On some geometric methods in scheduling theory: a survey

S.V. Sevast'janov

Institute of Mathematics, Novosibirsk 630090, Russian Federation

Received 23 January 1990; revised 4 March 1993

Abstract

Suppose we are given a closed walk consisting of n steps in m-dimensional space, each step having at most unit length (in a certain norm s). We wish to include the walk into a given region of the space by reordering its steps. It turns out that our ability to solve this problem in polynomial time for certain regions of the space (such as the right triangle or hexagon in the plane, the ball in $\mathbb{R}^m$, etc.), enables us to construct approximation algorithms with good performance guarantees and polynomial running time for scheduling flow shops, job shops and other problems of the type, known to be NP-hard. Some other geometric methods are also to be spoken of subject to their application to scheduling problems.

1. Introduction

The survey does not cover all scheduling problems; we restrict ourselves to such classes of problems as flow shop, job shop, open shop and their modifications and the volume calendar planning problem as well. We do not make it our aim to describe all known geometric methods in scheduling theory. We are going to speak only about:

- a set of algorithms that can be entitled as “a compact vector summation in m-dimensional space”;
- a geometrical problem of finding the parallelepiped vertex most close to a given interior point – approximation algorithms for the problem are to be discussed;
- applications of these algorithms to the scheduling problems mentioned above.

So, we will describe a range of problems and methods due to the author, although the results of other (mainly soviet) mathematicians in the area will be reviewed as well. There were some reasons for writing the paper. The main one was that the West’s specialists in scheduling theory were practically unfamiliar with papers in this direction. Thus, only the paper [7] was mentioned in the survey [13] and only two [7, 20] – in one of the latest surveys [25]. (More results in this area are presented in Table 1.) Such a blank of information can be explained by the fact that all papers in this direction have been written in Russian and Hungarian (exceptions are the two...
mentioned above, written in English). At the same time, the urgency of the direction is noted in every survey paper. For example, Blazewitch writes in [13]: “Thus, one is interested in evaluating some heuristic algorithms, but not much work has been done in this area.” Presenting the paper, we intend to convince the reader that such a pessimistic estimate is far from the truth, and moreover, a promising future is in store for this direction.

The other reason was that it was very hard to get the right information about these results from the Mathematical Review journal. (For example, if someone intends to learn something about the paper “Geometry in scheduling theory” [41], I would not advice him to appeal to MR [89m:90088].)

So, I hope the reader will find something new in this paper. (Those who are interested in more complete information about the present state of multi-operation scheduling are referred to [49].) A great number of the results we are to consider will regard polynomial approximation algorithms with absolute performance guarantees. (Exceptions are some results regarding polynomially solvable cases of the open shop problem.) Those guarantees will be described in terms of the maximum operation length and a function of the number of machines (m). Since they are proved to be independent of the number of jobs (n) and since the optimality criterion (such as the makespan) tends to infinity as n does so, the algorithms under consideration become asymptotically optimal for fixed m and increasing n. This fact together with polynomial running time make the algorithms attractive for practical scheduling.

The rest of the paper is organized as follows. In Sections 2 and 3 we state scheduling problems, and for each we say a few words about solving algorithms and their characteristics, such as accuracy and complexity. Then (in Sections 4 and 5) we indicate some possible directions for obtaining new results and state some open questions on the subject. The results in this area are listed in Table 1 supplied with references. In the text we sometimes will not write a precise formula of the result, indicating its list number in the table instead. Some comments on Table 1 as well as the notations used in it are also presented in Section 6.

2. A bit of history

2.1.

It was 1973 that my supervisor, V.A. Perepeliča stated two rather different scheduling problems for my choice. The first one was the well-known Johnson problem [23] (we will call it the flow shop problem, or for short, FS-problem). The other was the volume calendar planning problem (VCP) [46]. Before they are formulated, let us introduce some terms and notations to be used in further settings of scheduling problems. To begin with, let us formulate the following general problem (for short, G-problem).
G-problem. There are \( n \) jobs and \( m \) dedicated machines. Each job, \( j = 1, \ldots, n \), consists of \( r_j \) operations \( \{o_j^1, \ldots, o_j^r\} = O_j \). Each operation \( o_j^i \) may be processed in time \( t_j^i \) by any machine from a given set \( J_i^j \subseteq \{1, 2, \ldots, m\} \). For each job \( j \), a precedence relation on the set \( O_j \) is specified. It is derived from an acyclic directed graph \( R_j = \{(o_j^i, o_j^{i+1})\} \) with vertex set \( O_j \) where the inclusion \( (o_j^i, o_j^{i+1}) \in R_j \) means that \( o_j^{i+1} \) is completed before \( o_j^i \) can start. Finally, two conditions will be permanent in all scheduling problems: that no preemption of an operation is allowed and that no machine can process more than one job at a time. As a rule, the following property (*) will be required as well: no two operations of the same job may be processed simultaneously. (If this is not the case for a problem, we will mention this explicitly.) The objective is to find a schedule \( S = \{s^i\} \) which satisfies the above requirements and minimizes the makespan:

\[
F(S) = \max_{j, i} (s_j^i + t_j^i) \to \min.
\]

Using the notations of the G-problem, we can formulate the FS-problem as follows.

FS-problem. \( r_j \equiv m; \quad R_j = \{(o_j^k, o_j^{k+1}), \quad k = 1, \ldots, m - 1\}; \quad J_i^j = \{i\} \quad (j = 1, \ldots, n; \quad i = 1, \ldots, m).\)

Thus, all jobs follow the route \((1, 2, \ldots, m)\) on the machines.

VCP-problem [31]. The annual plan of an enterprise consists of \( n \) items. Each item is characterized by an \( m \)-dimensional vector \( t_j = (t_{j1}, \ldots, t_{jm}), \quad j = 1, \ldots, n \). The objective is to divide the annual plan into \( l \) parts as equal as possible:

\[
I(S) = \max_{p, l} \max_{k = 1, \ldots, m} \left| \sum_{j \in N_p} t_{jk} - \sum_{j \in N_{i}} t_{jk} \right| \to \min_S
\]

where \( S = \{N_1, \ldots, N_l\}, \ \bigcup_i N_i = \{1, \ldots, n\}; \quad N_i \cap N_j = \emptyset.\)

Another criterion to the problem was considered in [34]:

\[
L(S) = \max_{p = 1, \ldots, l} \left\| \sum_{j \in N_p} t_j \right\|_1 \max_{j = 1}^{n} \left\| \sum_{j = 1}^{n} t_{j}/l \right\|_s \to \min_S
\]

where \( s \) is a norm in \( \mathbb{R}^m \).

In addition to two above-mentioned settings of the VCP-problem we could formulate a scheduling problem similar to VCP, interpreting \( t_j^i \) as the length of the operation \( o_j^i \) (being processed on the \( i \)-th machine), and \( l \) as the number of alternative brigades each having the complete set of \( m \) machines. The objective in this problem is to minimize the makespan: \( F(S) = \max_{k, p} \sum_{j \in N_p} t_{jk} \to \min_s. \) (The problem with this criterion was stated in [32].) The property (*) is not required: \( R_j = \emptyset, \forall j.\)
When solving the VCP-problem, we have the only difficulty: how to divide the plan into \(l\) parts evenly along all parameters simultaneously. As for the Johnson problem, being well solvable in the case of two machines, it could be solved in no way in polynomial time in the three-machine case. (It was not yet known in 1973 that the problem was NP-hard.) So, I was advised to try solving it approximately. Besides that, the following interesting question was still open: how much can be the difference between the permutation optimum and the global one?

2.2.

