A unit-time scheduling problem with makespan criterion may be interpreted as an optimal coloring $\phi : V \rightarrow \{1, 2, \ldots, t\}$ of the vertices $V = \{v_1, v_2, \ldots, v_n\}$ of a mixed graph $(V, A, E)$, where the number $t$ of colors is minimal, $\phi(v_i) < \phi(v_j)$ for each arc $(v_i, v_j) \in A$, and $\phi(v_p) \neq \phi(v_q)$ for each edge $[v_p, v_q] \in E$. We consider an optimal coloring $\phi$ which defines a schedule that minimizes the makespan for the unit-time job-shop problem. For such a mixed graph $(V, A, E)$, it follows that digraph $(V, A, \emptyset)$ is the union of disjoint paths and graph $(V, \emptyset, E)$ is the union of disjoint cliques. We present complexity results for the case of short paths and small cliques, and for the case of long paths or large cliques. For the case of two paths, we improve a geometrical algorithm. For the general case of a unit-time job-shop problem, we develop three branch-and-bound algorithms and test them on randomly generated mixed graphs of order $n \leq 200$ for the exact solution and of order $n \leq 900$ for the approximate solution.

Keywords: Scheduling algorithm; Mixed graph; Coloring

1 PROBLEM SETTING AND NOTATIONS

Let $G = (V, A, E)$ denote a mixed graph with non-empty set $V$ of vertices, $V \neq \emptyset$, placed at the first position in parentheses, arc set $A$ at the second position, and edge set $E$ at the third position. Hereafter, arc $(v_i, v_j) \in A$ denotes the ordered pair of vertices $v_i$ and $v_j$, and edge $[v_p, v_q] \in E$ denotes the unordered pair of vertices $v_p$ and $v_q$. If $A = \emptyset$, we have a graph $(V, \emptyset, E)$. If $E = \emptyset$, we have a digraph $(V, A, \emptyset)$. A mixed graph coloring $\phi$ is defined as follows (see [7, 20, 21]).

**Definition 1** An integer-valued function $\phi : V \rightarrow \{1, 2, \ldots, t\}$ is a coloring of the mixed graph $G = (V, A, E)$ if $\phi(v_i) < \phi(v_j)$ for each arc $(v_i, v_j) \in A$, and $\phi(v_p) \neq \phi(v_q)$ for each edge $[v_p, v_q] \in E$. Coloring $\phi$ is optimal if it uses a minimum number $t = \gamma(G)$ of colors, such a number $\gamma(G)$ being called the chromatic number of the mixed graph $G$.

The minimization of the maximum completion time (i.e., the minimization of the length of a schedule) of $n$ partially ordered operations $V = \{v_1, v_2, \ldots, v_n\}$ with unit processing...
times $p_i = 1$, $i = 1, 2, \ldots, n$, (or what means the same, with equal processing times) may be interpreted as an optimal coloring $\phi$ of a mixed graph $G = (V, A, E)$, in which $V$ is the set of operations, arc set $A$ defines precedence constraints, and edge set $E$ defines capacity constraints. Due to Definition 1, a coloring $\phi : V \rightarrow \{1, 2, \ldots, t\}$ of a mixed graph $G$ defines a feasible assignment of operations $V$ to the following set of unit-time intervals:

$$[0, 1], (1, 2], (2, 3], \ldots, (t - 1, t].$$

(1)

An optimal coloring $\phi$ of a mixed graph $G$ defines an assignment of operations $V$ to a minimal number of unit-time intervals:

$$[0, 1], (1, 2], (2, 3], \ldots, (\gamma(G) - 1, \gamma(G)].$$

The latter operation assignment is makespan optimal; it defines a makespan optimal schedule of the operations $V$, the length of which is equal to the chromatic number $\gamma(G)$.

Finding an optimal coloring of a given mixed graph $(V, A, E)$ is an NP-hard problem even if $A = \emptyset$ (see [10]). In contrast to an optimal coloring of a graph $(V, \emptyset, E)$, which is well-studied, an optimal coloring of a mixed graph $(V, A, E)$ with $A \neq \emptyset$ and $E \neq \emptyset$ has found limited attention in the OR literature. To the best of our knowledge, all papers which deal with a mixed graph coloring are included in the list of references which follows.

Let $(V, \emptyset, E_A)$ denote the graph obtained from digraph $(V, A, \emptyset)$ by replacing the set of arcs $A$ by the set of edges $E_A = \{[v_i, v_j] : (v_i, v_j) \in A\}$. In [7], an $O(n^2)$-algorithm has been developed for an optimal coloring $\phi$ of a mixed graph $G$ for which graph $(V, \emptyset, E \cup E_A)$ is a tree, and a branch-and-bound algorithm was developed and tested for an optimal coloring of randomly generated mixed graphs of order $n \leq 70$ with a density of edges equal to 0.1 or 0.2. In [20, 21], the chromatic polynomial $f(G, t)$ and the chromatic number have been studied for a mixed graph $\psi$ for which arc $(v_i, v_j) \in A$ implies that the color of vertex $v_i$ is less than or equal to the color of vertex $v_j$: $\psi(v_i) \leq \psi(v_j)$ (see Definition 1 for a comparison of coloring $\phi$ and coloring $\psi$). For each $t$, the value of the chromatic polynomial $f(G, t)$ is equal to the number of different colorings $\psi$ of the mixed graph $G$ each of which uses $t$ colors.

Obviously, coloring $\phi$ may be considered as a special case of coloring $\psi$ provided that each inclusion $(v_i, v_j) \in A$ implies the inclusion $[v_i, v_j] \in E$ in the mixed graph $G = (V, A, E)$ which has to be colored. In [12], lower bounds for the chromatic number of a mixed graph in the case of coloring $\psi$ have been considered. In [11], an enumerative algorithm and a FORTRAN program for constructing an optimal coloring $\psi$ have been developed. In [2], a branch-and-bound algorithm for constructing an optimal coloring $\psi$ has been developed and tested on randomly generated mixed graphs of order $n \leq 150$ with a density of edges equal to 0.1 or 0.2.

In this paper, we consider an optimal coloring $\phi$ of a mixed graph $G$ which corresponds to a unit-time, minimum-length, job-shop scheduling problem denoted by $J|p_i = 1|C_{\text{max}}$ in the three-field scheme [6]. An optimal coloring $\phi$ of a mixed graph $G$ defines a makespan optimal schedule. Using graph terminology, we have to assume that the mixed graph $G = (V, A, E)$ under consideration has the following two mandatory properties.

**Property 1:** The partition $(V, \emptyset, E) = (V_1, \emptyset, E_1) \cup (V_2, \emptyset, E_2) \cup \ldots \cup (V_m, \emptyset, E_m)$ holds, where each graph $(V_k, \emptyset, E_k)$ is a clique for each $k = 1, 2, \ldots, m$, and $V_k \cap V_l = \emptyset$ for $k \neq l$.

**Property 2:** Digraph $(V, A, \emptyset)$ has no transitive arcs and the partition $(V, A, \emptyset) = (V^{(1)}, A^{(1)}, \emptyset) \cup (V^{(2)}, A^{(2)}, \emptyset) \cup \ldots \cup (V^{(j)}, A^{(j)}, \emptyset)$ holds, where each digraph $(V^{(k)}, A^{(k)}, \emptyset)$ is a path.
(v_{k_1}, v_{k_2}, \ldots, v_{k_r})$ for each $k = 1, 2, \ldots, j$, and $V^{(k)} \cap V^{(l)} = \emptyset$ for $k \neq l$.

Property 1 (Property 2, respectively) means that subgraph $(V, \emptyset, E)$ of a mixed graph $G$ is a union of disjoint cliques (subgraph $(V, A, \emptyset)$ is a union of disjoint paths). In a job-shop scheduling problem, the numbers $m$ and $j$ denote the cardinality of the machine set $M = \{M_1, M_2, \ldots, M_m\}$ and the cardinality of the job set $J = \{J_1, J_2, \ldots, J_j\}$, respectively. From Property 2 it follows that, if inclusion $v_i \in V^{(k)}$ holds, then operation $v_i$ is a part of job $J_k \in J$, and vice versa (see Definition 1). Each job $J_k \in J$ consists of a set $V^{(k)}$ of linearly ordered operations, i.e. job $J_k$ is represented as a path $(v_{k_1}, v_{k_2}, \ldots, v_{k_r})$, and the operations $V^{(k)}$ have to be processed in the order given by this path. From Property 1 it follows that, if $v_i \in V_k$, then operation $v_i$ has to be processed by machine $M_k \in M$, and vice versa. Due to Definition 1, Property 1 means that each machine $M_k \in M$ can process at most one operation within one unit-time interval from the set $(1)$.

Thus, Property 1 and Property 2 present the usual assumptions of scheduling theory in terms of a coloring of a mixed graph. Summarizing, we note that there exists a one-to-one correspondence between a coloring $\phi$ of a mixed graph $G = (V, A, E)$ with Properties 1 and 2, and the scheduling problem $J|p_i = 1|C_{\text{max}}$, namely:

\[
\begin{align*}
\{ \text{ vertex } v_i \} & \leftrightarrow \{ \text{ operation } v_i \}, \\
\{ \text{ set of vertices of the path } (V^{(k)}, A^{(k)}, \emptyset) \} & \leftrightarrow \{ \text{ set of operations of the job } J_k \}, \\
\{ \text{ set of vertices of the clique } (V_k, \emptyset, E_k) \} & \leftrightarrow \{ \text{ set of operations processed by machine } M_k \}, \\
\{ \text{ coloring } \phi \} & \leftrightarrow \{ \text{ schedule } \phi \}, \\
\{ \text{ optimal coloring } \phi \} & \leftrightarrow \{ \text{ makespan optimal schedule } \phi \}.
\end{align*}
\]

Although we will use both graph terminology and scheduling terminology for the problem under consideration, it is possible to describe each result and algorithm which follow using either only graph terminology or only scheduling terminology.

