ZESZYTY NAUKOWE POLITECHNIKI ŚLĄSKIEJ 2002
Seria: AUTOMATYKA z. 134 Nr kol. 1554

Jacek BŁAŻEWICZ[1)], Klaus ECKER[2)], Tamás KIS[3)], Michał TANAŚ[1)]
1) Institute of Computing Science, Poznań University of Technology, Poznań, Poland
2) Institute of Computing Science, Technical University Clausthal, Germany
3) University of Siegen, Germany

SCHEDULING COUPLED TASKS ON A SINGLE PROCESSOR

Summary. This paper considers a problem of coupled task scheduling on one processor, where all processing times are equal to 1, the gap has exact length h, precedence constraints are strict and the criterion is to minimize the schedule length. This problem is introduced e.g. in systems controlling radar operations. We show that the general problem is NP-hard. This paper also shows a fast approximation algorithm for chain precedence constraints.

SZEREGOWANIE ZADAŃ SPRZĘŻONYCH NA JEDNYM PROCESORZE

Streszczenie. W referacie zaprezentowano problem szeregowania zadań sprzężonych na jednym procesorze, z jednostkowymi czasami wykonywania operacji, stałą długością przerwy pomiędzy operacjami, gdzie celem jest minimalizacja długości uszeregowania. Problem ten często występuje w praktyce w systemach sterowania urządzeniami radarowymi. W referacie pokazujemy NP-trudność problemu w przypadku ogólnych ograniczeń kolejnościowych oraz szybki algorytm aproksymacyjny dla ograniczeń kolejnościowych typu „łańcuch".

1. Introduction

A scheduling problem is, in general, a problem answering a question of how to allocate some resources over time in order to perform a given set of tasks [1]. In practical applications resources are processors, money, manpower, tools, etc. Tasks can be described by a wide range of parameters, like ready times, due dates, relative urgency factors, precedence constraints and many more. Different criteria can be applied to measure the quality of a schedule. The general formulation of scheduling problems and the commonly used nota-

tion can be found in books such as [5], [15], [2], [14], [3] and [4]. A survey of the most important results is given in [10].

One branch of scheduling theory is concerned with scheduling of coupled tasks. A task is called *coupled* if it contains two operations where the second has to be processed some time after a completion of the first one. This problem, described in [16] and [17], often appears in radar-like devices, where the first operation is the transmission of an electromagnetic pulse and the second is the reception of its echo. Several algorithms designed to solve the problem of radar pulse scheduling can be found in [8] and [11].

The complexity of various scheduling problems with coupled tasks has been studied in [12]. Although most of the cases are NP-hard [12], some polynomial algorithms were found in [13].

A coupled task scheduling problem with variable length gap is surveyed in [9] and [6]. NP-hardness of this case is proved in [18], where some interesting connections between coupled tasks and flow shops are also given.

In this note, we complement the above results by presenting the NP-completeness proof for the problem of scheduling coupled tasks on a single processor, with all processing times equal to 1, exact, integer gap length, general strict precedence constraints and the optimization criteria of minimizing the schedule length.

An organization of the paper is as follows. The problem is formulated in Section 2. The NP-hardness proof is presented in Section 3. The approximation algorithm is presented in Section 4. We conclude in Section 5.

2. Problem formulation

We consider the problem of scheduling n coupled tasks on a single machine, where each coupled tasks T_i consists of two operations T_{i1} and T_{i2} and a *gap* between them. During the gap, another task can be processed. Let p_{i1} and p_{i2} denote the processing times of operations T_{i1} and T_{i2}, respectively.

The gap is exact when operation T_{i2} has to start exactly h_i units of time after the end of operation T_{i1}, where h_i denotes a length of the gap. In this paper, the only cases considered are those where all h_i are equal, i.e. $h_i = h$, $i = 1, 2, \ldots, n$.

Precedence constraints of coupled tasks can be strict or weak. $T_i \prec T_j$ means that $T_{i2} \prec T_{j1}$ if precedence constraints are *strict*, and $T_{i2} \prec T_{j1} \wedge T_{i2} \prec T_{j2}$ if they are *weak*.