While solving the VCP-problem, I hit upon the following idea of how one could ensure the uniform partition of the plan. Let \(T = \sum_{j=1}^{n} t_j\) and let \(\pi = (\pi_1, \ldots, \pi_n)\) be a permutation of integers \((1, 2, \ldots, n)\). Sum the vectors \(\{t_j\}\) one by one for \(j = \pi_1, \pi_2, \ldots, \pi_n\) and then consequently connect the points \(0, t_{\pi_1}, t_{\pi_2}, \ldots, t_{\pi_n} = T\), where \(t_{\pi_i}^k = \sum_{j=1}^{k} t_{\pi_j}\). The resulting polygonal path is called “the summing trajectory of vectors \(\{t_j\}\) according to the permutation \(\pi\)”. Let this trajectory be close to the straight line \(\{\lambda T, \lambda \in \mathbb{R}\}\) all the way along. In this case we can obtain the desired uniform partition by splitting the trajectory into \(l\) pieces by planes \(P_1, \ldots, P_{l-1}\) orthogonal to \(T\) and passing through the points \((1/l)T, (2/l)T, \ldots, ((l-1)/l)T\).

Now, to obtain a suitable summing trajectory of vectors \(\{t_j\}\), let us project them on the plane \(P_0\) in parallel with vector \(T\) (\(P_0\) is orthogonal to \(T\) and passes through the origin). The sum of the projections \(\{t_j^*\}\) is evidently equal to zero. We obtain the desired permutation \(\pi\) by finding a summing trajectory of the vectors \(\{t_j^*\}\), close to the origin all its way along. It is clear that the trajectory would be \(\kappa\) times farther from the origin if all vectors \(\{t_j^*\}\) were \(\kappa\) times longer. To be independent of this, we will estimate the proximity of the trajectory to the origin in terms of the maximum vector length. Thus, we come to the following “compact vector summation” problem (CVS).

**CVS-problem.** Let \(\mathbb{R}_+^m\) denote the \(m\)-dimensional space with a norm \(s\), and let \(\{x_1, \ldots, x_n\} \subset \mathbb{R}_+^m\) be a family of vectors such that \(\sum x_j = 0\), \(||x_j||_s \leq 1\) \(\forall j\) (such a family will be called an “\(s\)-family”). We wish to find a summing trajectory of vectors \(\{x_j\}\) close to the origin in the norm \(s\) all the way along. In other words, it is required to compute the permutation \(\pi = (\pi_1, \ldots, \pi_n)\) minimizing the functional

\[
C(\pi) = \max_{k=1, \ldots, n} \left\| \sum_{j=1}^{k} x_{\pi_j} \right\|_s.
\]

Since the reduction of the VCP-problem to the CVS-problem was not strict, we did not need to find the optimal solution of the CVS-problem. So, I was quite satisfied with having proved that it is possible to sum the vectors \(\{x_j\}\) within a ball of radius bounded from above by a certain function \(\psi(m)\) of space dimension. The desired permutation \(\pi\) was obtained in polynomial time. (Namely, in \(O(n \log n)\) steps for fixed \(m\). By the way, the complexity of the algorithm remains best until now among the
algorithms of summing the vectors within a ball of a limited radius.) This immediately
yields the algorithm of this complexity for constructing an approximate solution $S$ of
the VCP-problem with absolute performance guarantee independent of the number of
items ($n$):

$$
0 \leq L_d(S^{opt}) \leq L_d(S) \leq \psi_1(m) \cdot \max_j \|t_j\|
$$

where $\psi_1(m)$ is a function of $m$.

2.3.

Later on, the following idea crossed my mind. I thought, why should I not try to
apply the CVS-problem to the FS-problem? To my surprise, the attempt was success-
ful enough. It turned out that we could obtain a good permutation schedule $S_\pi$ by
ordering the jobs in such a way that for any $k = 1, \ldots, n$ the first $k$ jobs in this order
provide a near-equal load of all machines. (From now on, $S_\pi$ denotes a permutation
schedule, i.e. that one in which $n$ jobs pass through each machine in the same order
$\pi = (\pi_1, \ldots, \pi_n)$.) And this brings us again to the problem of finding a permutation
$\pi = (\pi_1, \ldots, \pi_n)$ of vectors $\{t_j = (t_{j1}, \ldots, t_{jn}), j = 1, \ldots, n\}$ for which the corresponding
summing trajectory is close to the straight line $\{\lambda T, \lambda \in \mathbb{R}\}$. Applying again the
"compact vector summation" algorithm to a certain $s$-family of vectors, we obtain
with this complexity an approximate permutation schedule $S_\pi$ which differs from the
optimum by at most

$$
F(S_\pi) - F(S^{opt}) \leq \psi_2(m)K,
$$

and this amount is independent of the number of jobs. (From now on, $K = \max_{j,i} t_{ij}$.)
This immediately implies that the permutation optimum differs from the global one
by at most this very amount.

These results were announced in the Proceedings of the All-Union Conference in
Theoretical Cybernetics (June, 1974, Novosibirsk) [30] and then completely pub-
lished in [31].

So, due to the successful application of the CVS-problem to two rather different
problems, I came to believe in its strength and universality. Later on, my belief got
stronger, as I learnt new interesting information about this problem.

2.4.

Firstly, it was soon found out that at the same time Belov and Stolin [10] were
following the same way in solving the FS-problem, which meant that the way was
natural. (Some time later, the connection between the FS- and the CVS-problems was
rediscovered by Bárany [7].) Secondly, three other papers by Belov with co-authors
[4, 2, 3] appeared later on, where the CVS-problem was applied to other scheduling
problems. Furthermore, I learnt that the CVS-problem had been successfully employed in functional analysis since Steinitz proved in 1913 the feasibility of a vector summation within a ball of a limited radius (the result is known since as the Steinitz lemma [47]). Finally, it became known that finding the minimum radius was the subject of several generations of researchers. In particular, the bound
\[ C(\pi^{op}) \leq \sqrt{4^m - 1}/3 \]
derived in [31] was actually proved in 1931 by Bergström [11] and again in 1953 by Kadec [24].

2.5.

The short review of the above results shows that they can develop in three directions.

The first one is the search for other scheduling problems which could be solved well with the help of the CVS-technique.

Secondly, since it appears that the scheduling problems can be reduced to the CVS-problem in different ways, the search for the most perfect reduction scheme (that would guarantee the least upper bound on the schedule length for a fixed function \(C_{x,m}^A\)) seems to be urgent.

Finally, it is essential to develop the methods of solving the CVS-problem itself subject to decreasing both the bound on the radius \(C_{x,m}^A\) and the complexity of the algorithm \(A\). Furthermore, as you will see soon, the compact vector summation within a ball is not the only problem on the subject. Methods of the compact vector summation within certain other regions of the space also need developing.

More detailed description of these three directions is presented in Section 3. And now let us return to the VCP-problem from which it all began.

2.6.

It became clear soon that the demand of the summing trajectory to be close to the straight line \(\{\lambda T, \lambda \in \mathbb{R}\}\) all the way along was redundant. In fact, it is sufficient for the VCP-problem that only some vertices of the trajectory lie close to the points \((1/l)T, (2/l)T, \ldots, ((l-1)/l)T\). And we can satisfy this reduced demand with better accuracy. The following "nearest vertex"-problem (NV) is to help us in this matter.

**NV-problem.** Suppose we are given a collection of vectors \(t_1, \ldots, t_m \in \mathbb{R}^m, \|t_j\|_s \leq 1, \forall j\), specifying the parallelepiped \(P = \{\sum \lambda_j t_j | \lambda_j \in [0, 1], j = 1, \ldots, m\} \in \mathbb{R}^m\). Given a point \(x \in P\), it is required to find the vertex \(v \in P\) nearest to \(x\) in the norm \(s\).

