in K_j} P_{j,K,t,k} \cdot x_{j,K} \le \Delta'_{tk}, \quad t &= 1, \ldots, m, \\ k &= 0, \ldots, |A| \end{aligned} \tag{3}$$

$$x_{j,K} \ge 0, \quad K \in K_j, J_j \in T \tag{4}$$

where $P_{j,K,t,k}$ is the total processing time of all operations O_{ij} of job $J_j \in T$ such that $M_{t_{ij}} = M_t$ and $k_i = k$. Less formally, constraint (3) means that the total length of operations assigned to interval k on machine M_t does not exceed the length of the augmented interval.

Recall that $a_{|A|+1} = C \in [a_{|A|}, (1+2\mu\varepsilon)mP]$. Let C_{min} be the smallest value C such that linear program (2)-(4) has a feasible solution for some outline. Since for $C = \widetilde{C}^*_{max}$ we can construct such outline from an

optimal solution of the constrained job shop problem, we conclude that $C_{min} \leq \widetilde{C}^*_{max}$. For any fixed value $\delta \geq 0$, a value of C satisfying: $C_{min} \leq C \leq C_{min} + \delta P$, can be found in polynomial time by binary search. Since C_{min} is a lower bound on $\widetilde{C}^*_{max}$, then $C \leq \widetilde{C}^*_{max} + \delta P \leq C^*_{max} + 2\mu m\varepsilon P$, for $\delta = \mu m\varepsilon$.

The linear program (2)-(4) has $|T| + m(|A|+1)$ constraints and $|T|(|A|+1)^\mu$ variables. Therefore, a basic feasible solution of this program is guaranteed to have at most $|T|+m(|A|+1)$ nonzero variables. This solution can have at most $m(|A|+1)$ fractional variables, since by constraint (2) every job must have at least one positive variable associated with it. (This kind of argument was first made and exploited by Potts [9] in context of parallel machine scheduling.) We now use a very simple rounding procedure to obtain an integral (an possibly infeasible) solution for the linear program. If job J_j has more than one nonzero variable associated with it we set one of them to be one and others to be zero in an arbitrary manner. In this solution the T-operations have a unique assignment to intervals of the second type. Let $D(k)$ be the total processing time of T-operations of jobs that received fractional assignments in interval k by the solution to the linear program. Notice, that $\sum_{k=0}^{|A|} D(k) \leq m(|A|+1)\mu\alpha\varepsilon P \leq 3m^2\mu\varepsilon P$.

2.4 Finding a Feasible Schedule

Consider some interval $[a_k, a_{k+1})$ of the second type. Let $p_{max}(k)$ be the length of the longest small or tiny operation assigned to this interval. By definition $d(k) \geq p_{max}(k)$. By construction, in the rounded solution the total length of operations assigned to this interval on each machine is at most $a_{k+1} - a_k + d(k) + D(k)$. We consider now the problem of scheduling the small and tiny operations in interval $[a_k, a_{k+1})$. This is simply a smaller instance of the job shop problem, and by using Sevastianov's algorithm [11] it is possible to find a feasible schedule for these operations of length at most $a_{k+1} - a_k + d(k) + D(k) + O(m\mu^3)p_{max}(k)$.

Note that we can schedule the small and tiny operations in each interval $[a_k, a_{k+1})$ independently from the operations in any other interval $[a_i, a_{i+1})$. Moreover, if we add $D(k) + O(m\mu^3)d(k)$ to the length of each interval, the union of the schedules for these intervals yields a feasible solution for the original constrained job shop problem (where we assume that all B-operations start at the beginning of some interval of the first type). The makespan of this schedule is at most

$$C + \sum_{k=0}^{|A|} \big(D(k) + O(m\mu^3)d(k)\big)$$

$$\leq C + 3m^2\mu\varepsilon P + O(m\mu^3)\Big(\sum_{O_{ij}\in S} p_{ij} + \sum_{k=0}^{|A|} \alpha\varepsilon P\Big)$$

$$\leq C + O(m^2\mu^3)\varepsilon P \leq C^*_{max} + O(m^2\mu^3)\varepsilon P.$$

Since m and μ are both constants and ε is an arbitrary rational number then our algorithm is a polynomial time approximation scheme for the job shop problem when m and μ are fixed.