The rest of the paper is organized as follows. In Section 2 we survey and develop complexity results for an optimal coloring of a mixed graph $G$ with long (short) paths or large (small) cliques. In Section 3 we improve a well-known geometrical algorithm for the case of two jobs (two paths). Section 4 deals with branch-and-bound algorithms for an optimal coloring $\phi$ of a mixed graph $G$ with mandatory Properties 1 and 2. Computational experiences for full and limited versions of the three branch-and-bound algorithms are reported in Section 5. Concluding remarks are given in Section 6.

## 2 COMPLEXITY RESULTS

Most results observed in this section (see also [17]) for the complexity of an optimal coloring $\phi$ follow from those for problem $J|p_i = 1|C_{\text{max}}$ (the corresponding references are given in Table I and Table II). For the new results, relevant proofs are given in this section and Section 3.

We assume that all mixed graphs $G$ considered in the rest of this paper satisfy Properties 1 and 2 without fail. Along with the mandatory Properties 1 and 2, we consider Property 3, which means that any two sequential operations of the same job $J_k \in J$ in a problem $J|p_i =
Subgraph \((V_k, A_k, \emptyset)\) of digraph \((V, A, \emptyset)\) is empty for each \(k = 1, 2, \ldots, m\), i.e. equality \(A_k = \emptyset\) holds.

In what follows, the notation \(J[p_i = 1]C_{\text{max}}\) is used for a unit-time job-shop problem if machine repetition (in processing two sequential operations of a job) is not allowed, i.e. if the mixed graph \(G\) has Property 3. If machine repetition is allowed (i.e. if Property 3 may be violated), the notation \(J[p_i = 1, \text{rep}]C_{\text{max}}\) is used. It should be noted that problem \(J[p_i = 1, \text{rep}]C_{\text{max}}\) is more general than problem \(J[p_i = 1, \text{rep}]C_{\text{max}}\), since for problem \(J[p_i = 1, \text{rep}]C_{\text{max}}\) machine repetitions are allowed but they are not obligatory. Obviously, problem \(J[p_i = 1, \text{rep}]C_{\text{max}}\) is equivalent to the job-shop problem \(J[p_i, \text{pmtn}]C_{\text{max}}\) with integer processing times and allowed preemptions of operations (hereafter the notation \([p_i]\) is used for the case of integer processing times).

Note that Property 3 may influence the complexity of a scheduling problem. An astonishing example of such an influence was given in [4], where it was proven that the job-shop problem \(J2[j = 3, p_i = 1, \text{rep}]C_{\text{max}}\) is NP-hard, while in [13, 24] polynomial algorithms for the corresponding job-shop problem without machine repetition have been derived.

Column 3 in Table I and column 5 in Table II are used to indicate whether the mixed graph \(G\) has Property 3 (in this case the column contains ‘yes’) or not (in this case the column contains ‘no’).

### 2.1 Short paths and small cliques

In Table I, we present complexity results for an optimal coloring of a mixed graph \(G\) when the chromatic number \(\gamma(G)\) is small. More precisely, the recognition of the inequality \(\gamma(G) \leq l\) is considered with a fixed positive integer number \(l\) (see column 2 in Table I).

Note that testing inequality \(\gamma(G) \leq 2\) is a trivial problem when either Property 3 holds or not. Indeed, equality \(\gamma(G) = 1\) holds if and only if \(E = \emptyset\) and \(A = \emptyset\). A simple criterion for the equality \(\gamma(G) = 2\) is given in the following claim.

**Lemma 1**  
Equality \(\gamma(G) = 2\) holds if and only if
1) \(|A| + |E| \geq 1\),
2) \(\max_{A \in J} |V^{(k)}| \leq 2\),
3) \(\max_{M \in A} |V_k| \leq 2\),
4) there are no two paths \((v_{k_1}, v_{k_2})\) and \((v_{s_1}, v_{s_2})\) such that \([v_{k_1}, v_{s_1}] \in E\) or \([v_{k_2}, v_{s_2}] \in E\).

**Proof.**  
**Necessity.** If condition 1) is violated, then \(\gamma(G) = 1\). If condition 1), 2) or 3) is violated, then \(\gamma(G) \geq 3\).

**Sufficiency.** From Properties 1 and 2 of the mixed graph \(G = (V, A, E)\) and conditions 2), 3) and 4), it follows that graph \((V, \emptyset, E \cup E_A)\) is a bipartite graph. (The definition of a graph \((V, \emptyset, E \cup E_A)\) is given in Section 1.) Moreover, we can present a partition \(V = V' \cup V''\), \(V' \cap V'' = \emptyset\), of this bipartite graph (here inclusion \([v_p, v_q] \in E \cup E_A\) implies \(v_p \in V'\) and \(v_q \in V''\) in such a way that each arc \((v_p, v_q) \in A\) implies both inclusions \(v_p \in V'\) and
In [25], it was proven that the problem of deciding if there is an optimal schedule for problem $J||p_i|C_{\text{max}}$ with a length of at most 3 can be reduced to the 2-satisfiability problem in $O(n)$ time. Since problem $J||p_i|C_{\text{max}}$ is a special case of problem $J||p_i||C_{\text{max}}$ and taking into account that the 2-satisfiability problem can be solved in $O(n)$ time (see [5]), we conclude that the recognition of the inequality $\gamma(G) \leq 3$ can be done in $O(n)$ time if Property 3 holds for the mixed graph $G$ (see line 1 in Table I).

Next, we show how to use the reduction from [25] for the recognition of the inequality $\gamma(G) \leq 3$ when Property 3 does not hold (see line 2 in Table I). Using a reduction similar to the one from [25], we show that deciding if there is a schedule with length 3 can be reduced to the 2-satisfiability problem. Recall that the latter problem is to decide whether a logical formula, in which each clause contains at most two logical variables, has a satisfying assignment (true or false) to all logical variables.

Obviously, for a coloring $\phi^* : V \rightarrow \{1, 2, 3\}$ (if any) only paths of a length of at most 3 and cliques of a cardinality of at most 3 are allowed. In the following algorithm, no more than three variables $x_{i1}, x_{i2}$ and $x_{i3}$ are prescribed to each vertex $v_i \in V$, and it is assumed that setting $x_{it}$ to be true means that $\phi^*(v_i) = t$, while setting $x_{it}$ to be false means that $\phi^*(v_i) \neq t$.

**Algorithm 1**

1. If $A = \emptyset$ and $E = \emptyset$, then $\gamma(G) = 1$. **Stop.** Otherwise go to Step 2.

2. If Lemma 1 holds, then $\gamma(G) = 2$. **Stop.** Otherwise go to Step 3.

3. If inequality $\max_{M_t \in M} |V_t| \leq 3$ holds, go to Step 4. Otherwise $\gamma(G) > 3$. **Stop.**

4. If inequality $\max_{J_t \in J} |V^{(k)}| \leq 3$ holds, go to Step 5. Otherwise $\gamma(G) > 3$. **Stop.**

5. Set $F = \emptyset$. For each vertex $v_{k_l}, l \in \{1, 2, 3\}$, in a path $(v_{k_1}, v_{k_2}, v_{k_3})$ add the clauses $(x_{k_1}), (x_{k_2})$ and $(x_{k_3})$ to the logical formula $F$.

6. For each path $(v_{k_1}, v_{k_2})$ add the clauses $(x_{k_1} \lor x_{k_2}), (\overline{x}_{k_1} \lor \overline{x}_{k_2}), (x_{k_2} \lor x_{k_3}), (\overline{x}_{k_2} \lor \overline{x}_{k_3})$ to the logical formula $F$.

7. For each vertex $v_{k_p} \in V_s$, $V_s \neq \emptyset$, in a path of length 2 or 3 and for each vertex $v_{k_l} \in V_s$, $l \neq k$, add to the logical formula $F$ the clauses $(\overline{x}_{k_p} \lor \overline{x}_{k_l})$ for each number $t \in \{1, 2, 3\}$, which is allowed as a color for both vertices $v_{k_p}$ and $v_{k_l}$.

8. Test the logical formula $F$ obtained after Steps 5, 6 and 7. If the logical formula $F$ is satisfiable, then $\gamma(G) = 3$, otherwise $\gamma(G) > 3$. **Stop.**

The correctness of Algorithm 1 follows from Theorem 1 which may be proven similarly to Theorem 4 proven in [25].
Table I: Complexity of a mixed graph coloring with short paths and small cliques

<table>
<thead>
<tr>
<th>$\gamma(G) \leq l$</th>
<th>Property 3</th>
<th>Complexity</th>
<th>References</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>$l = 3$</td>
<td>$O(n)$</td>
<td>[5, 25]</td>
</tr>
<tr>
<td>2</td>
<td>$l = 3$</td>
<td>$O(n)$</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>$l = 4$</td>
<td>$NP$-hard</td>
<td>[25]</td>
</tr>
</tbody>
</table>

**Theorem 1** The logical formula $F$ constructed by Algorithm 1 is satisfiable if and only if $\gamma(G) = 3$.

To demonstrate Algorithm 1, we use the following example of a mixed graph $G = (V, A, E)$, where $V = \{v_1, v_2, v_3, v_4, v_5\}$, $A = \{(v_1, v_2), (v_2, v_3), (v_4, v_5)\}$ and $E = \{[v_1, v_4], [v_3, v_5]\}$. Steps 5, 6 and 7 of Algorithm 1 generate the following logical formula $F = (x_{11}) \land (x_{22}) \land (x_{33}) \land (x_{41} \lor x_{42}) \land (\bar{x}_{41} \lor \bar{x}_{42}) \land (x_{52} \lor x_{53}) \land (\bar{x}_{52} \lor \bar{x}_{53}) \land (x_{11} \lor x_{41}) \land (\bar{x}_{33} \lor \bar{x}_{53})$. Since variables $x_{11}$ and $x_{33}$ have to be true (if the logical formula $F$ is satisfiable), variables $x_{41}$ and $x_{53}$ have to be false (see last two clauses in $F$). Thus, from clauses $(\bar{x}_{41} \lor \bar{x}_{42})$ and $(\bar{x}_{52} \lor \bar{x}_{53})$ it follows that the variables $x_{42}$ and $x_{52}$ have to be false, which contradicts to clause $(\bar{x}_{42} \lor \bar{x}_{52})$ in the logical formula $F$. Thus, we can conclude that the logical formula $F$ is not satisfiable and so $\gamma(G) > 3$ due to Theorem 1.