While speaking of complexity, the problem must be NP-hard and the search for its optimal solution requires, perhaps, looking through all \(2^m\) vertices. Yet an interesting
problem arises when we wonder what the distance from \( x \) to the nearest vertex of \( P \) can be in the worst case. It must be clearly a certain function of the norm \( s \) and the dimension \( m \) (let us denote it by \( \theta(s,m) \)). It is easily computed for the Euclidean norm \( \theta(s,m) = \sqrt{m/2} \) but already for the \( l_\infty \)-norm the problem of computing the function \( \theta(s,m) \) seems to be serious. The attempts to find bounds on this function lead us to the problem of existence of the Hadamard matrices and to other interesting problems, which is discussed in more detail in [41]. There is also an evident connection between the NV-problem and so called “balancing problems” considered by a number of authors (e.g. by Beck and Fiala [9]). This however requires a separate discussion, so let us revert to the VCP-problem.

To obtain its approximate solution, it is sufficient to solve the NV-problem approximately as well, e.g. for any norm \( s \) we can find (in \( \mathcal{O}(m) \) steps) a vertex \( v \) with \( \| v - x \|_s \leq m/2 \). This yields the approximate solution \( S \) of the VCP problem with bound

\[
L_s(S) \leq 1.00023m \cdot \max_j \| t_j \|_s.
\]

(For \( l \leq 250 \) the coefficient 1.00023 may be replaced by 1, see [41].) At the same time, the compact vector summation technique and the best bounds on \( C_s \) known today yield only the bound \( L_s(S) \leq 2(m - 1 + 1/m) \cdot \max_j \| t_j \|_s \). Thus, any further progress in the VCP-problem depends mainly on whether or not we succeed in constructing more precise polynomial algorithms for the NV-problem in the space \( \mathbb{R}^m \) with various norms (the \( l_\infty \)-norm, in particular).

3. The main ideas and results

Let us characterize the results in accordance with the three directions described in Section 2.5.

3.1.

The first direction was under the great influence of papers [2–4] already mentioned in Section 2.4. The most impressive was paper [2] where the CVS-problem was used for solving the Akers and Friedman problem (AF) in the case \( m = 3 \). To formulate the latter, we specify the parameters of the G-problem as follows.

**AF-problem** [1]. For any job, \( j = 1, \ldots, n, r_j \equiv m; R_j = \{(o^j_k, o^j_{k+1}), k = 1, \ldots, m - 1\}; J'_i = \{p'_i\}, i = 1, \ldots, m; \bigcup_{i=1}^{m} J'_i = \{1, 2, \ldots, m\}.

Thus, each job \( j \) follows a given route \((p'_1, p'_2, \ldots, p'_m)\) being a permutation of machines \( \{1, 2, \ldots, m\} \). Different jobs may have different routes.
Given the clear connection between the FS-problem and the CVS-problem (the vector permutation \( \pi \) coincides with the job permutation on each machine and thereby specifies a permutation schedule \( S_k \)), it is not the case for the AF-problem. Since we can have distinct job processing orders on different machines in this problem, it is not so simple to perceive where the vector permutation is hidden here. Paper [2] just contained the crucial idea in this direction.

Later on, an algorithm with much better performance guarantee and the same complexity was constructed for the problem based on the same idea (cf. Nos. 11 and 37 in Table 1). And although that algorithm could not be extended to the case \( m > 3 \), the result in [2] produced the confidence that a similar result is possible for the AF-problem in the general case, too. Really, the desired polynomial-time algorithm (based on another idea) was soon constructed which ensured a solution with accuracy bound polynomial of \( m \) for not only the AF-problem but also for the job shop problem and even for the most general G-problem, see Nos. 45, 46, 48–50 in Table 1. (To obtain the job shop problem, denoted by JS, from the G-problem, it suffices to set \( |D_i| = 1 \) and \( R_j = \{ (o_k^j, o_{k+1}^j), k = 1, \ldots, r_j - 1 \} \), \( \forall i, j \).) At the same time, an interesting property of the JS optimum was obtained. It turned out that for any instance of the JS-problem its optimum lay within an a priori known interval \( I_{JS} = [M, M + \mu_{JS}(m, r)K] \) with length independent of the number of jobs \( n \) and polynomial of \( m \) and \( r \). (From now on, \( M \) denotes the maximum machine load:

\[
M = \max_{k=1, \ldots, m} \sum_{o_{ij} \in \{p_j = k\}} t_{ij}^k; \ r = \max_j r_j.
\]

This gives rise to the natural question of finding the minimal such intervals for the G, JS, AF, FS and other scheduling problems with this property. Yet this question regards the second direction, and we will discuss it some later.

A few special cases of the G-problem being of self-dependent interest were provided with their own algorithms with considerably better performance guarantees (being compared with No. 50). As it has already been mentioned, one of the problems was the FS-problem. Let us formulate the others that generalize the FS-problem in various directions. (For the corresponding results see the Table 1.)

**AF\(_p\)-problem** is the special case of the AF-problem and generalizes the FS-problem to the case of \( p \) different routes of jobs through machines.

**AF\(_{2*}\)-problem** is the special case of the AF\(_2\)-problem, when the two routes are the counter ones: \((1, 2, \ldots, m)\) and \((m, m-1, \ldots, 1)\).