3 Extensions

Multiprocessor job shop problem. In the multiprocessor (or m-stage) job shop problem each machine of the classical job shop problem is replaced by a set of m_s parallel identical machines. Our polynomial time approximation scheme works also in this case if the number of machines m_s on each stage s is fixed. Let the machines on stage s be numbered $s_1, s_2, \ldots, s_{m_i}$. Now the outline scheme must specify on which machine each operation $O_{kj} \in B \cup S$ must be scheduled. In the linear program we use variables $x_{j,(k_1,\ldots,k_\mu),(a_1,\ldots,a_\mu)}$, where a_i indicates the machine where the i-th operation O_{ij} of job $J_j \in T$ is scheduled. The same techniques used for the job shop scheduling problem can be used to design a polynomial time approximation scheme for this more general problem.

Dag shop problem. Another generalization of the job shop problem is the dag shop problem [15] (also called G - problem by Sevastianov [11, 12]). Here each job consists of a set of operations $O_j = \{O_{1j}, \ldots, O_{\mu j}\}$, and each job $J_j \in J$ has associated an acyclic directed graph $R_j = (O_j, E_j)$. In this graph an arc $(O_{i'j}, O_{ij})$ indicates that operation O_{ij} has to be executed after operation $O_{i'j}$. The problem is to find a schedule of minimum length that respects these ordering constraints.

In our model the acyclic graph R_j can be translated directly into a set of tuples $K_j = \{(k_1, \ldots, k_\mu) \mid 0 \leq k_j \leq |A|$ for all $1 \leq j \leq \mu$ and $k_{i'} \leq k_i$ for every edge $(O_{i'j}, O_{ij}) \in E_j\}$ for each job $J_j \in T$. Again, the size of each set of tuples is constant, $|K_j| \leq (|A|+1)^\mu$, so we can use our algorithm with some small changes. Let us consider a single interval $[a_k, a_{k+1})$. Let $O(k)$ be the set of operations assigned to this interval. For each job J_j corresponding to the operations in $O(k)$ we use, instead of the acyclic graph $R_j(k)$ induced by the operations $O_{ij} \in O(k)$, a linear order that extends $R_j(k)$. Sevastianov's algorithm [11] can be used in this case also. The rest of the algorithm is as before.

A further extension by Sevastianov [13] allows undirected edges between operations of a job J_j. It is not complicated to generalize the approximation scheme also to this problem.

Job shop problem with release times. Our techniques can also handle the case in which each job J_j has a *release time* r_j when it becomes available for processing. Let r_{max} be the maximum release time. Then, $r_{max} \leq C^*_{max}$, $P \leq C^*_{max}$, and $C^*_{max} \leq r_{max} + mP$. Let us divide all processing times and release times by $\max\{P, r_{max}\}$ to get the bounds $1 \leq C^*_{max} \leq m + 1$. Now, we can use a technique by Hall and Shmoys [3] to obtain a problem instance with only a constant number of release times. The idea is to round each release time down to the nearest multiple of $\varepsilon/2$ for some value $\varepsilon > 0$. Next, we apply a $(1 + \varepsilon/2)$ approximation scheme (described below) that can handle $1 + 2m/\varepsilon$ release times and then add $\varepsilon/2$ to the starting time of each job. Since the makespan length is increased by at most $\varepsilon C^*_{max}/2$, this gives a $(1 + \varepsilon)$ approximation scheme.

We can easily modify the definition of the outline scheme and the linear program to allow a constant number of release dates. Now the number of intervals is larger since each release time marks the beginning of some interval, but the total number is still constant. Using the bounds for C^*_{max}, we can solve the linear program as before in polynomial time. The rest of the approximation scheme is similar to that for the job shop scheduling problem.

