Worst-Case Performance of Approximation Algorithms for Tool Management Problems

Yves Crama, Joris van de Klundert

1 Ecole d’Administration des Affaires, Université de Liège, Bâtiment B31, 4000 Liège, Belgium

2 Department of Mathematics, Faculty of General Sciences, University of Maastricht, 6200 MD Maastricht, The Netherlands

Received November 1996; revised August 1998; accepted 11 November 1998

Abstract: Since the introduction of flexible manufacturing systems, researchers have investigated various planning and scheduling problems faced by the users of such systems. Several of these problems are not encountered in more classical production settings, and so-called tool management problems appear to be among the more fundamental ones of these problems. Most tool management problems are hard to solve, so that numerous approximate solution techniques have been proposed to tackle them. In this paper, we investigate the quality of such algorithms by means of worst-case analysis. We consider several polynomial-time approximation algorithms described in the literature, and we show that all these algorithms exhibit rather poor worst-case behavior. We also study the complexity of solving tool management problems approximately. In this respect, we investigate the interrelationships among tool management problems, as well as their relationships with other well-known combinatorial problems such as the maximum clique problem or the set covering problem, and we prove several negative results on the approximability of various tool management problems. © 1999 John Wiley & Sons, Inc. Naval Research Logistics 46: 445–462, 1999

1. INTRODUCTION

Regardless of the precise definition of flexibility in the term “flexible manufacturing systems,” the ability of machines to perform various operations on various products or parts is a most vital component of this flexibility. This versatility is achieved, in part, by equipping each machine with a tool magazine. This magazine can hold a set of tools which the machine can use to perform a succession of operations while incurring low setup costs when switching from one tool to another. The resulting flexibility may be advantageous from a strategic or tactical viewpoint, but it comes at a price. The complexity of (operational) planning and scheduling decisions is considerably higher than in conventional environments, even when the machines are considered

Correspondence to: J. van de Klundert
Contract grant sponsor: Office of Naval Research; contract grant numbers: N00014-92-J1375 and N00014-92-J8043
Contract grant sponsor: NATO; contract grant number: 931531

© 1999 John Wiley & Sons, Inc. CCC 0894-069X/99/050445-18
in isolation. Apart from the usual job sequencing decisions, tool handling decisions must now also be tackled. Hence, scheduling problems for a single flexible machine differ essentially from classical single machine scheduling problems. For this reason, such problems have received considerable attention in the literature over the last 15 years.

In this paper we are primarily interested in the mathematical properties of single flexible machine scheduling problems. Most of the problems considered in this paper are known to be NP-hard in the strong sense. This implies that, unless $P = NP$, there do not exist polynomial-time algorithms that solve these problems exactly. This paper investigates the complexity of solving such problems approximately. More specifically, we will be interested in analyzing the worst-case ratios that are effectively achieved by various polynomial-time approximation algorithms for single machine tool management problems, as well as in determining the worst-case ratios that can possibly be achieved by polynomial-time approximation algorithms for these problems (assuming that $P \neq NP$). In this paper, we refer to the latter broad issue as that of analyzing the approximability of the problems at hand (see [18]).

In our study, we borrow from both theoretical and applied previous work. Indeed, most of the approximation algorithms proposed in the literature have been analyzed on empirical grounds only. Some of them have been shown to perform reasonably well in this framework (see, e.g., [7, 9] and other references cited in [5]). On the other hand, a proper classification of the complexity and the approximability of the models requires a suitable theoretical framework. We choose here to concentrate on the worst-case performance guarantee achieved by polynomial-time algorithms. As usual in this type of theoretical investigation, its aim is not to be directly applicable to the solution of real world problems. This paper rather investigates general models that apply to a wide variety of problems, arising in areas such as database management, circuit partitioning, and flexible machine scheduling, and this latter area in turn ranges over many different industries with different technological characteristics (see [5]). The results of our analysis of such general models therefore indicate and exclude directions for improved solution methods for a wide variety of applications. In addition, our investigation may contribute to better models and solution methods for special cases as they are encountered in practice. Let it be noted however, that little information has appeared in the literature about characteristics of real life problems (the computational experiments mentioned above have been carried out on randomly generated instances). This lack of information regarding, for instance, the distributional features of problem instances, also causes average case analysis to be less attractive.

In the next section, we first discuss several of the basic single machine tool management models, their complexity, and their relationships with other combinatorial problems. We also briefly review a few available results regarding the approximability of these problems. In Section 3, we investigate the worst-case behavior of several algorithms proposed in the literature and we show that all these algorithms have very poor worst-case behavior. However, as yet, it is not known whether polynomial-time approximation algorithms with a better worst-case behavior can exist for the scheduling problems considered here. We establish some negative results on this topic in Section 4. Finally, in Section 5, we conclude by discussing directions for further research.

2. MODELS AND COMPLEXITY

In this section, we briefly present several models that arise naturally in the context of flexible machine scheduling and we discuss their complexity. Our main purpose is to facilitate the analysis in subsequent sections and to underline the kinship of these problems to various fundamental problems of combinatorial optimization.
2.1. Models

To start with, let us take a look at the physical characteristics of flexible machine scheduling. First of all, there is a machine on which a set of jobs have to be processed. Processing means that the machine performs one or several operations on these jobs, and the execution of each of these operations requires one or more tools (as specified by the process plan of the job). The machine can store a limited number of tools in its tool magazine. More precisely, we assume that the magazine contains $C$ slots, and that each tool requires exactly one slot (although more general models are possible; see, e.g., [5]). We also assume that no job can be preempted, so that all the tools required by a job must be loaded in the tool magazine before this particular job can be processed. We define a loading strategy for a list of jobs as a specification of the contents of the tool magazine at the beginning of the processing of each job in the list.