It is easy to see that Steps 1 - 7 of Algorithm 1 take $O(n)$ time. Since the logical formula $F$ constructed by Algorithm 1 includes at most $3n$ variables, and each clause in $F$ has at most two variables, Step 8 of Algorithm 1 may be realized in $O(n)$ time using the algorithm from [5]. Therefore, to test inequality $\gamma(G) \leq 3$ when Property 3 does not hold takes also $O(n)$ time (see line 2 in Table I).

In [25], it was proven that deciding if there is an optimal schedule for problem $J|p_i|C_{\max}$ with a length of at most 4 is $NP$-hard. More precisely, a polynomial reduction was constructed from the restricted version of the 3-satisfiability problem (which is still $NP$-hard) to problem $J|p_i = 1|C_{\max}$ which is a special case of problem $J|p_i|C_{\max}$ (see line 3 in Table I).

Obviously, small values of the chromatic number $\gamma(G)$ may be possible only for a mixed graph $G$ with short paths $(V^{(k)}, A^{(k)}, \emptyset)$, $k = 1, 2, \ldots, j$, and small cliques $(V_t, \emptyset, E_t)$, $t = 1, 2, \ldots, m$, (see caption of Table 1). Indeed, inequality $\gamma(G) < l$ implies both inequalities $|V^{(k)}| < l$ and $|V_t| < l$.

As follows from Table I, the boundary between polynomially solvable and $NP$-hard tests of inequality $\gamma(G) \leq l$ is between $l = 3$ and $l = 4$. Remind that the recognition of the inequality $\gamma(V, \emptyset, E) \leq 3$ is an $NP$-hard problem (see [10]), while the recognition of the inequality $\gamma(V, \emptyset, E) \leq 2$ may be done in polynomial time (since $\gamma(V, \emptyset, E) \leq 2$ if and only if graph $(V, \emptyset, E)$ has no cycle with odd length).

### 2.2 Long paths and large cliques

In [23], it was proven that problem $J2|p_i = 1, r\in P|C_{\max}$ is $NP$-hard (see line 1 in Table II). In [14], it was proven that problem $J3|p_i = 1|C_{\max}$ is $NP$-hard (see line 2 in Table II). In [9, 24], an $O(n)$-algorithm has been developed for problem $J|p_i = 1|C_{\max}$ (see line 3 in Table II).
II). In the rest of this section and in Section 4, we continue the study of the complexity of an optimal coloring of a mixed graph \( G \) with Properties 1 and 2.

If \( j = n \) (see Property 2), then \((V, A, \emptyset) = (\{v_1\}, \emptyset, \emptyset) \cup (\{v_2\}, \emptyset, \emptyset) \cup \ldots \cup (\{v_n\}, \emptyset, \emptyset)\), i.e. the set of arcs \( A \) is empty and \( G = (V, \emptyset, E) \). In such a case, due to Property 1 the chromatic number \( \gamma(G) \) is equal to the maximum size of a clique in the graph \( G = (V, \emptyset, E) \):

\[
\gamma(G) = \max_{k=1}^{m} |V_k|.
\]

If the input data includes a list of adjacent vertices for each vertex \( v_i \in V \), then the calculation of \( \gamma(G) \) takes \( O(n) \) time. If the input data includes the sets \( V_1, V_2, \ldots, V_m \), then the calculation of \( \gamma(G) \) takes \( O(m) \) time.

If \( m = n \) (see Property 1), then \( M = \{M_1, M_2, \ldots, M_n\} \) and each operation \( v_i \in V \) has to be processed by a special machine \( M_i \in M \). Therefore, we have \( G = (V, A, \emptyset) \) and the chromatic number \( \gamma(G) \) is equal to the maximum length \( r = \max_{k=1}^{j} |V^{(k)}| \) of a path in the digraph \( G = (V, A, \emptyset) \):

\[
\gamma(G) = r = \max_{k=1}^{j} |V^{(k)}|.
\]

If the input data includes the sets \( V^{(1)}, V^{(2)}, \ldots, V^{(j)} \), then the calculation of \( \gamma(G) \) takes \( O(j) \) time, otherwise it takes \( O(n) \) time. Summarizing, we conclude that if \( m = n \) or \( j = n \), the chromatic number \( \gamma(G) \) can be found in the worst case in \( O(n) \) time (see lines 4 and 5 in Table II).

Next, we describe a polynomial time algorithm for an optimal coloring of a mixed graph \( G \) if the number of paths \( j \) is a constant (see lines 6 in Table II). This algorithm (we call it Algorithm 2) is based on constructing a circuit-free network \((W, B, \emptyset)\) and finding a shortest path in it from the source to the sink.

The set of vertices \( W \) is the set of states of the coloring process of the given mixed graph \( G = (V, A, E) \). The source state \( w_0 \) in the network \((W, B, \emptyset)\) is the state when no vertex from set \( V \) is colored. The sink state \( w_1 \) is the state when all vertices \( V \) are colored. An intermediate state \( w_p \in W \) defines the state of coloring the path \((V^{(k)}, A^{(k)}, \emptyset)\) for each \( k = 1, 2, \ldots, j \). More precisely, state \( w_p \) defines a subpath \((v_{k_1}, v_{k_2}, \ldots, v_{k_s}) \), \( s \leq r_k \), of the path \((v_{k_1}, v_{k_2}, \ldots, v_{k_s}) \), whose vertices are colored, while the remaining vertices \((v_{k_{s+1}}, v_{k_{s+2}}, \ldots, v_{k_{r_k}})\) of the path \((V^{(k)}, A^{(k)}, \emptyset)\) are not colored in this state.

Since each path \((V^{(k)}, A^{(k)}, \emptyset), k = 1, 2, \ldots, j\), has \( r_k \) possible states, the whole number \( |W| \) of states is equal to \( \prod_{k=1}^{j} (r_k + 1) \). Thus, we obtain \( |W| \leq (r+1)^j \), and the asymptotic bound of the number of states \( |W| \) is equal to \( O(r^j) \). If the number \( j \) is constant, the bound \( O(r^j) \) is a polynomial in \( r \).

Arc \((w_p, w_q)\) belongs to set \( B \) if and only if state \( w_q \) may be obtained from state \( w_p \) using one additional color (in this case the length of arc \((w_p, w_q)\) is assumed to be equal to 1) or without using an additional color (in this case the length of arc \((w_p, w_q)\) is assumed to be equal to 0). Thus, if \((w_p, w_q) \in B \), the number \( s \) of vertices colored in the path \((V^{(k)}, A^{(k)}, \emptyset)\) may be increased by no more than 1 due to passing from state \( w_p \) to state \( w_q \).

To construct all arcs \((w_p, w_q)\) with start state \( w_p \in W \), we have to look over \( \binom{j}{1} + \binom{j}{2} + \ldots + \binom{j}{j} = 2^j \) possible states \( w_q \in W \). The value \( 2^j \) is constant if the number \( j \) is constant. Since there are \( m \) machines in the set \( M \) and Property 1 holds, the number of arcs in the
set $B$ with start state $w_0 \in W$ is no more than $\binom{j}{m}$. Therefore, the whole number $|B|$ of arcs is no more than $(r + 1)^j \binom{j}{m}$. If both $j$ and $m$ are fixed (are constants), then value $\binom{j}{m}$ is a constant, too. If only number $j$ is fixed but $m$ not, the value $\binom{j}{m}$ cannot be more than $j$. Thus, in both cases an asymptotic bound of the number of arcs $B$ is $O(r^j)$ and to find a shortest path from the source vertex $w_0$ to the sink vertex $w_i$ in the circuit-free network $(W, B, \emptyset)$ takes $O(r^j)$ time. It is easy to convince that a shortest path from $w_0$ to $w_i$ defines an optimal coloring of a mixed graph $G$. Thus, if the number $j$ of paths is fixed, Algorithm 2 is a polynomial one in $r$ (see line 6 in Table II, where equality $j = k$ in the third column means that the number of jobs is equal to a constant $k$).

Unfortunately, Algorithm 2 has mainly theoretical significance: it is practically efficient only for small numbers $j$ and $r$. For the case $j = 2$, there are practically efficient algorithms based on a geometrical approach. E.g. for a mixed graph $G$ with $j = 2$, one can use the $O(r^2 \log r)$ algorithm developed in [15, 16] for the job-shop problem $J|j = 2|\Phi$ with two jobs, real processing times and any regular criterion $\Phi$. If all processing times $p_i$ are integers, then problem $J|j = 2, [p_i]|C_{\text{max}}$ is equivalent to problem $J|j = 2, p_i = 1, r e p|C_{\text{max}}$, in which $p_i$ unit-time operations correspond to one operation with an integer processing time equal to $p_i$; in problem $J|j = 2|C_{\text{max}}$. In Section 3, we show how to improve the geometrical $O(r^2 \log r)$ algorithm from [15, 16] for the case of unit-time operations.