Job shop problem with delivery times. In this problem each job J_j has an associated time q_j called its delivery time. If in a schedule job J_j completes its processing at time C_j, then its *delivery completion time* is equal to $C_j + q_j$. The goal is to minimize the maximum delivery completion time of any job. Again, we can solve this problem using an approach suggested by Hall and Shmoys [3].

4 Conclusions

We have proposed a general method to obtain a polynomial time approximation scheme for the job shop problem. Our method can be extended to design approximation schemes for multi-stage job shops and dag shops. Furthermore, we can handle also release times and delivery times.

Using further ideas, we can improve the running time of the approximation schemes to $O(n)$. We have also found an approximation scheme for the preemptive variant of the job shop problem. These results will appear in a forthcoming paper.

It is an open question to find a polynomial time approximation scheme for m-stage job shops (with a constant number m of stages and non-constant number m_i of parallel machines in each stage). This question remains open even for flow shops [2] and open shops [10].

Acknowledgments

We are grateful to Alexander Kononov, Lorant Porkolab and Sergey Sevastianov for many helpful discussions and comments.

References

[1] L. A. Goldberg, M. Paterson, A Srinivasan and E. Sweedyk, Better approximation guarantees for job-shop scheduling, In Proceedings of the 8th Symposium on Discrete Algorithms (1997), pp.599–608.

[2] L. A. Hall, Approximability of flow shop scheduling, Mathematical Programming **82** (1998), pp.175–190.

[3] L. A. Hall and D. B. Shmoys, Approximation algorithms for constrained scheduling problems, In Proceedings of the IEEE 30th Annual Symposium on Foundations of Computer Science (1989), pp.134–139.

[4] L. A. Hall and D. B. Shmoys, Near-optimal sequencing with precedence constraints, In Proceedings of IPCO (1990), pp.249–260.

[5] L. A. Hall and D. B. Shmoys, Jackson's rule for single -machine scheduling: Making a good heuristic better, Mathematics of Operation Research **17** (1992), pp.22–35.

[6] E. L. Lawler, J. K. Lenstra, A. H. G. Rinooy Kan and D. B. Shmoys, Sequencing and scheduling: Algorithms and complexity, In S. C. Graves, A. H. G. Rinooy Kan and P. H. Zipkin, editors, Handbook in Operation Research and Management Science, **v. 4**, Logistics of Production and Inventory, North-Holland (1993), pp.445–522.

[7] T. Leighton, B. Maggs and S. Rao, Packet routing and job-shop scheduling in O(Congestion+Dilation) Steps, Combinatorica **14** (1994), pp.167–186.

[8] T. Leighton, B. Maggs and A. Richa, Fast algorithms for finding O(congestion+dilation) packet routing schedules, to appear in Combinatorica.

[9] C. N. Potts, Analysis of a linear programming heuristic for scheduling unrelated parallel machines, Discrete Applied Mathematics **10** (1985), pp.155–164.

[10] P. Schuurman and G. J. Woeginger, Approximation algorithms for the multiprocessor open shop scheduling problem, Technical Report COSOR 97-23, Eindhoven University, The Netherlands.

[11] S. V. Sevast'janov, Bounding algorithm for the routing problem with arbitrary paths and alternative servers, Cybernetics **22** (1986), pp.773–780.

[12] S. V. Sevast'janov, On some geometric methods in scheduling theory: a survey, Discrete Applied Mathematics **55** (1994), pp.59–82.

[13] S.V. Sevast'janov, Nonstrict vector summation in multi-operation scheduling, Annals of Operations Research **83** (1998), pp.179–212.

[14] S. V. Sevastianov, G. J. Woeginger, Makespan minimization in open shops: A polynomial time approximation scheme, Mathematical Programming **82** (1998), pp.191–198.

[15] D. B. Shmoys, C. Stein and J. Wein, Improved approximation algorithms for shop scheduling problems, SIAM Journal of Computing **23** (1994), pp.617–632.

[16] D. P. Williamson, L. A. Hall, J. A. Hoogeveen, C. A. J. Hurkens, J. K. Lenstra, S. V. Sevast'janov and D. B. Shmoys, Short shop schedules, Operation Research **45** (1997), pp.288–294.