**FS\(_{1}\)-problem** generalizes the FS-problem to the case of \( l_i \) identical machines of the \( i \)-th type \((i = 1, 2, \ldots, m)\) so that the \( i \)-th operation of a job can be processes by any of these machines.
<table>
<thead>
<tr>
<th>Year</th>
<th>Authors [reference]</th>
<th>No.</th>
<th>Problem (param.)</th>
<th>Result</th>
<th>Algorithm complexity</th>
</tr>
</thead>
<tbody>
<tr>
<td>1953</td>
<td>Kadeč [24]</td>
<td>1</td>
<td>CVS</td>
<td>( C_{1,1}^A = \sqrt{(4^n - 1)/3} )</td>
<td>( mn^m )</td>
</tr>
<tr>
<td>1974</td>
<td>Sev. [30]</td>
<td>2</td>
<td>CVS</td>
<td>( C_{1,1}^A \leq \psi(m) )</td>
<td>( n \log n )</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3</td>
<td>FS</td>
<td>( F(S) - F(S^{opt}) \leq \psi_1(m)K )</td>
<td>( n \log n )</td>
</tr>
<tr>
<td></td>
<td></td>
<td>4</td>
<td>VCP</td>
<td>( L(S) - L(S^{opt}) \leq \psi_2(m) )</td>
<td>( n \log n )</td>
</tr>
<tr>
<td></td>
<td>Belov, Stclin [10]</td>
<td>5</td>
<td>FS</td>
<td>( M \leq F(S^{opt}) \leq F(S) ) ( \leq M + (m - 1)^{3/2}C_{1,2}^A \leq m - 1)K )</td>
<td>( T(A) )</td>
</tr>
<tr>
<td>1975</td>
<td>Sev. [31]</td>
<td>6</td>
<td>FS</td>
<td>( F(S_A) \leq M + (\sqrt{2m(m-1)} \cdot C_{1,2}^A + m - 1)K )</td>
<td>( T(A) )</td>
</tr>
<tr>
<td></td>
<td></td>
<td>7</td>
<td>VCP</td>
<td>( L(S) \leq 24(C_{1,2}^A)^2 + 1)^{1/2} )</td>
<td>( T(A) )</td>
</tr>
<tr>
<td></td>
<td></td>
<td>8*</td>
<td>CVS</td>
<td>( C_{1,1}^A = (1 - (3N + 2)/(4N^2 + 6N + 2)) \cdot (2 + 1/N)^m ), ( V N \geq 1 ), e.g. ( C_{1,1}^A \approx \tau^2 \cdot 2^m ) (for ( N = 1 ))</td>
<td>( Nm^2n \log n )</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>( C_{1,1}^A \approx \sqrt{e} \cdot 2^m ) (for ( N = m ))</td>
<td>( m^2n \log n )</td>
</tr>
<tr>
<td></td>
<td></td>
<td>9</td>
<td>CVS</td>
<td>The same as No. 1</td>
<td>( m^3n \log n )</td>
</tr>
<tr>
<td></td>
<td></td>
<td>10</td>
<td>CVS</td>
<td>( C_{1,2}^A = \sqrt{3} )</td>
<td>( n^2 )</td>
</tr>
<tr>
<td>1976</td>
<td>Babuskin, Basta, Belov [2]</td>
<td>11</td>
<td>AF(3)</td>
<td>( M \leq F(S^{opt}) \leq F(S) \leq M + 3(3\sqrt{2} + 5)K )</td>
<td>( n^2 )</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>( &lt; M + 28K )</td>
<td>( n^2 )</td>
</tr>
<tr>
<td></td>
<td>Stolin [48]</td>
<td>12</td>
<td>FS(3)</td>
<td>( F(S_A) \leq M + 4K )</td>
<td>( n^2 )</td>
</tr>
<tr>
<td>1977</td>
<td>Fiala [19]</td>
<td>13</td>
<td>FS(3)</td>
<td>( F(S_A) \leq M + 5K )</td>
<td>( n )</td>
</tr>
<tr>
<td></td>
<td>Babuskin, Basta, Belov [3]</td>
<td>14</td>
<td>AF_2^c</td>
<td>( F(S) \leq M + 3^m - (7 + C_{1,2}^A)K )</td>
<td>( T(A) )</td>
</tr>
<tr>
<td>Year</td>
<td>Authors [reference]</td>
<td>No.</td>
<td>Problem (param.)</td>
<td>Result</td>
<td>Algorithm complexity</td>
</tr>
<tr>
<td>------</td>
<td>----------------------</td>
<td>-----</td>
<td>------------------</td>
<td>--------</td>
<td>----------------------</td>
</tr>
<tr>
<td>1978</td>
<td>Sev. [32]</td>
<td>15</td>
<td>CVS</td>
<td>$C_{s,m} = m$</td>
<td>$m^{2n}n^2$</td>
</tr>
<tr>
<td></td>
<td></td>
<td>16</td>
<td>FS</td>
<td>$F(S_x) \leq M + ((m - 1)^{3/2} + m - 1)K$</td>
<td>$m^{2n}n^2$</td>
</tr>
<tr>
<td></td>
<td></td>
<td>17</td>
<td>VCP</td>
<td>$L_0(S) \leq \sqrt{4(m - 1)^2 + 1}$</td>
<td>$m^{2n}n^2$</td>
</tr>
<tr>
<td>1979</td>
<td>Sev. [33]</td>
<td>18</td>
<td>CVS(2)</td>
<td>$C_{i,j} = \sqrt{2}$</td>
<td>$n^2$</td>
</tr>
<tr>
<td>1980</td>
<td>Sev. [35]</td>
<td>19</td>
<td>FS</td>
<td>$F(S_x) \leq M + (m - 1)(C_{s,m-1} + 1)K$</td>
<td>$T(A)$</td>
</tr>
<tr>
<td></td>
<td></td>
<td>20</td>
<td>CVS</td>
<td>$C_{s,m} = m$</td>
<td>$m^2n^2$</td>
</tr>
<tr>
<td></td>
<td></td>
<td>21</td>
<td>FS</td>
<td>$F(S_x) \leq M + m(m - 1)K$</td>
<td>$m^2n^2$</td>
</tr>
<tr>
<td></td>
<td>Dusin [16]</td>
<td>22*</td>
<td>VCP</td>
<td>$L_2(S) \leq 1.06\sqrt{m}$</td>
<td>$m^4l + nm^2\log l$</td>
</tr>
<tr>
<td>1981</td>
<td>Bárány [7]</td>
<td>23</td>
<td>FS</td>
<td>$F(S_x) \leq M + (m - 1 + 2(m - 2)^{3/2} \cdot C_{i,j-2})K$</td>
<td>$T(A)$</td>
</tr>
<tr>
<td></td>
<td></td>
<td>24</td>
<td>FS(3)</td>
<td>$F(S_x) \leq M + 4K$</td>
<td>$n$</td>
</tr>
<tr>
<td></td>
<td></td>
<td>25</td>
<td>CVS(2)</td>
<td>$C_{i,j} = \sqrt{2}$</td>
<td>$n^2$</td>
</tr>
<tr>
<td></td>
<td></td>
<td>26</td>
<td>FS(4)</td>
<td>$F(S_x) \leq M + 11K$</td>
<td>$n^2$</td>
</tr>
<tr>
<td></td>
<td>Sev. [36]</td>
<td>27</td>
<td>FS</td>
<td>$F(S_x) \leq M + (m - 1)(C_{s,m-1} + 1)K$</td>
<td>$T(A)$</td>
</tr>
<tr>
<td></td>
<td></td>
<td>28</td>
<td>CVS</td>
<td>$C_{s,m} = \lfloor \frac{3}{2}m \rfloor$</td>
<td>$m^2n^2 + m^4n$</td>
</tr>
<tr>
<td></td>
<td></td>
<td>29</td>
<td>FS</td>
<td>$F(S_x) \leq M + (m - 1)(\frac{3}{2}m - \frac{1}{2})K$</td>
<td>$m^2n^2 + m^4n$</td>
</tr>
<tr>
<td></td>
<td>Sev. [36]</td>
<td>30*</td>
<td>CVS</td>
<td>$C_{s,m} = 1.5$</td>
<td>$n^2$</td>
</tr>
<tr>
<td></td>
<td>Sev. [37]</td>
<td>31*</td>
<td>AF</td>
<td>$F(S) \leq M + (2m^2 - (m + 3)/2)K$</td>
<td>$m^2n^2$</td>
</tr>
<tr>
<td></td>
<td></td>
<td>32</td>
<td>FS(4)</td>
<td>$F(S_x) \leq M + 9K$</td>
<td>$n^2$</td>
</tr>
<tr>
<td></td>
<td></td>
<td>33</td>
<td>FS1 ($l_i = l$)</td>
<td>$D \leq F(S^{max}) &lt; F(S) \leq D + (m^2 + 0.06)K$</td>
<td>$m^2n^2$</td>
</tr>
<tr>
<td></td>
<td></td>
<td>34</td>
<td>FS</td>
<td>$F(S_x) \leq M + (m - 1 + 2(m - 2) \cdot C_{s,m-2})K$</td>
<td>$T(A)$</td>
</tr>
<tr>
<td>Year</td>
<td>Authors [reference]</td>
<td>No.</td>
<td>Problem (param.)</td>
<td>Result</td>
<td>Algorithm complexity</td>
</tr>
<tr>
<td>------</td>
<td>---------------------</td>
<td>-----</td>
<td>------------------</td>
<td>--------</td>
<td>----------------------</td>
</tr>
<tr>
<td>1982</td>
<td>Sev. [38]</td>
<td>35#</td>
<td>FS(3)</td>
<td>( F(S_0) \leq M + 3K )</td>
<td>( n \log n )</td>
</tr>
<tr>
<td></td>
<td></td>
<td>36#</td>
<td>FS(3)</td>