Switching between tools already loaded in the tool magazine entails very short setups, and thus, as long as a set of jobs only uses tools from the magazine, total setup time remains negligible. However, if the number of tools needed by a set of jobs exceeds the tool magazine capacity $C$, then it becomes unavoidable that some tools will have to be removed from the magazine and replaced by other tools while processing the set of jobs. A (tool) switch is the replacement of one tool by another one in the tool magazine. A switching instant is any time interval elapsed between the end of a job and the start of the next one, and during which one or several tool switches take place.

In many cases, tool switches cannot be performed while the machine is operating and may require nonnegligible time, so that the production makespan is directly determined by the setup time induced by the tool switches. When tool switches must be incurred, total setup time is usually computed in one of two ways (see Tang and Denardo [22, 23] for a discussion of the relevance of each model). If switches take place sequentially rather than simultaneously, then the setup time is assumed to depend linearly on the total number of tool switches. On the other hand, when tool switches can be performed simultaneously, then total set up time is assumed to depend linearly on the number of switching instants.

We now identify four basic tool management models arising in this scheduling framework (see Crama [5] for a more detailed discussion of these models):

1. **Batch selection**: Given a collection of jobs, find the largest subgroup (batch) of jobs that can be processed without tool switches.

2. **Job grouping**: Given a collection of jobs, find a processing sequence for the jobs and a loading strategy for the tool magazine so as to minimize the total number of switching instants.

3. **Tool switching**: Given a collection of jobs, find a processing sequence for the jobs and a loading strategy for the tool magazine so as to minimize the total number of switches.

4. **Loading problem**: Given a collection of jobs and a processing sequence for these jobs, find a loading strategy for the tool magazine so as to minimize the total number of switches.

We have enumerated here the optimization versions of the four problems. We refer to their decision versions by the same names.

Both the tool switching problem and the job grouping problem attempt to minimize the total setup time required by a collection of jobs, albeit with different cost functions. Note that the objective function of the job grouping problem is less sensitive to the exact job input sequence than the objective function of the tool switching problem: Indeed, the set of jobs that are processed between any two consecutive switching instants can be processed in any order without
entailing any new switches. Actually, if we call batch any set of jobs that can be processed without tool switches (or equivalently, that requires at most $C$ tools), then the job grouping problem boils down to finding a partitioning of the jobs into a minimum number of batches. This observation motivates our interest for the first problem, viz. the batch selection problem.

The loading problem has been investigated in various contexts (e.g., in the context of computer memory management [3]), and has been shown to be solvable in polynomial time by greedy or network flow techniques [22, 19]. For obvious reasons, this makes the study of polynomial-time approximation algorithms for the loading problem less interesting. In this paper we will be primarily interested in the batch selection problem and the job grouping problem. The approximability of tool switching will be briefly discussed in Section 4, in relation with the approximability of job grouping.

Let us now give a more precise description of the input to the batch selection, job grouping and tool switching problems. In all cases, an instance consists of a tool magazine capacity $C$, a set of jobs $p_1, \ldots, p_n$ and a set of tools $t_1, \ldots, t_m$. Each job $p_i$, $i = 1, \ldots, n$, is completely characterized by the set of tools that it requires, in the following way. We introduce a tool-job matrix $A$ whose $m$ rows correspond to tools and whose $n$ columns correspond to jobs. We let $a_{ij} = 1$ if job $p_j$ requires tool $t_i$, and $a_{ij} = 0$ otherwise, for $i = 1, \ldots, n$ and $j = 1, \ldots, m$. Thus, in summary, an instance of batch selection, job grouping or tool switching is completely specified by the pair $(A, C)$, where $A$ is a $(0, 1)$-matrix and $C$ is a positive integer. In the remainder, we assume that for each pair of jobs $\{p_i, p_j\}$, $i \neq j$, the set of tools required by $p_i$ is not a subset of the set of tools required by $p_j$.

With these notations, a batch can be viewed as a subset $J$ of the columns of $A$ such that $|\{i = 1, \ldots, m \mid \sum_{j \in J} a_{ij} > 0\}| \leq C$. The batch selection problem is to find a batch of maximum cardinality in $A$, and the job grouping problem is to partition the columns of $A$ into a minimum number of batches.

The tool-job matrix $A$ may also be viewed as the node-edge incidence matrix of a hypergraph $H = (V, E)$. Each row $i$ corresponds to a vertex $v_i \in V$ and each column $j$ to a hyperedge $e_j \in E$, with $a_{ij} = 1$ if edge $e_j$ contains vertex $v_i$. In this setting, the batch selection problem is to find a densest subset of vertices of cardinality $C$, i.e., a subset $S$ of vertices with cardinality $|S| = C$ such that $S$ contains the largest possible number of hyperedges [6, 15].

In this version, the batch selection problem can be seen to generalize the well-known maximum clique problem: Indeed, if $H = (V, E)$ is a graph (i.e., if each job requires two tools), then checking whether $H$ has a clique of size $C$ is equivalent to checking whether the densest subset of vertices of $H$ with size $C$ contains $\frac{1}{2} C(C - 1)$ edges. This implies, in particular, that batch selection is already NP-hard when each job requires two tools [12].

Of course, job grouping may also be interpreted in terms of hypergraphs: Here, the goal is to find a minimum cardinality collection of subsets $S_1, \ldots, S_K \subseteq V$ such that the subhypergraphs $H' = (S_i, E_S)$, $i = 1, \ldots, K$, induced by these subsets form a covering of $H$, i.e., $\bigcup_{i=1}^K E_{S_i} = E$. As observed in [7], this problem can in fact be viewed as a strict generalization of the set covering problem (see also Theorem 7 in Section 4).

2.2. Complexity

At this point, we can conclude (somewhat informally) that the batch selection problem appears to be at least as difficult as the maximum clique problem, while the job grouping problem is at least as difficult as the set covering problem. Both clique and set covering are notoriously hard from the point of view of approximability. We will discuss their exact status
in Section 4. For the time being, we are simply going to give a brief overview of known results about the complexity status of the three tool management problems under study.