The special case of a coupled task problem involves identical tasks. Commonly, such tasks are denoted by (p_1, h, p_2), where $p_1 = p_{i1}$, $p_2 = p_{i2}$, $h = h_i$ for all $1 \leq i \leq n$.

Adapting the commonly accepted notation for scheduling problems [7], the scheduling problem considered in this paper can be denoted by $1|(1, h, 1) - coupled, strict\ prec, exact\ gap|C_{max}$, which means:

- There is one processor in a system.
- Tasks are coupled and identical, with processing times $p_{i1} = p_{i2} = 1$, $\forall_{1 \leq i \leq n}$
- Gaps are exact and have uniform length h.
- Every operation has a processing time equal to 1, and the length of
- Precedence constrains are strict.
- The optimization criterion is to minimize the schedule length $C_{max} = \max\{t_{j2}\}$, where t_{j2} is a completion time of T_j (its second operation).

3. NP-hardness of the general case

In case where precedence constraints are general the problem of scheduling coupled tasks on a single processor is NP-hard even for unit processing times. We will prove this by a series of lemmae showing NP-hardness of some intermediate problems.

Lemma 1. Problem of *Balanced Coloring of Graphs with Partially Ordered Vertices* is NP-hard.

Proof: The problem of *Balanced Coloring of Graphs with Partially Ordered Vertices* (*BCGPOV* for short) can be stated as follows:

Instance: A directed, acyclic graph $G = (V, E)$ where $|V| = q$. (It is clear that the arcs define a partial order in set V.)

Question: Can the vertices of G be colored with l colors such that no pair of adjacent vertices shares the same color and exactly q/l vertices are colored with the same color.

(We will call such a coloring the *balanced coloring.*) Without loss of generality we can assume that $q/l \in \mathbb{Z}^+$.

Firstly, we prove that *BCGPOV* belongs to *class NP.* To verify a solution of *BCGPOV* is enough to verify that

1. No pair of adjacent vertices is monochromatic. Because each vertex has no more than $(q-1)$ adjacent vertices the complexity of this step is $O(q^2)$.

2. Exactly q/l vertices are colored with each color. Complexity of this step is $O(q)$.

A solution of *BCGPOV* can be verified in polynomial time, which means that the problem *BCGPOV* belongs to *class NP.*

In order to prove NP-completeness of the problem *BCGPOV* we will use the *3-Partition* problem defined as follows:

Instance: A collection of $3r$ items, bound $B \in \mathbb{Z}^+$, and size $s(a_j) \in \mathbb{Z}^+$, $\forall_{a_j \in A}$ such that $B/4 < s(a_j) < B/2$ and such that $\sum_{a_j \in A} s(a_j) = rB$.

Question: Can A be partitioned into r disjoint sets $A_1, \ldots, A_r$ such that for $1 \leq i \leq r$, $\sum_{a_j \in A_i} s(a_j) = B$ (note that each A_i must contain exactly three elements from A)?

The transformation: For any instance of the *3-Partition* problem let us define the corresponding instance of the *BCGPOV* problem as follows:

- $q = 3r^2 + rB$
- $l = r$
- For each item $a_j \in A$, $1 \leq j \leq 3r$ let us construct graph G_j in the following way:

  1. Construct complete graph K_r^j on r vertices. Denote one vertex of K_r^j by v_j.
  2. Construct set $D^j_{s(a_j)}$ of $s(a_j)$ independent vertices.
  3. Create all possible edges (v_x, v_y) such that $v_x \in K_r^j$, $v_x \neq v_j$ and $v_y \in D^j_{s(a_j)}$.

  Graph G_j is fully defined by a triple $(j, r, s(a_j))$ so the construction has the complexity of $O(\log \max\{j, r, s(a_j)\})$. It is clear that edges of graph G_j can be directed to define a partial order in the set of vertices of G_j.

  An example of such a graph is shown in Fig. 1.