3 GEOMETRICAL ALGORITHMS FOR COLORING TWO PATHS

A geometrical approach for solving problem $J|j = 2|C_{\text{max}}$ was proposed in [1] and developed in [3, 8, 15, 16, 22]. In Subsections 3.1 and 3.2, we show how to improve the geometrical algorithm from [15, 16] for problems $J|j = 2, p_i = 1, r e p|C_{\text{max}}$ and $J|j = 2, p_i = 1|C_{\text{max}}$, respectively.

3.1 Problem $J|j = 2, p_i = 1, r e p|C_{\text{max}}$

Let $j$ be equal to 2, digraph $(V^{(1)}, A^{(1)}, \emptyset)$ be the path $(v_{1_1}, v_{1_2}, \ldots, v_{1_{r_1}})$ and digraph $(V^{(2)}, A^{(2)}, \emptyset)$ be the path $(v_{2_1}, v_{2_2}, \ldots, v_{2_{r_2}})$ (see Property 2). The rectangle $S$ with the corners $(0, 0), (r_1, 0), (0, r_2)$ and $(r_1, r_2)$ may be shown in the coordinate plane.

In Fig. 1 the rectangle $S$ is presented for the example of problem $J4|j = 2, p_i = 1, r e p|C_{\text{max}}$ with job $J_1$, which has to be processed by the machines $M = \{M_1, M_2, M_3, M_4, M_5\}$ in the order $M_1, M_1, M_2, M_3, M_3, M_4, M_3, M_1, M_1, M_1, M_1, M_2, M_2, M_3, M_4, M_4, M_2, M_2, M_2, M_2, M_2, M_1$ and with job $J_2$, which has to be processed in the order $M_1, M_2, M_2, M_4, M_3, M_3, M_1, M_1, M_1, M_4, M_4, M_4, M_2, M_3, M_3, M_2, M_4$. For this example $r_1 = 23$ and $r_2 = 17$.

Fig. 1

The coordinate $x$ (coordinate $y$) of point $(x, y) \in S$ defines the flow time of processing job $J_1$ (job $J_2$, respectively). The distance $d((x, y), (x', y'))$ between two points $(x, y)$ and $(x', y')$
in the rectangle $S$ is defined as follows:

$$d((x, y), (x', y')) = \max\{|x - x'|, |y - y'|\}. \quad (2)$$

The point $(0, 0)$ defines the start of processing one or both jobs from the set $J = \{J_1, J_2\}$, while point $(r_1, r_2)$ defines the end of processing both jobs $J_1$ and $J_2$. A schedule $\phi$ can be represented as a trajectory (broken line) $\tau$ in the rectangle $S$ which connects point $(0, 0)$ with point $(r_1, r_2)$. Such a trajectory $\tau$ may include three types of straight segments: horizontal (only job $J_1$ is processed), vertical (only job $J_2$ is processed), and diagonal (both jobs $J_1$ and $J_2$ are processed simultaneously by different machines from the set $M$). From Property 1 it follows that, if inclusion $[v_1, v_2] \in E$ holds, then the diagonal segment cannot pass through the points of the square $S_{u,z}$ with the corners $(u - 1, z - 1), (u, z - 1), (u - 1, z)$ and $(u, z)$. However, a diagonal segment can pass through the left-upper corner $(u - 1, z)$ and through the right-lower corner $(u, z - 1)$ of the square $S_{u,z}$.

In [1, 8, 22], it was shown that the shortest trajectory $\tau$ from the point $(0, 0)$ to the point $(r_1, r_2)$ with the distance based on formula (2) defines a makespan optimal schedule for problem $J||\text{rep}C_{\text{max}}$. Such a trajectory $\tau$ is called optimal. Finding an optimal trajectory $\tau$ is reduced to the determination of a shortest path in a network $(W', B', \emptyset)$. In [15], it was shown that constructing the network $(W', B', \emptyset)$ and finding a shortest path from vertex $(0, 0)$ to vertex $(r_1, r_2)$ in the network $(W', B', \emptyset)$ take $O(r^3 \log r)$ time. For the case $p_i = 1$ under consideration, network $(W', B', \emptyset)$ may be constructed as follows (we refer to [15, 16] for a more formal description of this construction).

The set of vertices $W'$ consists of the points $(0, 0), (r_1, r_2)$, the left-upper corners $(u - 1, z)$ and the right-lower corners $(u, z - 1)$ of the squares $S_{u,z}$, $u = 1, 2, \ldots, r_1$, $z = 1, 2, \ldots, r_2$. Each vertex $(x, y) \in W' \setminus \{(r_1, r_2)\}$ is a start vertex of one or two arcs in the set $B'$, or vertex $(x, y)$ may be an isolated one in the network $(W', B', \emptyset)$. To construct the set of arcs $B'$, we first set $B' = \emptyset$ and go from point $(0, 0)$ diagonally in the right-upper direction until we hit either the boundary of the rectangle $S$ or the left-lower corner $(u - 1, z)$ of a square $S_{u,z}$. In the former case, we set $B' := B' \cup \{((0, 0), (r_1, r_2))\}$ and conclude that network $(W', B', \emptyset)$ is constructed. In the latter case, we set $B' := B' \cup \{((0, 0), (u - 1, z)), ((0, 0), (u, z - 1))\}$. Then we continue similarly going from each vertex $(x, y) \in W' \setminus \{(0, 0)\}$ to which a path from vertex $(0, 0)$ in the network $(W', B', \emptyset)$ exists.

Next, we show how to restrict the set of vertices $W'$, which have to be constructed, and present a more efficient algorithm for constructing the network. Without loss of generality, we assume that

$$r_1 \geq r_2$$

in what follows. Starting from the point $(0, 0)$ diagonally in the right-upper direction, we construct a left bounding trajectory $\tau_0^1$ (a right bounding trajectory $\tau_0^2$) which includes the left-upper corner $(u - 1, z)$ (the right-lower corner $(u, z - 1)$, respectively) for each square $S_{u,z}$ hit. Let $D_0$ denote the closed region in rectangle $S$ between the trajectories $\tau_0^1$ and $\tau_0^2$.

In Fig. 1, the trajectory $\tau_0^1 = (((0, 0), (1, 0), (2, 0), (6, 3), (7, 3), (8, 4), (9, 4), (11, 6), (12, 6), (13, 6), (16, 9), (17, 9), (20, 12), (21, 12), (22, 12), (23, 13), (23, 17))$ and trajectory $\tau_0^2 = (((0, 0), (0, 1), (3, 4), (3, 5), (3, 6), (6, 9), (6, 10), (6, 11), (6, 12), (7, 13), (7, 14), (7, 15), (9, 17), (23, 17))$ are presented by dotted lines.
Similarly, starting from the point \((r_1, r_2)\) diagonally in the left-lower direction, we construct a left bounding trajectory \(\tau^2_r\) (a right bounding trajectory \(\tau^1_r\)) from the point \((r_1, r_2)\) to the point \((0, 0)\), which includes corner \((u - 1, z)\) (corner \((u, z - 1)\), respectively) for each square \(S_{uz}\) hit. Let \(D_r\) denote the closed region in rectangle \(S\) between the trajectories \(\tau^1_r\) and \(\tau^2_r\).

In Fig. 1, the trajectory \(\tau^1_r = ((23, 17), (22, 16), (22, 15), (20, 13), (20, 12), (8, 0), (0, 0))\) and trajectory \(\tau^2_r = ((23, 17), (22, 16), (21, 16), (20, 16), (19, 16), (18, 16), (17, 16), (15, 14), (14, 14), (5, 5), (3, 5), (0, 2), (0, 0))\) are presented by dotted lines.

**Algorithm 3**

1. Starting from point \((0, 0)\), construct a left bounding trajectory \(\tau^2_0 = ((0, 0), \ldots, (x_0, r_2), (r_1, r_2))\) and a right bounding trajectory \(\tau^1_0 = ((0, 0), \ldots, (x_1, y_1), (r_1, r_2))\).

2. If \(y_1 = r_2\), then trajectory \(\tau^1_0\) is optimal (it defines a makespan optimal schedule). \textbf{Stop.} Otherwise go to Step 3.

3. Starting from point \((r_1, r_2)\), construct a trajectory \(\tau^1_r = ((r_1, r_2), \ldots, (x^0, 0), (0, 0))\) and a trajectory \(\tau^2_r = ((r_1, r_2), \ldots, (x^1, y^1), (0, 0))\).

4. If \(y^1 = 0\), then the trajectory \(((0, 0), (x^1, y^1), \ldots, (r_1, r_2))\) is optimal (it defines a makespan optimal schedule). \textbf{Stop.} Otherwise go to Step 5.

5. Construct the network \((W^*, B^*, \emptyset)\), \(W^* \subseteq W', B^* \subseteq B'\), such that vertex \((x, y)\) belongs to set \(W^*\) if and only if point \((x, y)\) belongs to region \(D_0 \cap D_r\).

6. Find an optimal (shortest) path from vertex \((0, 0)\) to vertex \((r_1, r_2)\) in the network \((W^*, B^*, \emptyset)\). (This path defines a makespan optimal schedule.) \textbf{Stop.}

The correctness of Algorithm 3 is based on Lemma 2 and Theorem 2 which follows. Note that inequality \(y_1 \leq r_2\) holds, and first, we consider the case when \(y_1 = r_2\).

**Lemma 2** If in the right bounding trajectory \(\tau^0_0 = ((0, 0), \ldots, (x_1, y_1), (r_1, r_2))\) the equality \(y_1 = r_2\) holds, then this trajectory is optimal, i.e. it defines a makespan optimal schedule, the length \(C^*_\text{max}\) of this schedule being equal to \(r_1\).