<td>( \forall e &gt; 0 \exists \mu e; e(1, 2, 3, 5): F(S_m^e) &gt; M + (3 - e)K, \mu e(3) = 3 )</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>37*</td>
<td>AF(3)</td>
<td>( F(S) \leq M + (3 + 2A_{n-x}^e)K )</td>
<td>( T(A) )</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>( F(S) \leq M + 6K )</td>
<td>( n^2 )</td>
</tr>
<tr>
<td></td>
<td>Bárály, Fiala [8]</td>
<td>38</td>
<td>CVS</td>
<td>( C_{n-m} = m )</td>
<td>( n^2 m^3 )</td>
</tr>
<tr>
<td></td>
<td></td>
<td>39</td>
<td>FS</td>
<td>( F(S_S) \leq M + m(m - 1)K )</td>
<td>( n^2 m^3 )</td>
</tr>
<tr>
<td></td>
<td></td>
<td>40</td>
<td>OS</td>
<td>( M \geq (m^2 + 2m - 1)K \Rightarrow F(S) = M = F(S_n^m) )</td>
<td>( n^2 m^3 )</td>
</tr>
<tr>
<td></td>
<td></td>
<td>41</td>
<td>OS</td>
<td>( M \geq (8m \log_2 m + 5m)K \Rightarrow F(S) = M = F(S_n^m) )</td>
<td>( n m^3 )</td>
</tr>
<tr>
<td></td>
<td>Racsmayni [8, p. 179]</td>
<td>42#</td>
<td>OS</td>
<td>( F(S) \leq M + (m - 1)K, \mu _{OS}(m) = m - 1 )</td>
<td>( n m^2 )</td>
</tr>
<tr>
<td>1983</td>
<td>Fiala [20]</td>
<td>43</td>
<td>OS</td>
<td>( M \geq (16m \log_2 m + 5m)K \Rightarrow F(S) = M = F(S_n^m) )</td>
<td>( n^2 m^3 )</td>
</tr>
<tr>
<td>1984</td>
<td>Sev. [39]</td>
<td>44*</td>
<td>FS2</td>
<td>( F(S) \leq M + w(r^2 + w - 2)K )</td>
<td>( n^2 r^2 )</td>
</tr>
<tr>
<td></td>
<td></td>
<td>45</td>
<td>IS</td>
<td>( F(S) \leq M + r(m^2 r^2 + r - 7)K )</td>
<td>( n^2 r^2 m^2 )</td>
</tr>
<tr>
<td></td>
<td></td>
<td>46</td>
<td>G</td>
<td>( F(S) - F(S_n^m) \leq (m^2 r^3 + r^2 - 2r + 2)K )</td>
<td>( n^2 r^2 m^2 + (m + r)^3 \log(r n) )</td>
</tr>
<tr>
<td>1985</td>
<td>Dani'chenko a.o. [14]</td>
<td>47</td>
<td>FS(3)</td>
<td>The same as No. 12</td>
<td></td>
</tr>
<tr>
<td>1986</td>
<td>Sev. [40]</td>
<td>48*</td>
<td>JS</td>
<td>( F(S) \leq M + (r - 1)(m r^2 + 2r - 1)K )</td>
<td>( n^2 r^2 m^2 )</td>
</tr>
<tr>
<td></td>
<td></td>
<td>49*</td>
<td>AF</td>
<td>( F(S) \leq M + (m^3 + 2)(m - 1)K )</td>
<td>( n^2 m^2 )</td>
</tr>
<tr>
<td></td>
<td></td>
<td>50*</td>
<td>G</td>
<td>( F(S) - F(S_n^m) \leq (1 + (r - 1)(m r^2 + 2r - 1))K )</td>
<td>( n^2 r^2 m^2 )</td>
</tr>
<tr>
<td></td>
<td></td>
<td>51</td>
<td>FS1</td>
<td>( F(S) \leq D + ((L - 1)^2 + 2)K )</td>
<td>( n^2 L^2 )</td>
</tr>
<tr>
<td>1987</td>
<td>Banaszczyk [3]</td>
<td>52#</td>
<td>CVS(2)</td>
<td>( C_{n-x}^e \leq 3, \forall s )</td>
<td>( n^2 )</td>
</tr>
<tr>
<td></td>
<td></td>
<td>53*</td>
<td>CVS</td>
<td>( C_{n-m}^e \leq m - 1 + 1/m, \forall s )</td>
<td></td>
</tr>
<tr>
<td>Year</td>
<td>Authors</td>
<td>No.</td>
<td>Problem (param)</td>
<td>Result</td>
<td>Algorithm complexity</td>
</tr>
<tr>
<td>------</td>
<td>---------</td>
<td>-----</td>
<td>----------------</td>
<td>--------</td>
<td>---------------------</td>
</tr>
<tr>
<td>1988</td>
<td>Sev. [41]</td>
<td>54*</td>
<td>VCP ($t \geq 0$)</td>
<td>$L_1(S) &lt; 2C_m^*$</td>
<td>$n^2m^2$</td>
</tr>
<tr>
<td>1989</td>
<td>Dusin [17]</td>
<td>55*</td>
<td>VCP ($t \geq 0$)</td>
<td>$L_1(S) &lt; 10002m$</td>
<td>$T(4)$</td>
</tr>
<tr>
<td>1990</td>
<td>Sev. [42]</td>
<td>56*</td>
<td>AF, $F_C$</td>
<td>$F(S) &lt; 2M + 2mK$</td>
<td>$m^2n^2$</td>
</tr>
<tr>
<td></td>
<td>Dusin [18]</td>
<td>57*</td>
<td>AF, $F_C$</td>
<td>$C_m^* &lt; M + (pm^2 + 2m)K$</td>
<td>$m^2n^2$</td>
</tr>
<tr>
<td></td>
<td>Sev. [43]</td>
<td>58*</td>
<td>OS</td>
<td>$M &gt; (m + 1)/(m - 1)$</td>
<td>$n^2m^2$</td>
</tr>
<tr>
<td></td>
<td></td>
<td>59*</td>
<td>OS</td>
<td>$M &gt; (m + 1)/(m - 1)$</td>
<td>$n^2m^2$</td>
</tr>
<tr>
<td></td>
<td></td>
<td>60*</td>
<td>OS</td>
<td>$M &gt; (m + 1)/(m - 1)$</td>
<td>$n^2m^2$</td>
</tr>
<tr>
<td></td>
<td></td>
<td>61*</td>
<td>OS</td>
<td>$M &gt; (m + 1)/(m - 1)$</td>
<td>$n^2m^2$</td>
</tr>
<tr>
<td></td>
<td></td>
<td>62*</td>
<td>FS</td>
<td>$M &gt; (m + 1)/(m - 1)$</td>
<td>$n^2m^2$</td>
</tr>
<tr>
<td></td>
<td></td>
<td>63*</td>
<td>FS</td>
<td>$M &gt; (m + 1)/(m - 1)$</td>
<td>$n^2m^2$</td>
</tr>
<tr>
<td></td>
<td></td>
<td>64*</td>
<td>FS</td>
<td>$M &gt; (m + 1)/(m - 1)$</td>
<td>$n^2m^2$</td>
</tr>
<tr>
<td></td>
<td></td>
<td>65*</td>
<td>FS</td>
<td>$M &gt; (m + 1)/(m - 1)$</td>
<td>$n^2m^2$</td>
</tr>
<tr>
<td></td>
<td></td>
<td>66*</td>
<td>FS</td>
<td>$M &gt; (m + 1)/(m - 1)$</td>
<td>$n^2m^2$</td>
</tr>
<tr>
<td></td>
<td>Banaszczyk [6]</td>
<td>70*</td>
<td>CVS, [2]</td>
<td>$M &gt; (m + 1)/(m - 1)$</td>
<td>$n^2m^2$</td>
</tr>
<tr>
<td>Year</td>
<td>Authors</td>
<td>No.</td>
<td>Problem (param.)</td>
<td>Result</td>
<td>Algorithm complexity</td>
</tr>
<tr>
<td>-------</td>
<td>------------------</td>
<td>-----</td>
<td>------------------</td>
<td>-----------------------------</td>
<td>----------------------</td>
</tr>
<tr>
<td>1991</td>
<td>Neumyтов, Sev.</td>
<td>71 #</td>
<td>AF₂*(3)</td>
<td>( F(S) \leq M + 3K )</td>
<td>( n \log n )</td>
</tr>
<tr>
<td></td>
<td></td>
<td>72*</td>
<td>OS(3)</td>
<td>( M \geq 7K \implies F(S) = M = F(S^{opt}) )</td>
<td>( n^2 )</td>
</tr>
<tr>
<td></td>
<td></td>
<td>73*</td>
<td>OS1 (( tI \equiv 1 ))</td>
<td>( S^{opt} )</td>
<td>( R^2m^2 + R^3n^{3/2} )</td>
</tr>
<tr>
<td></td>
<td></td>
<td>74*</td>
<td>OS1 (( I,J \equiv 1 ))</td>
<td>( F(S) \leq M + (r - 1)K )</td>
<td>( R + nm^2 )</td>
</tr>
<tr>
<td></td>
<td>Strusevitch</td>
<td>75*</td>
<td>OS1</td>
<td>( F(S) - \tilde{r}(S^{opt}) \leq rK )</td>
<td>( R^2m^2 )</td>
</tr>
<tr>
<td></td>
<td></td>
<td>76*</td>
<td>AL</td>
<td>( F(S) \leq M + (1 + C_{d,n-1})K )</td>
<td>( T(A) )</td>
</tr>
<tr>
<td></td>
<td></td>
<td>77 #</td>
<td>AL(3)</td>
<td>( F(S) \leq M + 1.25K )</td>
<td>( n \log n )</td>
</tr>
</tbody>
</table>
**FS2-problem** \((w\text{-cyclic route }[15])\): all jobs have the same machine passage route \((1, 2, \ldots, m, 1, 2, \ldots, m)\) which is the \(w\)-times iterated route \((1, 2, \ldots, m)\).