We have already mentioned that batch selection is NP-hard [12, 14]. The job grouping problem is also known to be NP-hard [23]. In particular, the problem is strongly NP-hard even for fixed $C \geq 3$, and it is NP-complete to decide whether there exists a partitioning of the set of jobs into two batches [7]. Finally, the tool switching problem is strongly NP-hard even for fixed $C \geq 2$ [9].

Few attempts have been made to date to classify the problems discussed in this paper with respect to their approximability. Rajagopalan [20] establishes that a simple “First Fit Decreasing” heuristic “can do almost arbitrarily bad” for certain batching problems. Kortsarz and Peleg [16] consider the special case of the batch selection problem corresponding to the problem of finding a densest subset of vertices in a graph $G = (V, E)$. They construct a polynomial-time approximation algorithm with worst-case ratio $O(|V|^{7/18})$ for this problem. Goldschmidt et al. [15] also propose several polynomial-time approximation algorithms for special cases of both the batch selection and the job grouping problem. Their algorithms have worst-case ratios $O(1)$, $O(\ln C)$, or $O(C)$, depending on the restrictions placed on the problem. Goldschmidt, Nehme, and Yu [14] suggest a dynamic programming formulation for the batch selection problem and discuss conditions under which it can be implemented to run in polynomial time.

Besides the papers cited above, there is a vast amount of literature dealing with the same tool management problems, but focusing on the development and on the empirical testing of “practical” heuristics, rather than on a theoretical study of their worst-case performance. Several of these heuristics will be discussed in the next section. We refer to [5] for a more detailed overview of this topic.

3. PERFORMANCE ANALYSIS OF ALGORITHMS

We start this section with a description of a number of polynomial-time approximation algorithms for the batch selection problem, as they have been proposed in the literature. Each algorithm, except the last one, is characterized by a simple selection rule which specifies how to add a next job to a current batch. Unless otherwise stated, ties occurring from the application of the selection rule are arbitrarily broken. The selection rule is to be repeatedly applied until no more job can be added to the batch without exceeding the tool magazine capacity.

**Approximation algorithms for batch selection:**

1. **Maximal Intersection (MI).** Selection rule: Select the job which has the largest number of tools in common with the jobs already in the batch.
2. **Minimal Union (MU).** Selection rule: Select the job which, when added to the batch, requires the smallest number of additional tools.
3. **Maximal Intersection, Minimal Union (MIMU)** [23]. Selection rule: Select the job which has the largest number of tools in common with the jobs already in the batch. In case of a tie, select the job which requires the smallest number of additional tools.
4. **Whitney and Gaul** [24]. Selection rule: Let $t(Y)$ denote the number of tools required by the jobs in batch $Y$, and let $B$ be the current batch; select the job $p$ that maximizes the ratio $t(B \cup \{ p \}) + 1)/(t(\{ p \}) + 1)$.
5. **Rajagopalan** [20]. Selection rule: Define the weight of each tool to be the number of jobs that require it among the jobs not yet assigned to the batch; select the job for which the sum of the weights of the tools that are to be added when this job is selected is maximum.
6. **Modified Rajagopalan** [7]. Selection rule: Define the weight of a tool to be the number of jobs already selected that require this tool; select the job for which the sum of the weights of the tools needed by this job is maximum.

7. **Marginal Gain** [10]. Selection rule: Define the weight of a job to be the number of jobs that can be added without tool addition when this job is selected; select the job with maximum weight.

8. **Attila** [4]. Create an initial (infeasible) batch consisting of all jobs and requiring all tools. Then, iterate deleting tools from the current set of tools until the number of remaining tools equals the magazine capacity. In each iteration, delete the tool which causes the smallest number of jobs to be eliminated from the batch.

Each of the above algorithms or, more generally, every approximation algorithm $A_{BS}$ for the batch selection problem can be used to define a greedy-type approximation algorithm $A_{JG}$ for job grouping, in the following manner.

—apply $A_{BS}$ to create a first batch;
—eliminate from the instance all the jobs in this batch and apply $A_{BS}$ to the remaining jobs;
—repeat this procedure until there are no jobs left.

The sequence of batches that is iteratively produced by $A_{BS}$ forms a solution of the job grouping problem on the same data. As a matter of fact, all heuristics for the job grouping problem known to the authors are of this type. In the sequel, we use the same name (e.g., MIMU, Rajagopalan, etc.) when we refer to a batch selection algorithm or to the associated job grouping algorithm.

In the remainder of this section, we analyze the worst-case ratio of the eight approximation algorithms described above. For the batch selection problem, this worst-case ratio is defined as the supremum of the ratio

\[
\frac{\text{optimal value of the instance}}{\text{value provided by the approximation algorithm}}
\]

over all possible instances of the problem, whereas for job grouping it is the supremum of the ratio

\[
\frac{\text{value provided by the approximation algorithm}}{\text{optimal value of the instance}}
\]

over all instances of the problem (so that, in both cases, this ratio is at least equal to 1). An $r$-approximation algorithm is an algorithm with worst-case ratio at most $r$, where $r$ may be a constant or a function of the input. A polynomial-time approximation scheme is a family of algorithms $\{A^r \mid r > 1\}$ such that, for each $r > 1$, $A^r$ is a polynomial-time $r$-approximation algorithm (see, e.g., [13]).

In Theorems 1–4 below, we establish a number of lower bounds which are valid both for the worst-case ratios of batch selection algorithms and of the associated job grouping algorithms. In order to put these lower bounds in perspective, let us first derive two simple upper bounds on the worst-case ratio of any batch selection (or job grouping) algorithm. First, the number of jobs, viz. $n$, is such a trivial upper bound for any (reasonable) heuristic which puts at least one job in every batch. Next, observe that, for a magazine capacity $C$, the number of (distinct) jobs in
a batch cannot exceed \( \frac{C}{2} \). Hence, \( \frac{C}{2} \) also is an upper bound on the worst-case ratio of every heuristic (for instance, one that would construct batches randomly).