**Proof.** From equality \(y_1 = r_2\) and the definition of a right bounding trajectory \(\tau^1_0\), it follows that the length of trajectory \(((0, 0), \ldots, (x_1, r_2), (r_1, r_2))\) is equal to \(r_1\) and so Lemma 2 follows from the obvious lower bound for \(C^*_\text{max}\):

\[
C^*_\text{max} \geq \max\{r_1, r_2\} = r_1.
\]

The latter equality follows from the above assumption \(r_1 \geq r_2\). \hfill \blacksquare

**Theorem 2** If in the right bounding trajectory \(\tau^1_0 = ((0, 0), \ldots, (x_1, y_1), (r_1, r_2))\) the equality \(x_1 = r_1\) holds, then there exists an optimal trajectory which completely belongs to region \(D_0 \cap D_r\).
From Lemma 1 and Theorem 2 the correctness of Algorithm 3 follows. It is easy to see that
\[ \text{Proof.} \]

Note that there exists an optimal trajectory \( \tau_0 \) which completely belongs to region \( D_0 : \tau_0 \in D_0 \). This follows from the well-known fact that there exists an optimal semiactive schedule (see e.g. [23]). Remind that a schedule is called semiactive if no operation can start earlier without delaying the processing of some other operation or without violating the operation order defined by this schedule on some machine.

If the inclusion \( \tau_0 \in D_r \) holds as well, then \( \tau_0 \in D_0 \cap D_r \) and so the claim of Theorem 2 is valid. If \( \tau_0 \not\in D_r \), then let \( (x', y') \) be the first common point of trajectory \( \tau_0 \) and trajectory \( \tau_r \). Point \( (x', y') \) divides trajectory \( \tau_0 \) into two parts. The first part \( \tau' = ((0, 0), \ldots, (x', y')) \) completely belongs to region \( D_0 \), while for the second part \( \tau'' = ((x', y'), \ldots, (r_1, r_2)) \), only the end points \( (x', y') \) and \( (r_1, r_2) \) belong to region \( D_0 \). Let us consider the subproblem of the original problem in the rectangle \( S' \) defined by the corners \( (x', y'), (r_1, y'), (x', r_2) \) and \( (r_1, r_2) \). For this subproblem, Lemma 1 holds if point \( (r_1, r_2) \) is considered as the origin point of the coordinate plane instead of point \( (0, 0) \). Therefore, part \( \tau'' = ((r_1, r_2), \ldots, (x', y')) \) of the trajectory \( \tau_r \) is optimal for this subproblem due to Lemma 1. Thus, if we substitute part \( \tau'' \) by part \( \tau'' \) in trajectory \( \tau_0 \), we obtain an optimal trajectory which completely belongs to region \( D_0 \cap D_r \).

From Lemma 2 and Theorem 2 the correctness of Algorithm 3 follows. It is easy to see that Steps 1 - 4 of Algorithm 3 take \( O(r) \) time. The number \( |W^*| \) of vertices is no more than \( r^2 \), and so Step 6 in Algorithm 3 takes \( O(r^2) \) time. Next, we describe Step 5 of Algorithm 3 more in detail.

Let \( W^D \) denote the set of all vertices of the trajectories \( \tau_0, \tau_1, \tau_r \) and \( \tau_r \) which belong to the boundary of region \( D_0 \cap D_r \). Let \( \tau = ((0, 0), \ldots, (x', y'), (r_1, r_2)) \) be the left bounding trajectory of the region \( D_0 \cap D_r \), and \( \tau = ((0, 0), \ldots, (x', y'), (r_1, r_2)) \) be the right bounding trajectory of the region \( D_0 \cap D_r \).

**Algorithm** 3° (Step 5 of Algorithm 3)

1°. Set \( W^* = W^D \) and \( B^* = \emptyset \).

2°. Take a vertex \( (x, y) \in W^* \setminus \{(r_1, r_2)\} \) with zero outdegree, set \( k = 1 \) and go to Step 3°. If there is no vertex in the set \( (x, y) \in W^* \setminus \{(r_1, r_2)\} \) with zero outdegree, then network \( (W^*, B^*, \emptyset) \) is constructed. Go to Step 6 of Algorithm 3.

3°. Set \( x' = x + k \) and \( y' = y + k \).

4°. If \( M_{x'} = M_{y'} \), then go to Step 5°. Otherwise set \( k := k + 1 \) and go to Step 3°.

5°. If \( (x', y') \in \tau \), then set \( W^* := W^* \cup \{(x', y' - 1)\}, B^* := B^* \cup \{(x, y), (x', y' - 1)\} \) and go to Step 6°. If \( (x', y') \in \tau \), then set \( W^* := W^* \cup \{(x' - 1, y')\}, B^* := B^* \cup \{(x, y), (x' - 1, y')\} \) and go to Step 6°. If \( (x', y') \not\in \tau \) and \( (x', y') \not\in \tau \), then set \( W^* := W^* \cup \{(x', y' - 1)\}, B^* := B^* \cup \{(x, y), (x', y' - 1)\} \) and go to Step 6°.

6°. Calculate the length of the new arcs added to the set \( B^* \) using formula (2). Go to Step 2°.
It is easy to convince that Algorithm 3 takes $O(r^2)$ time. Summarizing, we conclude that the whole Algorithm 3 takes $O(r)$ or $O(r^2)$ time depending on which step (Step 2, Step 4 or Step 6) is the last in the realization of Algorithm 3 (see line 7 in Table II).

In Fig. 1 the length of a shortest path from vertex $(0, 0)$ to each vertex $w_p \in W^*$ is presented near the point $w_p$. The following trajectory is optimal: $((0, 0), (1, 0), (2, 0), (6, 3), (7, 3), (9, 4), (12, 6), (13, 6), (17, 9), (20, 12), (20, 13), (22, 15), (22, 16), (23, 17))$ and $C^*_{max} = 25$. The point $(20, 12)$ is the above point $(x', y')$ which divides the optimal trajectory into two parts $\tau' = ((0, 0), (1, 0), (2, 0), (6, 3), (7, 3), (9, 4), (12, 6), (13, 6), (17, 9), (20, 12))$ and $\tau'' = ((20, 12), (20, 13), (22, 15), (22, 16), (23, 17))$ (see proof of Theorem 2).

3.2 Problem $J|j = 2, p_i = 1|C_{max}$

If Property 3 holds, then inequality $r_1 \geq 2r_2$ implies equality $y_1 = r_2$ in the right bounding trajectory $\tau^1_0 = ((0, 0), (x_1, y_1), (r_1, r_2))$. In Fig. 2 the worst cases’ of the trajectory $\tau^1_0 = ((0, 0), (x_1, y_1), (r_1, r_2))$ are shown if equality $r_1 = 2r_2$ holds. Thus, from Lemma 2 it follows that trajectory $\tau^1_0 = ((0, 0), (x_1, y_1), (r_1, r_2))$ is optimal. Therefore, problem $J|j = 2, p_i = 1|C_{max}$ is not trivial only if inequalities

$$0.5r_2 < r_1 < 2r_2$$

hold. Next, we show that in the general case problem $J|j = 2, p_i = 1|C_{max}$ can be solved in $O(r)$ time. Again without loss of generality, we assume that $r_1 \geq r_2$.

Fig. 2

Algorithm 4

1. Set $W^0 = \{(0, 0), (r_1, r_2)\}$ and $B^0 = \emptyset$.

2. Take a vertex $(x, y) \in W^0 \setminus \{(r_1, r_2)\}$ with zero outdegree, set $k = 1$ and go to Step 3. If there is no vertex in the set $W^0 \setminus \{(r_1, r_2)\}$ with zero outdegree, then go to Step 8.

3. Set $x' = x + k$, and $y' = y + k$. If $x' < r_1$, go to Step 4. Otherwise, set $B^0 := B^0 \cup \{(x, y), (r_1, r_2)\}$ and go to Step 6.

4. If $y' < r_2$, go to Step 5. Otherwise, set $B^0 := B^0 \cup \{(x, y), (r_1, r_2)\}$ and go to Step 6.

5. If $M_{x'} = M_{y'}$, go to Step 6. Otherwise, set $k := k + 1$ and go to Step 6.

6. (i) If $x' - y' < r_1 - r_2$, then set $W^0 := W^0 \cup \{(x', y' - 1)\}$, $B^0 := B^0 \cup \{((x, y), (x', y' - 1))\}$ and go to Step 7.

(ii) If $x' - y' > r_1 - r_2$, then set $W^0 := W^0 \cup \{(x' - 1, y')\}$, $B^0 := B^0 \cup \{((x, y), (x' - 1, y'))\}$ and go to Step 7.

(iii) If $x' - y' = r_1 - r_2$, then set $W^0 := W^0 \cup \{(x', y' - 1), (x' - 1, y')\}$, $B^0 := B^0 \cup \{((x, y), (x', y' - 1), (x, y), (x' - 1, y'))\}$ and go to Step 7.

7. Using formula (2), calculate the length of the new arcs included into the set $B^0$. Go to Step 2.
8. Find an optimal (shortest) path from vertex \((0,0)\) to vertex \((r_1, r_2)\) in the network \((W^0, B^0, \emptyset)\). Stop.

The correctness of Algorithm 4 follows from Theorem 3.

**Theorem 3** If Property 3 holds, then network \((W^0, B^0, \emptyset)\) contains an optimal path.

**Proof.** In Algorithm 4, a new vertex is added (or two new vertices are added) to the network in Step 6. In cases (i) and (ii), only one vertex is added, namely: the right-lower corner of square \(W_{x'y'}\) if \(x' - y' > r_1 - r_2\), or the left-upper corner of square \(W_{x'y'}\) if \(x' - y' < r_1 - r_2\). In case (iii), the vertices \((x' - 1, y')\) and \((x', y' - 1)\) are added to the network. We show by induction with respect to the number \(\lambda\) of squares hit going diagonally from the origin \((0,0)\) that network \((W^0, B^0, \emptyset)\) contains an optimal path.