**AL-problem** (assembly line) differs from the FS problem on the points:
1. \(R_j = \{(o_i^k, o_i^m), k = 1, \ldots, m - 1\}\) for each job \(j\); thus, the "machining" operations \(\{o_1^1, \ldots, o_{m-1}^m\}\) may be processed on machines \(\{1, 2, \ldots, m - 1\}\) independently and in parallel, and after they are all completed, the details must be assembled on the assembly line (\(m\) machine). So,
2. the property (*) may be violated for any pair of machining operations.

**OS-problem** (open shop \([21]\)) is the special case of the G-problem when \(r_j = m, J^j_i = \{i\}, R_j = \emptyset, \forall j = 1, \ldots, n; i = 1, \ldots, m\).

**OS1-problem** arises from the G-problem when \(R_j = \emptyset, \forall j\).

There is a fine approximation algorithm for the OS-problem suggested by Racsmany. (See its parameters in Table 1, No. 42.) Yet our discussion about the OS-problem will concern another question also connected with the compact vector summation.

In \([20]\) Fiala observed an interesting property of the OS-problem. It turns out that there exists a function \(\eta(m)\) such that whenever an instance of the \(m\)-machine OS-problem satisfies
\[
M \geq \eta(m) K
\]
(which is inevitable for increasing number of jobs), then \(F(S^{opt}) = M\), and the optimal schedule can be constructed in polynomial time. This implies the natural question of finding the minimum function \(\eta(m)\) with this property.

There are two different ways to obtain upper bounds on \(\eta(m)\) and to construct the optimal schedule of OS provided the condition (1) holds. One of the two, suggested by Fiala \([20]\), makes use of solving a problem similar to the VCP-problem. A collection of vectors should also be divided there uniformly into \(m\) subcollections. The vectors being specific (each vector has at most two nonzero items), the graph-theory language proved to be convenient. This was improved then in \([8]\) and \([43]\) (see Nos. 43, 41 and 60). The other way suggested in \([8]\) and improved then in \([44]\) uses a reduction of the OS-problem to the CVS-problem. It yields worse (by the order of magnitude) bounds on the function \(\eta(m)\) (see Nos. 40 and 65) still improving the bounds of the first method for small values of \(m\).

The compact vector summation technique was also applied to some scheduling problems not reducible to the G-problem. Yet to avoid representing special notations, we will neither state them completely here nor characterize their algorithms in the table referring the reader to the corresponding papers instead. These problems generalize the FS-problem in various ways. These are:
The problem on parties of jobs [26] where the set of jobs is divided into parties so that jobs of each party should visit each machine in succession, without mixing any job of another party. For an approximation algorithm see [37].

The problem ([26]) with an extended interpretation of the relation \((o_i^t, o_{i+1}^t) \in R_j\) (the property (*) can be violated here). Furthermore, two time lags are specified for each operation \(o_i^t\): to prepare the \(i\)th machine for processing the operation and to return it to the initial state after the operation is completed. Mitten's [27] and Nabeshima's [28] problems are the special cases of that one. For an approximation algorithm see [37].

The assembly line problem [4] where each machine if started must work without an idle time, the interpretation of the relation \((o_i^t, o_{i+1}^t) \in R_j\) is specific and the property (*) can be violated. Approximation algorithms can be found in [4] and [37].

3.2.

As for the second direction, the FS-problem was its object for a long time as the simplest and most convenient model. After several improvements (see Nos. 5, 6, 19, 23, 27, 34 and 61), a polynomial-time approximation algorithm for the \(m\)-machine FS-problem was constructed [44] whose absolute performance guarantee depended on the radius we could assure for the CVS-problem in \((m-2)\)-dimensional space:

\[ M < F(S_{\text{opt}}) \leq F(S_a) \leq M + (m - 1)(C_{m-2} + 1)K \]

(the norm \(s_a\) is defined in Section 6.10). For \(m = 3\) Dushan's algorithm of complexity \(O(n)\) constructs a schedule \(S_a\) with bound \(F(S_a) \leq M + 4K\) (see No. 24). Yet the best possible performance guarantee belongs to the algorithm that reduces the FS\((m = 3)\)-problem to a so called "unstrict vector summation" within the angle \(Q = \{(x, y) \in \mathbb{R}^2 | x \leq 0, y \leq 1\}\) in the plane. (The input collection of vectors belongs to \(B_{x,y,2}\), where \(B_{x,y,2}\) denotes the unit ball in \(\mathbb{R}^m\).) The "unstrict vector summation" can be defined as follows.

Suppose a polyhedron \(P = \{z \in \mathbb{R}^m | (z, g_i) \leq c_i, i = 1, \ldots, l\} \subset \mathbb{R}^m\) is specified by \(l\) hyperplanes (where \(g_i \in \mathbb{R}^m, c_i \in \mathbb{R}, i = 1, \ldots, l\)), \{\(x_1, \ldots, x_n\}\} is an input vector family in \(\mathbb{R}^m\). We say that a permutation \(\pi = (\pi_1, \ldots, \pi_n)\) determines an unstrict summation of vectors \(\{x_i\}\) within \(P\) if any inequality \((\sum_{i=1}^k x_{\pi_i}, g_v) > c_v\) for some \(k\) and \(v\), implies the inequality \((\sum_{i=1}^{k+1} x_{\pi_i}, g_v) \leq c_v\).