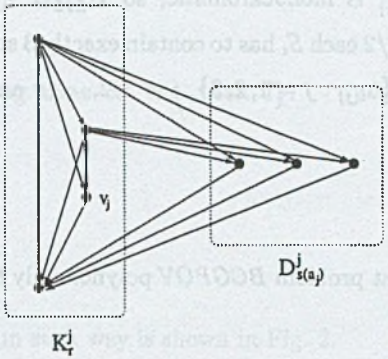

Fig. 1. An example of graph G_j, where $q = 4, s(a_j) = 3$
Rys. 1. Przykład grafu G_j, dla $q = 4, s(a_j) = 3$

- $G = \bigcup_{i=1}^{3r} G_j$

The complexity of this transformation is $O(r + \log B)$.

It is clear that G_j can be colored with r colors only in the following way: each vertex from K^j_r has a different color and all vertices from $D^j_{s(a_j)}$ are colored with the same color as vertex v_j.

No two vertices of any subgraph K^j_r, $1 \leq j \leq 3r$ can share the same color, so after coloring of all K^j_r, exactly $3r$ vertices are colored with the same color. All vertices of $D^j_{s(a_j)}$ are connected with all but v_j vertices of K^j_r, so all vertices of $D^j_{s(a_j)}$ have to be colored with the same color as vertex v_j. So, the set of vertices of any $D^j_{s(a_j)}$ has to be monochromatic.

Let $A_1, \ldots, A_j$ be a solution for the given instance of *3-Partition* problem and let $A_i = \{a_{i(1)}, a_{i(2)}, a_{i(3)}\}$. Let us denote $S_i = G_{i(1)} \cup G_{i(2)} \cup G_{i(3)}$. G can be colored such that sets D_i and D_j share the same color if and only if both D_i and D_j are in the same S_i. It means that for all i exactly $s(a_{i(1)}) + s(a_{i(2)}) + s(a_{i(3)}) = B$ vertices of sets D, so exactly $3r + B$ vertices of graph G, are colored with each color.

On the other hand, let $S_1, S_2, \ldots, S_r$ be a disjoint sets of vertices of graph G, such that $V(G) = \sum_{i=1}^{r} S_i$, each vertex in S_i is colored with i-th color and $\forall_i\ |S_i| = 3r + B$. Let S_i^D be a subset of S_i that contains only vertices belonging to $D^j_{s(a_j)}$ subgraphs. S_i has to contain $3r$ vertices belonging to K^j_r, so $|S_i^D| = B$ for each i. For each fea-

sible coloring each $D^j_{s(a_j)}$ is monochromatic, so $\forall_{1\leq j\leq 3r}\ \exists_{1\leq i\leq r}\ D^j_{s(a_j)} \in S_i$. Because $\forall_{1\leq j\leq 3r}\ B/4 < s(a_j) < B/2$ each S_i has to contain exactly 3 subgraphs $D^{i(j)}_{s(a_{i(j)})}$, $j = 1, 2, 3$. So, we can denote $A_i = \{a_{i(j)} : j = 1, 2, 3\}$, $i = 1, 2, \ldots, r$ and this is the solution of the *3-Partition* problem.

□

Now, we will show that problem *BCGPOV* polynomially transforms to our scheduling problem.

Lemma 2. The problem of *Balanced Coloring of Graphs with Partially Ordered Vertices* polynomially transforms to $1|(1, h, 1) - coupled, strict\ prec, exact\ gap|C_{max}$.

Proof: Let $G = (V, E)$ be an instance of problem *BCGPOV*. Let it contain all transitive arcs. Let us define a corresponding instance of $1|(1, h, 1) - coupled, strict\ prec, exact\ gap|C_{max}$ problem in the following way:

- $n = q$
- $h = q/l - 1$
- For each vertex v_i of graph G define the coupled task T_i.
- For each arc (v_i, v_j) of graph G define an arc $T_i \prec T_j$ of precedence constraints in the scheduling problem.
- $y = C_{max} = 2n$.