The next theorem shows that Whitney and Gaul and Rajagopalan algorithms can achieve these very poor upper bounds.

**THEOREM 1:** The worst-case ratio of Whitney and Gaul and Rajagopalan algorithms is at least \( \frac{C^2}{\sqrt{C}} \) and \( \Omega(n) \).

**PROOF:** Let \( k \) be some even integer. We create an instance of batch selection and job grouping involving two types of tools, respectively called “top tools” and “bottom tools.” There are \( k \) “top tools” and \( k \) “bottom tools.” Each job requires \( k/2 \) tools, which are either \( k/2 \) top tools or \( k/2 \) bottom tools. Hence we can also speak of “top jobs” and “bottom jobs.” More precisely, there is a top job for each possible choice of \( k/2 \) top tools and a bottom job for each possible choice of \( k/2 \) bottom tools. Thus we have \( 2 \times \binom{k}{k/2} \) jobs (see Fig. 1). The tool magazine capacity is \( C = k \).

Obviously, both the set of all top jobs and the set of all bottom jobs are optimal solutions to the batch selection problem. Moreover, the optimal solution for the job grouping problem is to form two groups (each of them an optimal batch).

It is left to the reader to check that Whitney and Gaul and Rajagopalan algorithms may construct batches of size 2 consisting of a top job and a bottom job, whereas the optimal batches consist of \( \binom{k}{k/2} \) jobs. This yields worst-case ratios of \( \Omega\left(\frac{C^2}{\sqrt{C}}\right) \) and \( \Omega(n) \) for batch selection and job grouping. \( \square \)

Notice that, in case all jobs require the same number of tools, the Whitney and Gaul rule boils down to a selection rule based on the “Maximal Union” principle: Select the job which, when added to the batch, requires the largest number of additional tools. Rajagopalan [20] already analyzed this “Maximal Union” rule (which he calls First Fit Decreasing) and proved that it can do “arbitrarily bad.” This was his motivation to introduce Rule 5 described above. We just showed, however, that this new rule can perform just as poorly (in the worst-case sense).

Algorithms MI, MU, MIMU and Modified Rajagopalan would solve optimally the instances constructed in the proof of Theorem 1. Their worst-case performance, however, is also very bad.

**THEOREM 2:** The worst-case ratio of algorithms MI, MU, MIMU, and Modified Rajagopalan is at least \( \Omega\left(\frac{C^2}{\sqrt{C}}\right) \) and \( \Omega(n/\log^2 n) \).
PROOF: Let $k$ be some even integer. Again, there are two sets of tools, called “top tools” and “bottom tools.” There are $k$ top tools, of which each job requires $k/2$, and there are $k/2 + 1$ bottom tools, of which each job requires only one. The jobs are defined as follows: There is a set of $k/2 + 1$ jobs for each possible choice of $k/2$ top tools, one job for each possible bottom tool. Thus, we have $\binom{k}{k/2} \times (k/2 + 1)$ jobs (see Fig. 2). The tool magazine capacity is defined to be $C = k + 1$.

Obviously, for the batch selection problem, an optimal batch is the set of jobs requiring the same bottom tool. Moreover, the optimal solution for the job grouping problem is to form $k/2 + 1$ groups (each of them an optimal batch), one for each bottom tool.

On the other hand, it is not hard to see that algorithms MI, MU, MIMU, and Modified Rajagopalan start with an arbitrary job and may subsequently select the job requiring the same top tools but another bottom tool. In this way, these algorithms construct batches of size $k/2 + 1$, whereas the optimal batches contain $\binom{k}{k/2}$ jobs. The ratio between the number of jobs in an optimal batch and the number of jobs in a batch found by either of the heuristics is therefore $\Omega\left(\frac{2^k}{C^2}\right)$ The same ratio applies for job grouping. \qed

The heuristic proposed by Dietrich, Lee, and Lee [10] solves the previous instances optimally. We have, however, the following theorem:

THEOREM 3: The worst-case ratio of the Marginal Gain algorithm is at least $\Omega\left(\left(\frac{\sqrt{C}}{2}\right)^{\sqrt{C}}\right)$ and $\Omega(n)$.

PROOF: Let $k$ be an even integer. First, we construct a matrix $D$ which will eventually become a submatrix of the tool-job matrix. The columns of $D$ correspond to all possible $k/2$-element subsets of $\{1, \ldots, k\}$. The rows of $D$ correspond to all 2-element subsets of $\{1, \ldots, k\}$. We let $d_{ij} = 1$ if the $i$th 2-element subset is contained in the $j$th $(k/2)$-element subset, and $d_{ij} = 0$ otherwise (see Fig. 3).

The columns of $D$ satisfy the following property:

(P) For every three distinct columns $r, s, t$ of $D$, there exists a row $i$ of $D$ such that $d_{ir} = 1$ and $d_{is} = d_{it} = 0$.

Based on the matrix $D$, we now construct an instance of the batch selection and job grouping problems. Again we introduce top tools and bottom tools. The top tools correspond to the rows

Figure 2. Tool-job matrix used in the proof of Theorem 2, for $k = 4$. 

<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

(continued on next page)
of matrix $D$. Moreover, there are two disjoint sets of bottom tools, called bottom sets, each with cardinality

$$\binom{k}{2} - \binom{k/2}{2}.$$  

There are two jobs for each column of $D$: Each job requires all the top tools defined by the corresponding column of $D$, as well as all the bottom tools in one of the bottom sets (and none of the other bottom tools). Thus, there are $2 \times \binom{k}{k/2}$ jobs. Finally, we set $C = 2^{\frac{k}{2}} - \binom{k/2}{2}$.