Thus, whenever one of \(l\) hyperplanes specifying \(P\) is crossed by the "unstrict" summing trajectory in the inadmissible direction then it must cross it backwards the next step. Note that it is not necessary for the trajectory to return to the polyhedron \(P\) every time. Moreover, it may happen that no vertex of the trajectory belongs to \(P\).

I am sure that further progress in FS\((m)\) will be connected with constructing algorithms of unstrict vector summation within an \(m\)-dimensional angle.

The current best algorithm for the AF-problem is the one described in [40] (see No. 49). Yet for \(m = 3\) it yields only the bound \(F(S) \leq M + 58K\), whereas the algorithm [38] with the same running time assures the considerably better bound on
schedule length:

\[ F(S) \leq M + 6K. \] (2)

I guess, this algorithm, being generalized somehow to the case of arbitrary \( m \), could improve the current bound (No. 49) considerably. Moreover, the bound (2) does not seem to be tight (we have so far the same lower bound on the function \( \mu_{AF}(3) \) as on \( \mu_{FS}(3) \): \( \mu_{AF}(3) \geq 3 \)). It, probably, could be improved by means of unstrict summation of vectors from \( B_{n,2} \) within a stretched ball \( \lambda B_{n,2} \) with \( \lambda \) as small as possible. Given \( \lambda = 1.5 \) for the strict summation algorithm [36] (see No. 30), we may expect the lesser value of \( \lambda \) for the unstrict summation.

The current best algorithm for the \( AF^*_n \)-problem is No. 31. Yet for \( m = 3 \) it assures only the bound \( F(S) \leq M + 15K \), while the better one yields \( F(S) \leq M + 3K \) (see No. 71). Perhaps, the problem can be reduced to a CVS-type problem in a more suitable way in the general case, too.

Finally, the most appropriate method for estimating the value \( \eta(3) \) for the OS \( (m = 3) \)-problem is based not on the compact vector summation within a ball but on the unstrict summation of vectors from \( B_{n,2} \) within a triangle with sides parallel to three odd (even) sides of the hexagon \( B_{n,2} \), a location of the triangle in the plane being not fixed in advance. The method yields the bound \( \eta(3) \leq 4 + T \) where \( T \) is the length of each side of the triangle in the norm, \( s_{\infty} \) (they all coincide). We can sum the vectors unstrictly within a triangle with sides \( T = 3 \), which already ensures the bound improving No. 65 for \( m = 3 \). And it seems very likely that \( T \) can be diminished.

3.3. Finally, a few words about the third direction.

The radius \( C_{s,m}^A \) was first decreased in [32] to \( m \) for any norm \( s \), the algorithm complexity being polynomial of \( n \) and exponential of \( m \) (see No. 15). Then in [35] the complexity was diminished to \( O(m^2n^2) \). (Later, the paper [8] appeared with worse complexity, cf. No. 38.) Finally, it was announced in [5] that the radius can be diminished to \( C_{s,m}^A = m - 1 + (1/m) \) for any symmetric norm \( s \). A polynomial algorithm that realizes this bound in time \( O(m^2n^2) \) can be found in [42]. It is also shown there that the bound remains true for any asymmetric norm whose unit ball is symmetric (not necessary relatively to the origin).

4. Further research directions

A progress in the following directions will enable us to improve the current performance guarantees of the approximation algorithms, to obtain interesting properties of the optimal solutions and to extend the classes of polynomially solvable cases for the scheduling problems in question.
4.1.

To find the minimum \( \lambda \) which for any \( s_\alpha \)-family in \( \mathbb{R}^m \) admits its unstrict summation within an \( m \)-dimensional angle

\[
P_1(\vec{\lambda}) = \{ x = (x^1, \ldots, x^m) \in \mathbb{R}^m \mid x^i - x^{i+1} \leq \lambda_i, i = 1, \ldots, m - 1; x^m \leq \lambda_m \}
\]

specified by a vector \( \vec{\lambda} = (\lambda_1, \ldots, \lambda_m) \in \mathbb{R}^m \) with \( \sum \lambda_i = \lambda \). This promises significant improvement of the accuracy bound for the FS\((m + 1)\)-problem.

4.2.

To find the minimum \( \lambda \) which for any \( s_\alpha \)-family in \( \mathbb{R}^2 \) admits its unstrict summation within \( \lambda \) times stretched unit ball \((\lambda \mathcal{B}, 0, \mathcal{Z})\).

This ensures better performance guarantees of the approximation algorithm for the AF\((m = 3)\)-problem.

4.3.

To generalize the approximation algorithm for the AF\((m = 3)\)-problem [38] to the case of arbitrary \( m \).

4.4.

To find the minimum \( \lambda \) which for any \( s_\alpha \)-family in \( \mathbb{R}^m \) admits its unstrict summation within a simplex

\[
P_2(\vec{\lambda}, \pi) = \{ x \in \mathbb{R}^m \mid -x^1 \leq \lambda_1; x^{ni} - x^{ni+1} \leq \lambda_i, i = 1, \ldots, m - 1; x^m \leq \lambda_m \}
\]

specified by a vector \( \vec{\lambda} = (\lambda_1, \ldots, \lambda_m) \in \mathbb{R}^m \) with \( \sum \lambda_i = \lambda \) and a permutation \( \pi = (\pi_1, \ldots, \pi_m) \) of indices. (Both \( \vec{\lambda} \) and \( \pi \) are not fixed in advance.) This ensures the optimal schedule length \( M \) for the OS\((m + 1)\)-problem as soon as its instance satisfies the condition \( M \geq (2m + \lambda)K \).

4.5.

To improve the bound on the radius in the CVS-problem for the case of \( l_\alpha \)-family of "sparse" vectors in \( \mathbb{R}^m \) (with at most \( k \) nonzero items). This improves the performance guarantees of the approximation algorithms for the G- and the JS-problems.

4.6.

To generalize the algorithm No. 71 for the AF\( \alpha^2(m = 3) \)-problem to the case of arbitrary \( m \), which promises improving the bound No. 31.
4.7.

To improve the parameters of the algorithm No. 59 for the CVS-problem in the space \( \mathbb{R}^m \), which ensures improvements in many scheduling problems considered above.

5. Some open questions

5.1.

What are the minimum functions \( \mu_{FS}(m) \) and \( \tilde{\mu}_{FS}(m) \) such that the optimum and the permutation optimum of any instance of the FS-problem correspondingly belong to the intervals:

\[
F(S_{opt}) \in [M, M + \mu_{FS}(m)K],
\]

\[
F(S_{opt}) \in [M, M + \tilde{\mu}_{FS}(m)K].
\]

It is known that \( \tilde{\mu}_{FS}(2) - \mu_{FS}(2) = 1, \tilde{\mu}_{FS}(3) = \mu_{FS}(3) = 3, \)

\[
m - 1 + \left\lfloor \frac{m - 1}{2} \right\rfloor \leq \tilde{\mu}_{FS}(m) \leq m^2 - 3m + 3 + \frac{1}{m - 2}.
\]

5.2.

What are the minimum functions \( \mu_X(m), X \in \{AF, AF_2, AF^*_2\} \), such that for any \( m \)-machine instance of an \( X \)-problem, its optimum belongs to the interval \( I^*_X = [M, M + \mu_X(m)K] \)?

What is the similar function \( \mu_{JS}(m, r) \) for the \( m \)-machine JS-problem with each job having at most \( r \) operations?

Known bounds on the functions \( \mu_X \) can be found in Table 1.

5.3.