Let us assume that a balanced coloring of G exists. Let $S_1, S_2, \ldots, S_l$ be subsets of $V(G)$ such that $\bigcup_{i=1}^{l} S_i = V(G)$, and all vertices that belong to S_i are colored with the i-th color and $\forall_{1\leq i\leq l}\ |S_i| = q/l$. Sets S_i are partially ordered because vertices of G are partially ordered and G contains all transitive arcs. Schedule sets S_i in accordance to the partial order using the following algorithm:

Algorithm 1

begin

$s := 0$

repeat

Get a task $T_j \in S_i$.

Let s be the starting time of the task T_j.

$S_i := S_i / T_j$

$s := s + 1$

until $S_i \neq \emptyset$

end;

The schedule generated in such way is shown in Fig. 2.

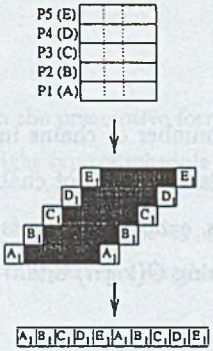

Fig. 2. Schedule of set $S_i = \{T_a, T_b, T_c, T_d, T_e\}$ where $h = 4$
Rys. 2. Uszeregowanie zbioru $S_i = \{T_a, T_b, T_c, T_d, T_e\}$ dla $h = 4$

This procedure guarantees that no precedence constraint will be violated. The schedule contains no idle intervals, so its length is y.

On the other hand, let us assume that a schedule of length y for the given instance of the coupled tasks problem exists. The schedule does not contain idle intervals, so it has to be a sequence of segments as shown in Fig. 2. Each segment contains $h+1$ independent tasks, which means that the corresponding vertices in G are also independent. So, the vertices from one segment can be colored with the same color, which means G can be colored with $n/(h+1)$ colors such that exactly $h+1$ vertices shares the same color.

□

Now we can conclude.

Theorem 1. Problem $1|(1,h,1) - coupled,strict\ prec,\ exact\ gap|C_{max}$ is NP-hard.

Proof: Follows immediately Lemmae 1 and 2.

□

4. The approximation algorithm

In this section we present a fast approximation algorithm for the problem of scheduling coupled tasks on a single processor, with all processing times equal to 1, exact, integer gap of length $2k$, general strict precedence constraints and the optimisation criteria of minimising the schedule length.

4.1. The algorithm

Let c denote the number the number of chains in precedence graph. Let s_j and f_j denote respective starting and finishing times of chain C_j. An instance of the problem $1|(1,2k,1) - coupled,\ strict\ chains,\ exact\ gap|C_{max}$ is fully described by given lengths of the chains, so it can be encoded using $O(\log n)$ bits.

Algorithm 2.

Input: A set of chains of coupled tasks.

Step 1: We can consider a chain of coupled tasks as one preemptive task, so our problem converts to $2k+1|pmtn|C_{max}$ with preemptions allowed only in integer points of time.

Step 2: Sort the preemptive tasks in descending order of their lengths.

Step 3: Compute value:

$$D = \max\left\{\left\lceil \frac{1}{2k+1}\cdot\sum_{i=1}^{c} p_i \right\rceil,\ \max_{1\le j\le c} p_j\right\} \tag{1}$$

Step 4: Solve the preemptive problem using McNaughton's rule in time window of length D. Denote a completion time of the last task processed on machine P_i by b_i.

Step 5: If chain C_j is scheduled in such a way that

$$s_j < b_i < f_j \qquad i \neq j \tag{2}$$

for any $1 \le i \le 2k+1$, it should be split into two subchains C_j^1 and C_j^2, whose processing finishes and starts at time b_i respectively. Note that no chain splits into more than $2k+1$ subchains.

Output $b_1, b_2, \ldots, b_{2k-1}$ and a set of fours (j, i, s_j^i, f_j^i) where j is the number of chain, i is the number of subchain, s_j^i and f_j^i are respective the start and the finish time of the subchain in the preemptive schedule.