For this instance, there are two optimal batches of size $\binom{k}{k/2}$, associated with the two bottom sets. On the other hand, the Marginal Gain algorithm may pick batches in which all bottom tools are required and only $\binom{k/2}{2}$ top tools, as follows. The algorithm selects an arbitrary job to begin with. All jobs whose top tool requirements and bottom tool requirements differ from the requirements of the already selected job cannot be added to the batch, since this would require more than $C$ tools. The single job that has the same top tool requirements, but requires the other set of bottom tools, has weight zero. Moreover, because of property $(P)$, all jobs that have the same bottom tool requirements, but different top tool requirements, also have weight zero. Thus, the rule may select the single job with the same top tool requirements, filling up the tool magazine completely. In this way the rule selects a batch of size 2. Since the optimal batch has size $\binom{k}{k/2}$, and $C = \Theta(k^2)$, this yields the desired ratio.  

Finally, for the Attila algorithm (originally proposed by Chaillou, Hansen, and Mahieu [4] for the special case where each job requires exactly two tools), we obtain the next theorem.

**THEOREM 4:** The worst-case ratio of algorithm Attila is at least $\Omega\left(\frac{\sqrt{C}}{\sqrt{2}}\right)$ and $\Omega(n/\log n)$.

**PROOF:** Again, we let $k$ be some even integer and we introduce top and bottom tools. There are $k$ top tools, of which each job requires $k/2$, and there are $(k/2 + 1) \times k/2$ bottom tools, of which each job requires $k/2 \times k/2$. The bottom tools are divided into $k/2 + 1$ disjoint sets, called bottom sets, each consisting of $k/2$ tools.
We have a set of \( \frac{k}{2} \) jobs for each possible choice of \( \frac{k}{2} \) top tools; each job in this set requires all tools in all but one of the bottom sets, and none of the tools in the remaining bottom set. Thus, there are \( \binom{k}{\frac{k}{2}} \times (k/2 + 1) \) jobs (see Fig. 4). The tool magazine capacity is \( C = (k/2)^2 + k \).

Obviously, for the batch selection problem, an optimal batch is a set of \( \binom{k}{\frac{k}{2}} \) jobs with identical bottom tool requirements. Moreover, an optimal solution for the job grouping problem is to form \( k/2 + 1 \) groups (each of them an optimal batch), one for each possible requirement of bottom tools.

Let us now study the behavior of algorithm Attila. Starting with all the tools, the algorithm must delete \( k/2 \) tools. We claim that it deletes (or may delete) \( k/2 \) top tools. Suppose that, after iteration \( i \ (i \in \{0, \ldots, k/2 - 1\}) \), the algorithm has not yet deleted any bottom tools. Then, the number of remaining jobs equals \( \binom{k-i}{\frac{k}{2}} \times (k/2 + 1) \). By symmetry, every bottom tool is required by a fraction of \( \frac{k}{k/2 + 1} \) of all jobs. Similarly, every top tool is required by a fraction of \( \frac{k}{k/2} \) of all jobs. Hence the algorithm may select a top tool again. After \( k/2 \) such iterations, we thus end up with \( k/2 \) top tools, all bottom tools and a batch of \( k/2 + 1 \) jobs. Now, since \( C = \Theta(k^2) \), this yields a ratio of

\[
\Omega\left(\left(\frac{\sqrt{C}}{\sqrt{C/2}}\right) / \sqrt{C}\right)
\]

for the batch selection problem.

We now show that the heuristic performs equally bad on the job grouping problem. Actually, when solving the job grouping problem by repeatedly applying algorithm Attila, we may create a batch for each possible top tool requirement. In view of the discussion above, it suffices to

\[
\begin{pmatrix}
1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\
1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 1 & 1 & 0 \\
0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\
0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 1 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1
\end{pmatrix}
\]

Figure 4. Tool-job matrix used in the proof of Theorem 4, for \( k = 4 \).
notice that, for every set of jobs $J$ such that $J$ contains at least two jobs with distinct top tool requirements, there is always some top tool that is required by at most $\frac{k^2}{k^2 + 1}$ jobs.

Although the bounds we have obtained in this section imply a rather poor worst-case performance for the various algorithms, we have not shown that these bounds are tight. Thus, some of the bounds (especially those derived in Theorems 3 and 4) may be subject to improvement.

4. LIMITS ON APPROXIMABILITY

The results in the previous section being disappointing, the question arises whether polynomial-time approximation algorithms with better worst-case ratios can exist for batch selection and job grouping. The relationships (hinted at in Section 2) between batch selection and maximum clique problems on the one hand, and between job grouping and set covering problems on the other hand, suggest that good approximation algorithms may not exist for the problems under study: Indeed, both maximum clique and set covering are notoriously hard to approximate. Let us now pursue this line of reasoning even further.

Arora et al. [1] have recently proved that, for some constant $\epsilon > 0$, a polynomial-time approximation algorithm with worst-case ratio $O(|V|^{\epsilon})$ cannot exist for the maximum clique problem unless $P = NP$. Long before this, however, it was known that the existence of a polynomial-time approximation algorithm with constant worst-case ratio for the maximum clique problem would imply the existence of a polynomial-time approximation scheme for this problem (an event that was regarded as highly unlikely; see, e.g., [13] for a discussion). We now present a similar result for batch selection.

THEOREM 5: If there is a polynomial-time approximation algorithm with constant worst-case ratio for the batch selection problem, then there is a polynomial-time approximation scheme for this problem.

PROOF: Suppose that $H$ is a polynomial-time approximation algorithm for batch selection with constant worst-case ratio $r \geq 1$. Fix $\epsilon > 1$ and let $l_\epsilon$ be the smallest integer such that $r^{1/l_\epsilon} < \epsilon$. We intend to construct a polynomial-time $\epsilon$-approximation algorithm for batch selection.