We know the minimal intervals \( I^*_X \) for the problems \( X = FS(m \leq 3) \) and \( X = OS \). For this problems we can also construct in polynomial time near-optimal schedules \( S \) with values \( F(S) \in I^*_X \). For some other problems \( X \) we have polynomial approximation algorithms producing schedules \( S \) with length from a certain interval \( I_X \) (\( F(S) \in I_X \)). Yet we know no scheduling problem \( X \) such that constructing a schedule \( S \) of length \( F(S) \in I^*_X \) would be NP-hard. Does there exist such a problem?

5.4.

Does the complexity of finding a schedule \( S \) with bound \( F(S) \leq M + \mu'K \) increases gradually as the value \( \mu' \) decreases, or are there certain threshold values of \( \mu' \) where
the complexity jumps up? For example, the schedule $S$ with bound $F(S) \leq M + 4K$ for the $FS(m = 3)$-problem can be found in $O(n)$ steps, while we cannot find a schedule $S$ with bound $F(S) \leq M + 3K$ faster than in $O(n \log n)$ steps.

5.5.

What is the minimum function $\eta(m)$ such that if an instance of the $OS(m)$-problem satisfies $M \geq \eta(m)K$ then it immediately implies the property

$$F(S_{opt}) = M?$$

The reader can find some lower and upper bounds on $\eta(m)$ in the table.

5.6.

It is shown in [44] that for any $\lambda \in (1, 2m - 3)$, $m \geq 3$, the verification of the property (3) within the class of instances of the $OS(m)$-problem with property $M = \lambda K$ is NP-hard. At the same time, (3) always holds for $\lambda \geq m^2$. The question is: what is the maximum function $\eta'(m)$ for which the verification of the property (3) within the class of instances $OS(m)| M = \eta'(m)K$ is NP-hard?

6. Some comments on Table 1

6.1.

Table 1 contains (not all) results on the theme we are discussing. Firstly, we did not include in it the results concerning applications of the CVS-technique to those scheduling problems that have unwieldy settings. Secondly, we have granted no place in the table to those vector summation algorithms which have so far no applications to scheduling problems. Thirdly, the results concerning bounds on the Steinitz function (i.e. the minimum radius of the ball containing a summing trajectory of any $s$-family of vectors in the CVS-problem) are omitted here as well. The reader can find them in [41]. Yet I hope, the rest of the results (those given in the table) are sufficient for the reader to get the whole notion about this research direction.

6.2.

The results are numbered chronologically. Column 1 indicates the year when the result has been published or submitted or obtained (for not submitted papers).
6.3.

The long name of the author is reduced to "Sev." in Column 2.

6.4.

The current best performance guarantees (for fixed running time) are marked by the asterisk in Column 3. The best possible bounds (that cannot be improved in the same terms) are marked by the sharp instead.

6.5.

If the abbreviation of a scheduling problem has a constant in parentheses (in Column 4), this means the number of machines. An abbreviation without parameters denotes a problem with arbitrary number of machines.

6.6.

Column 5 (tilted as "results") usually shows an algorithm performance guarantee. Yet sometimes (namely, for Nos. 36, 40, 41, 43, 60, 63-67 and 69) it contains results of another type.

6.7.

The function in Column 6 shows the algorithm complexity. (For example, a function $\psi(m, n)$ means that algorithm complexity is $O(\psi(m, n))$.)

6.8.

If an approximate solution of a problem was obtained by reducing the latter to the CVS-problem then its accuracy bound depends on the parameter $C_{A_{m}}$ and its complexity – on the parameter $T(A)$. Those are the bound on the radius that an algorithm $A$ can assure for the CVS-problem in the space $R_{s}^{m}$ and the complexity of the algorithm $A$, correspondingly. Concrete bounds on accuracy and complexity of an approximation algorithm can be obtained by replacing those abstract parameters by the corresponding characteristics of a concrete Algorithm $A$.

6.9.

We will miss the factor $\max_{j} \|t_{j}\|_{s}$ in accuracy bounds of approximate solutions of the VCP-problem, assuming that $\|t_{j}\|_{s} \leq 1, \forall j$. We also assume the norm $s = l_{\infty}$ for the functional $L(S)$. 
The norm $s_n$ that plays an important role in our research is specified by its unit ball $B_{s,n,m}$ which is defined as follows:

$$B_{s,n,m} = \{ x = (x_1, \ldots, x^n) \in \mathbb{R}^n | |x_i| \leq 1, |x_i - x_j| \leq 1, \forall i, j \}.$$  

Note that $B_{s,2}$ can be linearly transformed to the right hexagon.

The functions $\psi(m), \psi_1(m), \psi_2(m)$ in notation of results Nos. 2–4 denote just a function of the space dimension in the CVS-problem, a function of the number of machines in the FS-problem and a function of the number of parameters in the VCP-problem correspondingly. We mean by this that the functions are independent of other parameters of these problems.

The result No. 30 is a special case of a more general result (for arbitrary norm $s$), where the radius in the CVS-problem is bounded from above by a function $\sigma_x$ being a simple geometric characteristic of the convex body $B_{s,2}$. Yet we will not define the function $\sigma_x$ here.

The accuracy bounds Nos. 18 and 25 were obtained long before by Gross [22] and Bergström [12].

Result No. 53 was announced without a proof.

Using the result of [41], we may replace the coefficient 1.06 in No. 22 and the coefficient 0.06 in No. 33 by 1.00023 and 0.00023 (and by 1 and 0 for $l \leq 250$), correspondingly.

Results Nos. 31, 44, 48–50, 54, 57 and 58 marked by the asterisk could be slightly improved by means of result No. 59.
6.17.

To avoid searching (through the whole paper) for notations used in the table, let us list them below.

(a) Problem abbreviation:
- G — general (see Section 2.1),
- FS — flow shop (see Section 2.1),
- JS — job shop (see Section 3.1),
- OS — open shop (see Section 3.1),
- OS1 — G-problem without precedence constraints (Section 3.1),
- AL — assembly line (Section 3.1),
- AF — Akers–Friedman problem (Section 3.1),
- AF_p — AF-problem with p routes (Section 3.1),
- AF^2 — counter routes problem (Section 3.1),
- FS1 — FS-problem with alternative machines (Section 3.1),
- FS2 — w-cyclic route of jobs (Section 3.1),
- VCP — volume calendar planning problem (Section 2.1),
- CVS — compact vector summation within a ball (Section 2.2).

(b) Optimality criteria:
- F — the makespan in scheduling,
- L, L_2 — criteria in the VCP-problem, s is a norm in \( \mathbb{R}^m \),
- C — criterion of the CVS-problem,
- \( C_{1,m}^A \) — the value of the criterion C, an algorithm A can guarantee for the CVS-problem in the space \( \mathbb{R}^m \).

(c) Miscellaneous notations:
- \( n \) — the number of jobs, vectors,
- \( m \) — the number of machines, parameters; space dimension, \( m' = 2 \log_2 m \),
- \( l_i \) — the number of alternative machines for the \( i \)th operation in the FS1-problem; \( L = \sum l_i \), \( D = \max_{j=1,\ldots,m} \sum_{i=1}^{r_j} t_{ij} / l_i \),
- \( S \) — a solution (a schedule in scheduling problems, a distribution of items in the VCP-problem),
- \( S_{\text{opt}} \) — the optimal schedule,
- \( S_{\pi} \) — the permutation schedule specified by a permutation \( \pi \),
- \( M \) — Maximum machine load,
- \( K \) — maximum operation length, \( (t_{ij}) \) — matrix of operation lengths,
- \( r_j \) — the number of operations of job \( j \),
- \( R = \sum r_j \),
- \( w \) — the multiplicity of the cyclic route \( (1, 2, \ldots, m, 1) \) in the FS2-problem.

References