It is easy to see that, if \(\lambda = 0\) or \(\lambda = 1\), Theorem 3 is valid. Let us make the inductive assumption that Theorem 3 is valid for number \(\lambda\) with \(0 \leq \lambda \leq k\). Moreover, let \(k + 1\) squares be hit via the realization of Algorithm 4.

First, we consider the case when \(r_1 = r_2\). Let \(S_{x'y'}, S_{x''y''}\) and \(S_{x''y''}\) be the first, second and third squares hit in the realization of Algorithm 4 (see Fig. 3). We can introduce two subproblems: the first subproblem in the rectangle \(S'\) with the corners \((x'' - 1, y'' - 1), (r_1, y'' - 1), (x'' - 1, r_2), (r_1, r_2)\), and the second subproblem in the rectangle \(S''\) with the corners \((x''' - 1, y''' - 1), (r_1, y''' - 1), (x''' - 1, r_2), (r_1, r_2)\).

For the subproblem in rectangle \(S'\), inequality \(x'' - y'' > r_1 - r_2\) holds. Thus, we have case (ii) in Step 6 and so vertex \((x'' - 1, y'')\) has to be added to the network. For the subproblem in rectangle \(S''\), inequality \(x''' - y''' > r_1 - r_2\) holds (see Fig. 3). Thus, we have case (i) in Step 6 and vertex \((x''' - 1, y''' - 1)\) has to be added to the network. Due to the inductive assumption, there exists an optimal trajectory \(\tau^*\) (trajectory \(\tau^{**}\)) for the subproblem in rectangle \(S'\) (in rectangle \(S''\), respectively) constructed by Algorithm 4. Thus, either trajectory \(((0, 0), (x' - 1, y'), (x''' - 1, y''' - 1), \tau^*)\) or trajectory \(((0, 0), (x' - 1, y'), (x''' - 1, y''' - 1), \tau^{**})\) is an optimal trajectory for the original problem in rectangle \(S\). It is easy to see that this trajectory belongs to network \((W^0, B^0, \emptyset)\).

The case when \(r_1 > r_2\) may be considered similarly to the above case \(r_1 = r_2\) (note that in the case \(r_1 > r_2\) only one subproblem in rectangle \(S'\) has to be considered).

Figure 4 presents the rectangle \(S\) and the network \((W^0, B^0, \emptyset)\) for the example of problem \(J_4[j = 2, p_i = 1|C_{max}]\). The machine order for job \(J_i\) is presented under the axis of \(x\)-coordinates. The machine order for job \(J_2\) is presented at the left of the axis of \(y\)-coordinates. The length of a shortest path from vertex \((0,0)\) to vertex \((x, y) \in W^0\) is shown near the point \((x, y)\). The following trajectory is optimal: \(((0, 0), (1, 0), (3, 1), (7, 4), (10, 6), (12, 7), (14, 10), (19, 14), (21, 15), (27, 21), (27, 22))\) and \(C^*_{max} = 29\).
Table II: Complexity of a mixed graph coloring with long paths or large cliques

<table>
<thead>
<tr>
<th>Notation of the scheduling problem</th>
<th>Number of paths in $(V, A, \emptyset)$</th>
<th>Number of cliques in $(V, \emptyset, E)$</th>
<th>Property</th>
<th>Complexity</th>
<th>References</th>
</tr>
</thead>
<tbody>
<tr>
<td>$J2</td>
<td>p_i = 1, rep</td>
<td>C_{max}$</td>
<td>$j$</td>
<td>$2$</td>
<td>no</td>
</tr>
<tr>
<td>$J3</td>
<td>p_i = 1</td>
<td>C_{max}$</td>
<td>$j$</td>
<td>$3$</td>
<td>yes</td>
</tr>
<tr>
<td>$J2</td>
<td>p_i = 1</td>
<td>C_{max}$</td>
<td>$j$</td>
<td>$2$</td>
<td>yes</td>
</tr>
<tr>
<td>$Jm</td>
<td>j = n, p_i = 1</td>
<td>C_{max}$</td>
<td>$n$</td>
<td>$m$</td>
<td>yes</td>
</tr>
<tr>
<td>$Jn</td>
<td>j = 1</td>
<td>C_{max}$</td>
<td>$j$</td>
<td>$n$</td>
<td>yes</td>
</tr>
<tr>
<td>$J[j = k, p_i = 1</td>
<td>C_{max}$</td>
<td>$j = k$</td>
<td>$m$</td>
<td>yes</td>
<td>$O(r^k)$</td>
</tr>
<tr>
<td>$J[j = 2, p_i = 1, rep</td>
<td>C_{max}$</td>
<td>$2$</td>
<td>$m$</td>
<td>no</td>
<td>$O(r)$ or $O(r^2)$</td>
</tr>
<tr>
<td>$J[j = 2, p_i = 1</td>
<td>C_{max}$</td>
<td>$2$</td>
<td>$m$</td>
<td>yes</td>
<td>$O(r)$</td>
</tr>
</tbody>
</table>

It is easy to see that $|W^0| \leq 2r$ (see Step 6 of Algorithm 4) and that constructing network $(W^0, B^0, \emptyset)$ takes $O(r)$ time (see Steps 1 - 6 of Algorithm 4). Finding a shortest (optimal) path in the network $(W^0, B^0, \emptyset)$ takes $O(r)$ time. Thus, solving problem $J[j = 2, p_i = 1|C_{max}$ takes $O(r)$ time as well (see line 8 in Table II). In contrast to Algorithm 2 described in Section 2, both Algorithm 3 and Algorithm 4 are practically efficient: within a reasonable CPU-time they solve problem $J[j = 2, p_i = 1, rep|C_{max}$ and problem $J[j = 2, p_i = 1|C_{max}$ with large numbers $n, m$ and $r$.

The main idea of the above geometrical approach is to go diagonally in the rectangle $S$ (i.e. to process both jobs $J_1$ and $J_2$ simultaneously) whenever it is possible. In Section 4, we use the same idea for an arbitrary number of jobs. Of course, if $j > 2$, the graphical presentation on the plane is not so useful as for the above case $j = 2$. By the way, for the case $j = 2$ all steps of Algorithms 3 and 4 may be realized analytically without any graphical presentation.

In Table II we present complexity results for an optimal coloring of a mixed graph $G$ with Properties 1 and 2 when $n \to \infty$. If the number $j$ of paths is fixed (see lines 6, 7 and 8), then $n \to \infty$ implies the existence of a long path, i.e. $r \to \infty$. If the number $m$ of cliques is fixed (see lines 1, 2 and 3), then $n \to \infty$ implies $j \to \infty$ or the existence of a large clique.

## 4 THE BRANCH-AND-BOUND ALGORITHMS

Three branch-and-bound algorithms for a mixed graph coloring will be determined in this section by the description of the solution tree, the branching procedure, lower bounds and upper bounds for the chromatic number $\gamma(G)$. These elements of the algorithms will be given using the example of problem $J5|j = 4, p_i = 1, rep|C_{max}$, whose mixed graph $G$ of order $n = 24$ is presented in Fig. 5.

![Fig. 5](image_url)

For this example, job $J_1$ has to be processed by the machines $M = \{M_1, M_2, M_3, M_4, M_5\}$ in the order $M_1, M_2, M_3, M_4, M_1, M_5$, job $J_2$ in the order $M_5, M_5, M_2, M_4, M_1, M_2, M_4$, job $J_3$ in
the order $M_1, M_2, M_3, M_4$, and job $J_4$ in the order $M_2, M_1, M_3, M_2, M_1, M_5$. In Fig. 5, the vertices $V = \{v_1, v_2, \ldots, v_{24}\}$ are shown as squares with index $i$ of vertex $v_i \in V$ being presented under the corresponding square. For simplicity, all edges $E$ of the mixed graph $G = (V, A, E)$ are omitted in Fig. 5. To indicate machine $M_k \in M$, which has to process operation $v_i$, different pictures are shown within the square corresponding to operation $v_i$. The machine order for job $J_1$ is also given at the top of Fig. 5 to indicate the correspondence of pictures to machines.

4.1 Solution tree and branching

The solution tree $T = (W, R, \emptyset)$ for this example is presented in Fig. 6. Each vertex $w^{(i)} \in W$ of a solution tree $T = (W, R, \emptyset)$ is a vector with $j$ integer components which describe the state of the coloring process (similarly to Algorithm 2 from Section 2). The first component $w_1^{(i)}$ of the vector ($state$) $w^{(i)} \in W$ corresponds to job $J_1 \in J$, the second component $w_2^{(i)}$ to job $J_2$, and so on, the last component $w_j^{(i)}$ to job $J_j$. At the initial iteration, the algorithm starts with vector $w^{(0)}$ each component of which is equal to zero: $w_k^{(0)} = 0, k = 1, 2, \ldots, j$, (see root vertex of the solution tree $T$ shown in Fig. 6). State $w^{(0)}$ means that no vertex of the mixed graph $G$ is colored. At the first iteration, the algorithm tries to color a maximum possible number of vertices from the set $V^1 = \{v_{11}, v_{12}, \ldots, v_{j_1}\}$ by the color 1, provided that such a coloring does not cause any conflict due to an edge existing in the mixed graph $G$. Remind that $v_{k_i}$ means the first operation in the path $(v_{k_1}, v_{k_2}, \ldots, v_{k_{r_k}})$, $k = 1, 2, \ldots, j$, (see Property 2).