To this end, consider any instance $I$ of batch selection, consisting of a magazine capacity $C$ and an $m \times n$ tool-job matrix $A$. We are now going to describe a squaring procedure which produces in polynomial-time a new instance $I'$ of batch selection, with capacity $C'$ and tool-job matrix $A'$, such that

(i) from any solution of $I'$ with value $s'$, we can construct in polynomial time a solution of $I$ with value $s$ such that $s^2 \geq s'$, and
(ii) $OPT(I') = OPT(I)^2$, where $OPT(I)$ and $OPT(I')$ are the optimal values of $I$ and $I'$, respectively.

Notice that this suffices to prove the theorem. Indeed, given an instance $I$ of batch selection, we can iterate $\lceil \log l_\epsilon \rceil$ times the squaring procedure to produce a new instance $I^*$. We then apply $H$ to $I^*$ and, from its solution with value $s^*$, we construct a solution of $I$ with value $s$. From the definitions of $H$, $l_\epsilon$, and the squaring procedure, there follows that $OPT(I^*) \leq rs^*$, and hence $OPT(I) \leq \epsilon s$. Moreover, the running time of the whole procedure is polynomial in the size of
(since \( l_e \) is a constant), and thus we have obtained a polynomial-time \( \epsilon \)-approximation for the batch selection problem, as required.

Let us now describe the squaring procedure. We construct \( C' \) and \( A' \) from \( C \) and \( A \) as follows. The magazine capacity \( C' \) is set equal to \((C + 3)C\). The tool-job matrix \( A' \) has \((C + 3)m \) rows and \( n^2 \) columns. For each row of \( A \), there are \( C + 3 \) rows in \( A' \), and for each column of \( A \) there are \( n \) columns in \( A' \). More precisely, for all \( j, l = 1, \ldots, n \), column \((j - 1)n + l \) of \( A' \) depends as follows on columns \( j \) and \( l \) of \( A \):

1. For all \( i = 1, \ldots, m \), \( a'_{i,(j-1)n+l} = a_{i,j} \).
2. For all \( i = 1, \ldots, m \), \( a'_{(C+2)m+i,(j-1)n+l} = a_{i,l} \).
3. For all \( i = 1, \ldots, m \) and \( k = 1, \ldots, C + 1 \), \( a'_{m+(C+1)(i-1)+k,(j-1)n+l} = 1 \) if either \( a_{i,j} = 1 \) or \( a_{i,l} = 1 \), and \( a'_{m+(C+1)(i-1)+k,(j-1)n+l} = 0 \) otherwise.

An example of this transformation is displayed in Figures 5 and 6, where we assumed that \( C = 2 \). In general, the transformed instance \( I' \) may be viewed as follows. The tools of \( I' \) are partitioned into \( m + 2 \) subsets:

\[
\begin{pmatrix}
1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\
1 & 1 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\
1 & 1 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 1 & 0 & 1 & 1 & 1 & 0 & 1 & 0 & 0 \\
0 & 1 & 0 & 1 & 1 & 1 & 0 & 1 & 0 & 0 \\
0 & 1 & 0 & 1 & 1 & 1 & 0 & 1 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \\
0 & 0 & 1 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \\
0 & 0 & 1 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \\
1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\
0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0
\end{pmatrix}
\]

**Figure 5.** Tool-job matrix \( A \).

**Figure 6.** Transformed tool-job matrix \( A' \).
For any given tool in block $i$, we call $t_i$ the corresponding top tool, and we call $t'_{m+(C+1)i}$ the corresponding bottom tool.

Now, for each ordered pair of jobs $(p_j, p_l)$ in $I$, we obtain a job $p'_{(j-1)m+l}$ in $I'$. For this job, the top tool requirements duplicate the tool requirements of job $p_j$ and the bottom tool requirements duplicate the tool requirements of job $p_l$. Moreover, if tool $t_i$ is required by either $p_j$ or $p_l$, for some $1 \leq i \leq m$, then all the tools of block $i$ are required by $p'_{(j-1)m+l}$.

It follows from this description that if a job of $I'$ requires any tool from block $i$, $1 \leq i \leq m$, then it requires all the tools from this block. Thus, in any solution of the batch selection problem, it is pointless to select any tool from a block unless we select all of them. In the following, we assume without loss of generality that any solution contains either all or no tools from each block.

Similarly, when a job of $I'$ requires one of the top tools $t'_i$ or one of the bottom tools $t'_{(C+2)i}$, $1 \leq i \leq m$, then the same job also requires all the tools from block $i$. Therefore, if a solution of the batch selection problem selects $w$ blocks, then we may assume that it selects at most $w$ top tools and at most $w$ bottom tools. Since $C' = (C + 3)C$, and each block consists of $C + 1$ tools, we may as well assume that it selects at least $C$ blocks. On the other hand, since $C' = (C + 3)C < (C + 2)(C + 1)$, it cannot use $C + 2$ blocks.

So, let us first consider the case where the solution selects exactly $C + 1$ blocks. This means it can select $C(C + 3) - (C + 1)(C + 1) = C - 1$ top and bottom tools altogether. But this in turn implies that it has selected at least two blocks for which there are no corresponding top and bottom tools. Hence, we can unselect one of these blocks without reducing the number of jobs in the batch, which brings us back to a situation where the solution contains only $C$ blocks.

This leaves us with the case in which the solution requires $C$ blocks. In this case, we can select $2C$ top and bottom tools altogether. We know however, that there is no benefit in selecting top and bottom tools whose corresponding block is not selected. Hence, given a selection of $C$ blocks, we may assume that the remaining tools in the solution are the corresponding top and bottom tools.

We are now ready to conclude the proof by showing that the transformation $I \to I'$ satisfies conditions (i) and (ii). Consider first a solution of $I'$ with value $s'$, i.e., a batch $S'$ of size $|S'| = s'$ for $I'$. Without loss of generality, this batch requires $C$ blocks and their corresponding top and bottom tools. Let $i_1, \ldots, i_C$ be these $C$ blocks, and let $S = \{p_1, \ldots, p_s\}$ be the batch of $I$ that can be processed using tools $t_{i_1}, \ldots, t_{i_C}$. Then, it is easy to check that every job in $S'$ corresponds to some ordered pair $(p_j, p_l)$ such that $p_j, p_l \in S$. Hence, $s' \leq s^2$ and condition (i) is satisfied.

Conversely, consider a batch of jobs $S = \{p_1, \ldots, p_s\}$ of the original instance $I$, and let $t_{i_1}, \ldots, t_{i_C}$ denote the tools required by $S$. We can construct a solution to $I'$ by selecting the top tools $t'_{i_1}, \ldots, t'_{i_C}$ as well as the corresponding bottom tools and blocks. This choice of tools allows to process exactly those jobs of $I'$ corresponding to all ordered pairs $(p_j, p_l)$ such that $p_j, p_l \in S$, that is a batch of size $s^2$. Together with the previous observations, this implies that $OPT(I') = OPT(I)^2$. Thus, condition (ii) is satisfied and the proof is complete. \qed

In view of Theorem 5, polynomial-time approximation algorithms with constant worst-case ratio can be ruled out for the batch selection problem if we can show that, for some constant $\varepsilon$...
> 1, no polynomial-time \( \varepsilon \)-approximation algorithm exists for the problem. We have not been able to establish such a result. We note, however, that Bellare [2] has obtained a result of this nature for a generalization of the batch selection problem which he calls the maximum system of representatives (MSR) problem. In our terminology, MSR can be described as follows: Given a set of jobs, a partition of the set of tools into \( K \) classes \( \{ T_1, \ldots, T_K \} \), and \( K \) integers \( C_1, \ldots, C_K \), find the largest batch of jobs that can be processed by using at most \( C_i \) tools from \( T_i \), for \( i = 1, \ldots, K \). (Observe that the batch selection problem arises when all tools are in the same class.) It is not hard to extend Theorem 5 to MSR. On the other hand, Bellare [2] has shown that there is a constant \( \varepsilon > 1 \) such that the existence of a polynomial-time \( \varepsilon \)-approximation algorithm for MSR would imply \( P = NP \). (Actually, Bellare proved this result for the special case where \( C_1 = \cdots = C_K = 1 \) and all jobs require exactly two tools. The proof is based on the connection between interactive proofs and approximation algorithms established by Feige et al. [11].) Combining these observations, we conclude that polynomial-time approximation algorithms with constant worst-case ratio cannot exist for MSR unless \( P = NP \). We omit a more formal proof of this statement, since it seems to be of limited interest.

Let us now turn to the job grouping problem and its close relative, the set covering problem. We first mention an interesting relationship between the approximability of batch selection and job grouping, which follows easily from more general results on set covering problems:

**THEOREM 6:** For every \( r \geq 1 \), if there exists a polynomial-time \( r \)-approximation algorithm for the batch selection problem, then there exists a polynomial-time approximation algorithm with worst-case ratio \( O(rC) \) and \( O(r \log(n/r)) \) for the job grouping problem.

**PROOF:** As implied by Theorem 3 in [8], the greedy-type algorithm \( A_{JG} \) for job grouping presented in Section 3 has worst-case ratio \( O(r \log(N/r)) \), where \( r \) is the worst-case ratio of the approximation algorithm \( A_{BS} \) used to generate the sequence of batches and \( N \) is an upper bound on the number of jobs in a feasible batch. Trivially, \( N \leq n \) and, as discussed in Section 3, \( N \leq O\left(\frac{2^C}{C^C}\right) \). The claims follow from these observations. \( \square \)

Thus, the job grouping problem is, in a sense, not “much more” difficult to approximate than the batch selection problem.

On the other hand, the same breakthroughs that led to the negative result regarding the approximability of clique enabled Lund and Yannakakis [17] to prove that, for any \( d, 0 < d < \frac{1}{2}, \) a polynomial-time approximation algorithm for set covering with worst-case ratio \( d \log n \) cannot exist, unless \( NP \subseteq DTIME[n^{poly \log n}] \) (here \( n \) is the cardinality of the set to be covered). We now show that this result can be extended to the job grouping problem.

**THEOREM 7:** For any \( 0 < d < \frac{1}{2}, \) the job grouping problem cannot be approximated within a factor of \( d \log n \) in polynomial time unless \( NP \subseteq DTIME[n^{poly \log n}] \), even if \( C = m - 1 \).

**PROOF:** The statement follows from Lund and Yannakakis’ result [17] and from a transformation given by Crama and Oerlemans [7], which implies that set covering can be interpreted as a special case of job grouping with \( C = m - 1 \). For the sake of completeness, we reproduce here this simple transformation.

Consider an instance of the set covering problem: \( \mathcal{F} = \{ S_1, \ldots, S_m \} \) are subsets of the ground-set \( \mathcal{E} = \{ e_1, \ldots, e_n \} \), and the problem is to find a minimum cardinality subcollection of \( \mathcal{F} \) whose union is \( \mathcal{E} \). Let us now create an instance of the job grouping problem involving \( m \) tools and \( n \) jobs, where we assume that job \( p_j \) requires tool \( t_i \) if and only if \( S_i \) does not contain
$e_j (i = 1, \ldots, m, j = 1, \ldots, n)$ and where we set $C = m - 1$. Now, observe that a set of jobs $J$ constitutes a feasible batch if and only if there is some tool, say $t_i$, that is not required by any job in $J$. Equivalently, $J$ is a feasible batch if and only if there is a subset $S_i \in \mathcal{F}$ that contains $\{ e_j \mid p_j \in J \}$. Thus, any covering of the job set by a collection of feasible batches corresponds to a covering of $\mathcal{F}$ by a subcollection of $\mathcal{F}$, and this implies that the set covering problem is equivalent to a special case of the job grouping problem. (Observe that the batch selection problem can be solved trivially for this instance, so that finding a minimum cover poses the only difficulty.)