For the example shown in Fig. 5, the first operation $v_1$ of job $J_1$ and the first operation $v_{13}$ of job $J_3$ cannot be colored by the same color, since these operations are connected by the edge $[v_1, v_{13}] \in E$ (since these operations have to be processed by the same machine $M_1$). Therefore, at the first iteration the algorithm generates two new vectors ($states$) $w^{(1)}$ and $w^{(2)}$ in the solution tree (it is a branching, see Fig. 6). In the state $w^{(1)}$, the first operations of the jobs $J_1, J_2$ and $J_4$ (i.e. operations $v_1, v_{13}$ and $v_{19}$) are colored by color 1. In the state $w^{(2)}$ the first operations of the jobs $J_2, J_3$ and $J_4$ (i.e. operations $v_7, v_{13}$ and $v_{19}$) are colored by color 1. The component $w_1^{(2)}$ corresponding to the first operation $v_1$ of job $J_1$ and the component $w_3^{(1)}$ corresponding to the first operation $v_{13}$ of job $J_3$, which are not colored at the first iteration, are equal to $-1$ to indicate that the first operations are not colored for these jobs at the first iteration of the algorithm. (In Fig. 6, index $i$ of vector $w^{(i)}$ is indicated at the left of the corresponding vector.)

In general, each component $w_2^{(i)}$ of the vector
from the solution tree $T$ either is equal to the order number $l$ of vertex $v_k$ from the path $(v_{k_1}, \ldots, v_{k_l}, \ldots, v_{k_r})$ (if vertex $v_{k_1}$ is colored at this iteration), or the component $w^{(i)}_k$ is equal to $-l$ in the case when at this iteration no operation of job $J_k$ is colored. Similarly to Algorithm 2, the component $w^{(i)}_k$ defines the state of coloring a path $(v_{k_1}, v_{k_2}, \ldots, v_{k_r})$, $k = 1, 2, \ldots, j$. At the current iteration, all vertices of the subpath $(v_{k_1}, v_{k_2}, \ldots, v_{k_r})$ are colored while the remaining vertices $(v_{k_1+1}, \ldots, v_{k_r})$ are not colored. Here $s^{(i)}_k = w^{(i)}_k$ if $w^{(i)}_k$ is positive, and $s^{(i)}_k = -w^{(i)}_k - 1$, if $w^{(i)}_k$ is negative.

The arc $(w^{(i)}, w^{(k)}) \in R$ in the solution tree $T = (W, R, \emptyset)$ connects vector $w^{(i)}$ with vector $w^{(k)}$ if state $w^{(k)}$ is generated from state $w^{(i)}$ and for generating state $w^{(i)}$ color $c$ was used, while for generating state $w^{(k)}$ color $c + 1$ was used. The corresponding colors $c$ are shown at the top of Fig. 6.

If the mixed graph $G$ has no edges between the vertices $v_{k_{s^{(i)}}}, k = 1, 2, \ldots, j$, which are ready to be colored at the current iteration, then they all are colored at this iteration and as a result, only one new state is generated in the solution tree (e.g, in Fig. 6, from the state $w^{(24)}$ only one state $w^{(25)}$ is generated). To overcome the conflict when some vertices from the set $S^{(i)} = \{v_{1_{s^{(i)}1}}, v_{1_{s^{(i)}2}}, \ldots, v_{j_{s^{(i)}1}}, v_{j_{s^{(i)}2}}, \ldots, v_{j_{s^{(i)}1}}\}$ cannot be colored by the same color due to the existence of edges in the mixed graph $G$, the algorithm uses a branching of the set of possible colorings, i.e., the algorithm generates several new states for all possible alternatives for coloring vertices by the current color.

For the example under consideration, at the second iteration three new states $w^{(3)}, w^{(4)}$ and $w^{(5)}$ are generated from the state $w^{(2)}$ (see Fig. 6).

In general, the number $\lambda^{(i)}$ of states generated in the solution tree $T$ from the state $w^{(i)}_1 \in W$ may be calculated as follows. Let $(S^{(i)}, \emptyset, E^{(i)})$ be the subgraph of the mixed graph $G$ and $(S^{(i)}, \emptyset, E^{(i)}) = (V^{(i)}_1, \emptyset, E^{(i)}_1) \cup (V^{(i)}_2, \emptyset, E^{(i)}_2) \cup \ldots \cup (V^{(i)}_m, \emptyset, E^{(i)}_m)$ be a partition of the graph $(S^{(i)}, \emptyset, E^{(i)})$ into cliques (see Property 1). Then we have

$$\lambda^{(i)} = \prod_{1 \leq k \leq m, |V^{(i)}_k| > 0} |V^{(i)}_k|.$$

The terminal state of the solution tree $T$ is either that which defines a coloring of all vertices $V$ of the mixed graph $G$ or that for which the lower bound for the chromatic number $\phi(G)$ calculated for this state is not less than the upper bound for $\phi(G)$. In all three algorithms, the upper bound is equal to the record (minimal) number of colors used for the best coloring constructed at previous iterations.
Note that in Fig. 6 vector \( w^{(20)} \) (vector \( w^{(33)} \)) has only one non-empty component (only three non-empty components, respectively) while the other components of these vectors are empty. Such situations hold since at the previous iterations of the algorithm, the whole operations of sets \( V^{(2)} \), \( V^{(3)} \) and \( V^{(4)} \) of the jobs \( J_2, J_3 \) and \( J_4 \) (the whole operations of set \( V^{(1)} \) of job \( J_1 \), respectively) have been colored.

### 4.2 Lower and upper bounds

Two lower bounds (for brevity \( LB \)) for the chromatic number \( \gamma(G) \) have been tested in the experiments. The *global lower bound* \( LB1 \) is based on fixing the machine \( M_k \in M \) and calculating the sum of the cardinality of the set \( V_k \) and the minimum number \( h_k^d \) (the minimum number \( t_k^d \)) of operations before the first operation (after the last operation), which needs machine \( M_k \):

\[
\gamma(G) \geq LB1 = \max_{M_k \in M} \left\{ \min_{J_d \in J_k} |V_k| + \min_{J_d \in J_k} t_k^d \right\}.
\]

The *local lower bound* \( LB2 \) is equal to the maximum of the sums \( r_k + l_k \) calculated for each job \( J_k \in J \), where \( l_k \) denotes the number of colors which are omitted for the operations of job \( J_i \) at the previous iterations of the algorithm. Let state \( w^{(i)} \) be generated using color \( c^{(i)} \), then \( l_k = c^{(i)} - |S_k^{(i)}| \). Thus, we have

\[
\gamma(G) \geq LB2 = \max_{J_d \in J} \{r_k + l_k\} = \max_{J_d \in J} \{r_k + c^{(i)} - |S_k^{(i)}|\}.
\]

It should be noted that to calculate \( LB2 \) at the current iteration \( i \), we need only vector \( w^{(i)} \) and vector \( (r_1, r_2, \ldots, r_j) \). So, \( LB2 \) can easily be calculated.

At the initial iteration, all three branch-and-bound algorithms use the trivial upper bound \( UB0 = \sum_{i=1}^{j} |V_i| \). The number of colors used in the record coloring (i.e., in the best coloring constructed before the current iteration) is used as an upper bound (for brevity \( UB \)) for the chromatic number \( \gamma(G) \).

We coded three branch-and-bound algorithms depending on the lower bound used and on the selection of a vertex for branching from the partial solution tree constructed.

### 4.3 Selecting a vertex in the solution tree

The depth-first search strategy is used in all three branch-and-bound algorithms. The first and the second algorithms use the global lower bound \( LB1 \). In the first algorithm (we call it GLOBAL-1), the state \( w^{(i)} \in W \) is selected from the set \( W^* \) of all states generated at the current iteration if for this state \( w^{(i)} \), the lower bound for \( \gamma(G) \) has the minimum value among all other states from the set \( W^* \). If state \( w^{(i)} \) defines a coloring of all vertices \( V \) of the mixed graph \( G \), then the algorithm selects the state \( w^{(r)} \) for the next branching which has the minimum value of \( LB1 \). If such a state \( w^{(r)} \) is not uniquely determined, then one with the longest path from the root state \( w^{(0)} \) to the state \( w^{(r)} \) is selected for the next branching.

The solution tree of algorithm GLOBAL-1 for the example under consideration is presented in Fig. 6. An optimal coloring for this example obtained by Algorithm GLOBAL-1 is given.
in Fig. 7. The colors used for the corresponding vertices of the mixed graph $G$ are presented at the bottom of Fig. 7.

Fig. 7

The second algorithm (we call it GLOBAL-2) works as follows. If there are no terminal states among the states $W^*$ just generated, algorithm GLOBAL-2 selects the last state with minimal value $LB_1$. If there exists a terminal state among the vertices $W^*$ just generated, algorithm GLOBAL-2 selects the state with minimal lower bound $LB_2$ less than $UB_0$.

The third algorithm (we call it LOCAL) uses only $LB_2$ which is calculated very fast, but it is usually smaller than $LB_1$. Algorithm LOCAL uses the same rule as algorithm GLOBAL-1 for selecting a state for branching from the solution tree constructed.

5 COMPUTATIONAL RESULTS

All three branch-and-bound algorithms have been implemented in C++ and tested on a PC Pentium II-350 with 133 MB RAM. In Table III and Table IV, the computational results obtained for an optimal coloring of mixed graphs with Properties 1 and 2 are reported for Algorithms GLOBAL-1 and GLOBAL-2. Table V and Table VI present computational results for approximate mixed graph colorings, i.e. for restricted versions of Algorithms GLOBAL-1 and LOCAL with a limited number of vertices in the solution tree being tested.

Tables III - VI

In all experiments we considered (pseudo)random mixed graphs with $r_1 = r_2 = \ldots = r_j$. As follows from Section 3, such a mixed graph is more difficult for the approach used in Algorithms 3 and 4. Our preliminary computer experiments have shown that for Algorithms GLOBAL-1, GLOBAL-2 and LOCAL mixed graphs with $r = r_1 = r_2 = \ldots = r_j$ were also more difficult in comparison with mixed graphs with different lengths of paths.

The running times in seconds are presented in the last column of Table III and Table IV for randomly generated mixed graphs $G$ of orders $n = 100$ and $n = 200$, respectively. The first column in each table denotes the main parameters of the mixed graphs: the number of cliques (the number of machines $m$), the number of paths (the number of jobs $j$), the length of the path (the number $r = r_k$ of operations per job $J_k$), the number $|A|$ of arcs (the number of pairs of sequential operations of jobs $J_k$, $k = 1, 2, \ldots, j$), and the number $|E|$ of edges (the number of pairs of the operations $V$ which have to be processed by the same machine). Each row in Table III and Table IV represents the results for 10 (pseudo) random instances of a series of mixed graphs with the same parameters $m, j, r_k, |A|$ and $|E|$, the number $r_k$ of operations per job $J_k$ being the same for all jobs in each instance: $r_k = r$.

The last column of the tables contains the average value of the CPU-time (for all 10 instances in the series). The second column contains the average first lower bound constructed for the chromatic number. The third column contains the average value of the chromatic number. The fourth column is equal to the percentage of problems for which the record coloring constructed was proven to be optimal within a limited number $L$ of vertices in the solution.
tree $T$: $|W| \leq L$. If the fourth column is not equal to 100, it means that for at least one instance in this series, the upper limit $L$ of the number of vertices $W$ in the solution tree $T = (W, R, \emptyset)$ was not sufficient to prove the optimality of the record coloring constructed. In the experiments reported, limit $L$ was assumed to be equal to 30,000,000 in the case of optimal colorings, i.e. for the results presented in Tables III and Table IV. The fifth column contains the average differences $UB - LB$ for the 10 instances in the series. If for all 10 instances optimal colorings were constructed and their optimality were proven, then $UB - LB = 0$.

We tested all three branch-and-bound algorithms for the exact solution of a mixed graph coloring problem. However, Table III and Table IV represent only results for Algorithm GLOBAL-1 in the odd lines and for Algorithm GLOBAL-2 in the even lines. Algorithm LOCAL found optimal colorings for a smaller number of problems than Algorithms GLOBAL-1 or GLOBAL-2. Moreover, Algorithm LOCAL often constructs a larger solution tree (and as a result it needs more CPU-time) to prove the optimality of the record coloring constructed. Thus, we can conclude that Algorithm LOCAL is worse for an optimal coloring than Algorithm GLOBAL-1 and Algorithm GLOBAL-2. Computational results for an optimal coloring of mixed graphs of orders $n = 120$, $n = 150$, $n = 180$ and $n = 200$ for all three branch-and-bound algorithms are given in [18, 19].

Table V and Table VI contain the same columns as Table III and Table IV with the only exception that the third column contains the average values of $UB$ instead of the average values of $\gamma(G)$. For the approximate solution, we also tested all three algorithms but Tables V and VI present computational results for Algorithm GLOBAL-1 in the odd lines and for Algorithm LOCAL in the even lines. For most series the difference $UB - LB$ was larger for Algorithm GLOBAL-2 than that for Algorithm GLOBAL-1. If the difference $UB - LB$ was larger for Algorithm GLOBAL-1, the CPU-time was often smaller for Algorithm GLOBAL-1 than that for Algorithm GLOBAL-2. In all series the CPU-times for Algorithm GLOBAL-2 and GLOBAL-1 were larger than the CPU-times for Algorithm LOCAL. Algorithm LOCAL is faster: the largest CPU-time for constructing the solution tree $T = (W, R, \emptyset)$ with $|W| = L = 2,000,000$ was equal to 46.116 s. (For the approximate solution, the limit $L$ was equal to 2,000,000: $|W| \leq 2,000,000$.) The quality of the coloring constructed by Algorithm LOCAL was not much worse than that by Algorithm GLOBAL-1 and Algorithm GLOBAL-2.

More detailed computational results for mixed graph colorings are given in [19]. In particular, the algorithms were tested on randomly generated mixed graphs of orders 120, 150, 180, and 600.

6 CONCLUSION

The unit-time job-shop problem $J|p_i = 1|C_{max}$ may be interpreted as an optimal coloring $\phi$ of the vertices of a mixed graph $G$, which has Property 1 and Property 2. All the above results observed and developed may be presented using only mixed graph terminology which is widely used in OR literature. E.g. one can use terms vertex, path, cliques, coloring, etc. instead of terms operation, job, machine, schedule, etc. Such an interpretation of the known and new results seems to be useful for the reader that prefers the graph terminology instead of the terminology of scheduling theory. Due to such an interpretation, a lot of results from
scheduling theory may be considered as a branch of graph theory. Since \( p_i = 1 \) for each \( v_i \in V \), one can consider the scheduling problem \( J \mid p_i = 1 \mid C_{\text{max}} \) as one ‘without numerical input data’, and so the complexity of such a problem depends on the ‘structural input data’ presented via the mixed graph \( G \) (see Tables I and II).

In this paper, we have shown that an optimal coloring may be constructed in \( O(r^k) \) time, if the number \( j \) of paths in \( G \) is a constant \( k \) and Property 3 holds. We have also shown that an optimal coloring may be constructed in \( O(r) \) time (in \( O(r^2) \) time, respectively) if \( j = 2 \) and the mixed graph \( G \) has Property 3 (has not Property 3). For the general case of a mixed graph \( G \) with Properties 1 and 2, we developed three branch-and-bound algorithms which demonstrate acceptable computational results for randomly generated mixed graphs with Properties 1 and 2. Algorithms GLOBAL-1 and GLOBAL-2 were able to find an optimal coloring for almost all instances of the mixed graph \( G \) with a number of vertices \( n \leq 200 \) and with 1000 edges, the number of arcs being not critical for a mixed graph coloring. For one of the hardest series of mixed graphs with \( n = 200, m = 20, j = 20, r = 10, |A| = 180 \) and \( |E| = 1003 \), for 7 instances from 10 instances considered it was proven that the record coloring is optimal.

The best results for the approximate solution of a mixed graph coloring problem were obtained by Algorithm LOCAL, which is able to construct colorings of mixed graphs with \( n \leq 900 \) and about 25,000 edges, the difference between the upper bound \( UB \) and the lower bound \( LB \) for the chromatic number \( \gamma \) being not much worse than that for Algorithm GLOBAL-1 or Algorithm GLOBAL-2. To obtain such a coloring, Algorithm LOCAL takes no more than one minute of CPU-time. Thus, only memory limit seems to be critical for the current version of Algorithm LOCAL (which uses only internal memory). Algorithm LOCAL is so fast that it may be used for calculating an upper bound for \( \gamma(G) \) within Algorithms GLOBAL-1 or GLOBAL-2, which are able to get a better approximation (than Algorithm LOCAL), but need essentially more CPU-time.

It should be noted that Algorithms GLOBAL-1, GLOBAL-2 and LOCAL may be considered as a generalization of the geometrical approach (see Algorithms 3 and 4 in Section 3) to the case when the number of paths \( j \) is more than 2. Indeed, at each iteration, each of these algorithms tries to use the next (recurrent) color for the most possible vertices of the mixed graph \( G \), and if a conflict arises, these algorithms consider all possible alternatives (if the limit \( L \) of the vertices of the solution tree is sufficient). Note also that all algorithms developed in this paper (except Algorithm 1) are based on a presentation of the coloring process via a network. In Algorithms 2, 3 and 4, the search of a shortest path in a network is used as an essential part.

In conclusion we note that the mixed graph coloring model may be used for a more general scheduling problem with the makespan criterion and the assumption \( p_i = 1 \) for each \( v_i \in V \). The latter assumption is realized in many practical scheduling scenarios, e.g., in scheduling lectures and lessons at the university and at the school, in scheduling games in sport competitions when each game needs the same time, in scheduling medical procedures in the hospital, etc.

Acknowledgements:

The authors acknowledge G.V. Andreev for the qualified help in coding the algorithms and Prof. M.-C. Portmann for useful comments on an earlier draft of the paper. This research
was supported by INTAS (Project 96-0820) and ISTC (Project B-104-98). The first author acknowledges NATO for financial support during his stay at the University of Technology of Troyes, where this research was finished.

References


CAPTIONS OF FIGURES

Figure 1. Solution of problem $J4|j = 2, p_i = 1, rep|C_{max}$ by Algorithm 3

Figure 2. The right bounding trajectory $\tau_0^1$ when $r_1 = 2r_2$

Figure 3. Three squares which generate two subproblems: in the rectangle $S'$ and in the rectangle $S''$

Figure 4. The network $(W^0, B^0, \emptyset)$ for the example of problem $J4|j = 2, p_i = 1|C_{max}$

Figure 5. Mixed graph $G$ with Properties 1 and 2 (all edges are omitted)

Figure 6. Solution tree for Algorithm GLOBAL-1

Figure 7. An optimal coloring constructed by Algorithm GLOBAL-1
Figure 1. Solution of problem $J4|j = 2, p_i = 1, rep|C_{\text{max}}$ by Algorithm 3
Figure 2. The right bounding trajectory $\tau_0^1$ when $r_1 = 2r_2$.
Figure 3. Three squares which generate two subproblems: in the rectangle $S'$ and in the rectangle $S''$. 
Figure 4. The network \((W_0, B^0, \emptyset)\) for the example of problem \(J4|j = 2, p_i = 1|C_{\text{max}}\)