Finally, we close this section by discussing the relationship between approximation algorithms for the job grouping problem and for the tool switching problem. In the same spirit as Theorem 6, we obtain:

**Theorem 8:** For every $r \geq 1$,

(i) if there exists a polynomial-time $r$-approximation algorithm for the job grouping problem, then there exists a polynomial-time $(rC)$-approximation algorithm for the tool switching problem;

(ii) if there exists a polynomial-time $r$-approximation algorithm for the tool switching problem, then there exists a polynomial-time $(rC)$-approximation algorithm for the job grouping problem.

**Proof:** Consider a fixed instance $I = (A, C)$. Denote by $OPT_{JG}$ (resp. $OPT_{TS}$) the optimal value of the job grouping (resp. tool switching) problem on this instance, and denote by $v_{JG}(S)$ (resp. $v_{TS}(S)$) the value of solution $S$ to the job grouping (resp. tool switching) problem.

Consider any solution $S$ to the tool switching problem. This solution consists of a job processing sequence and a corresponding sequence of tool switches (i.e., a loading strategy). Trivially, the number of switching instants implied by the loading strategy does not exceed the number of switches. Therefore, $S$ can be used to produce a solution $S'$ of the job grouping problem with value $v_{JG}(S') \leq v_{TS}(S)$ and this implies

$$OPT_{JG} \leq OPT_{TS}.$$ 

Conversely, let $S'$ be a solution to the job grouping problem, i.e. a collection of batches. Between any two successive batches, at most $C$ tool switches must be incurred. Hence, there is a solution $S$ to the tool switching problem with value $v_{TS}(S) \leq C \times v_{JG}(S')$, which in turn implies that

$$OPT_{TS} \leq C \times OPT_{JG}.$$ 

Now, suppose that we have an approximation algorithm $H'$ with worst-case ratio $r$ for the job grouping problem. When applied to instance $I$, this algorithm outputs a solution $S'$ from which we can produce (as above) a solution $S$ of the tool switching problem such that

$$r \geq \frac{v_{JG}(S')}{OPT_{JG}} \geq \frac{v_{JG}(S')}{OPT_{TS}} \geq \frac{1}{C} \times \frac{v_{TS}(S)}{OPT_{TS}}.$$
Thus, we have derived an approximation algorithm with worst-case ratio \( Cr \) for the tool switching problem.

On the other hand, suppose that \( H \) is an \( r \)-approximation algorithm for the tool switching problem which, when applied to \( I \), outputs a solution \( S \). From this solution, we can construct a solution \( S' \) to the job grouping problem such that

\[
1 - \frac{v_{TS}(S)}{OPT_{TS}} \leq \frac{v_{JG}(S')}{OPT_{JG}} \leq \frac{1}{C} \times \frac{v_{JG}(S')}{OPT_{JG}}.
\]

Thus, we have obtained an approximation algorithm with worst-case ratio \( Cr \) for the job grouping problem. \( \square \)

5. FURTHER RESEARCH

The negative results presented in Section 4 strongly suggest that a worst-case ratio bounded by a constant (in the case of batch selection, Theorem 5) or by \( O(\log n) \) (in the case of job grouping, Theorem 7) cannot be achieved in polynomial time. There still remains a rather big gap, however, between the worst-case ratio attained by the approximation algorithms proposed to date (Theorems 1–4) and the ratios that are unlikely to be achievable in polynomial time. We conjecture that in order to close this gap both stronger negative results as well as better approximation algorithms are required. As yet, we have not investigated whether the bounds we derived on the worst-case ratios of the approximation algorithms discussed in Section 3 are tight, nor have we investigated methods that would enable tight worst-case analysis. Such methods are also required in order to close the aforementioned gap, and appear to be an interesting direction for further research.

Furthermore, we have seen that all approximation algorithms analyzed in Section 3 display worst-case ratios that are superpolynomial functions of the magazine capacity \( C \). Goldschmidt et al. [15] also carried out the analysis of their algorithms in terms of \( C \). The negative results of Section 4, however, bring no information as to the worst-case ratios that are achievable as functions of \( C \). It may well be the case that, for some \( \epsilon > 0 \), there exists no polynomial-time approximation algorithms with worst-case ratio smaller than \( 2^{c^\epsilon} \) for batch selection and job grouping (unless \( P = NP \)). Proving such strong negative results provides an additional interesting challenge.

On the positive side, in view of Theorem 6, a strengthening of the negative result regarding job grouping would have implications for batch selection as well. For example, if one could show that there is some \( \epsilon \) such that there does not exist a polynomial-time \( n^{\epsilon} \)-approximation algorithm for job grouping, then this would automatically rule out the existence of a polynomial-time \( n^{\epsilon} \)-approximation algorithm for batch selection. Notice that results of this nature exist for problems that are related in a similar fashion, e.g., graph coloring and maximum independent set problems [17].

The importance of good solution methods for tool management problems is widely recognized. In this paper, we did not contribute to this field in terms of proposing improved solution methods for real life problems. Instead, we studied mathematical properties of tool management problems and their currently known solution methods. In our view, this step is necessary for a proper understanding of these problems, and therefore to indicate and to exclude directions for developing methods that would eventually yield better solutions to real life problems. Our investigations reveal that the hardness of tool management problems, measured in terms of
performance guarantees, is not completely understood yet. Tool management problems belong to the most basic scheduling problems in contemporary manufacturing and, at the same time, feature interesting relationships with fundamental combinatorial problems. Therefore, we hope that the research reported in this paper and the open questions we have identified will stimulate further research regarding their algorithmic complexity.

ACKNOWLEDGMENTS

The authors gratefully acknowledge improvements suggested by the associate editor and by two anonymous referees. The first author was partially supported in the course of this research by the Office of Naval Research (Grants N00014-92-J1375 and N00014-92-J4083) and by NATO (Grant CRG 931531).

REFERENCES