□

The total complexity of the algorithm is

$$\overbrace{c}^{\text{step 1}} + \overbrace{c \log c}^{\text{step 2}} + \overbrace{c}^{\text{Step 3}} + \overbrace{c + 2k + 1}^{\text{Step 4}} + \overbrace{c(2k+1)}^{\text{Step 5}} = O(c \log c)$$

for fixed k.

4.2. Segments

Now we have the schedule still in the preemptive form. It should be converted back to the coupled tasks form by choosing the right type of schedule (called *segment*) in the following way:

- S1: Schedule all subchains C_j^i such that $f_j^i \leq b_{2k+1}$ in the way shown in Fig. 2.
  The coupled tasks schedule of this part has length of $2(2k+1)b_{2k+1}$ and contains no idle time units.

- S2: Compute
  $$b'_{2k} = b_{2k} - R(b_{2k} - b_{2k+1}, 2k+1)$$
  where $R(x, y)$ is the remainder of division x by y. Schedule all subchains C_j^i such that $s_j^i \geq b_{2k+1}$ and $f_j^i \leq b'_{2k}$ in the way shown in Fig. 3.
  Notice that $2k+1$ such transformations covers entire rectangular area in the preemptive schedule, as shown in Fig. 4. The coupled tasks schedule of this part has length of $2 \cdot 2k(2k+1)(b'_{2k} - b_{2k+1}) + 2$ and contains 2 units of idle time regardless of the length of this part.

- S1′: Schedule all subchains C_j^i such that $s_j^i \geq b'_{2k}$ and $f_j^i \leq b_{2k}$ in form of segment *S1* (shown in Fig. 2) without the task E, which renders 1 unit of idle time per segment. The couple tasks schedule of this part has length $(2 \cdot 2k+1)(b_{2k} - b'_{2k})$ and contains $(b_{2k} - b'_{2k})$ idle time units. Off course $(b_{2k} - b'_{2k}) \leq 2k$.

- S3: Schedule all subchains C_j^i such that $s_j^i \geq b_{2k}$ in the way shown in Fig. 5. The coupled tasks schedule of this part has length of $(2k+2)(b_1 - b_{2k}) + 2 \cdot \max\{i : 2 \leq$

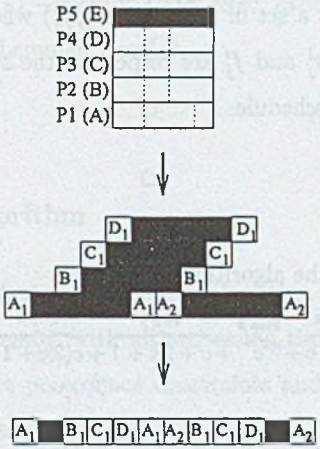

Fig. 3. Conversion to schedule type (segment) S2. Only one time slot is converted (using a task borrowed from the next one) in the figure. In the next time slot the two tasks should be taken from another processor

Rys. 3. Konstrukcja segmentu S2. Wykorzystano jedną jednostkę czasu z uszeregowania dla problemu $2k+1|pmtn|C_{max}$ i jedno zadanie "pożyczone" z kolejnej jednostki

Fig. 4. A few "time slots" that are used one after another to construct a segment S2. Notice that the shape remains rectangular if its length can be divided by $2k+1$ without remainder

Rys. 4. Po utworzeniu $2k+1$ kolejnych segmentów S2, wykorzystana część uszeregowania dla problemu $2k+1|pmtn|C_{max}$ odzyskuje kształt prostokąta

this part if $b_{2k-1} = b_1$. If $b_{2k-1} < b_1$ the number of idle time slots is not important — the machine $P1$ process only one long chain, so shortening of the schedule will violate precedence constraints.

4.3. Worst case analysis

There are only three following forms of the preemptive schedule possible, in general.

- A part $S1$ followed by a part $S2$ followed by a part $S1'$. The schedule contains no more than $0+2+2k$ units of idle time, so

$$C_{max} - C_{max}^{opt} \leq 2k+2$$

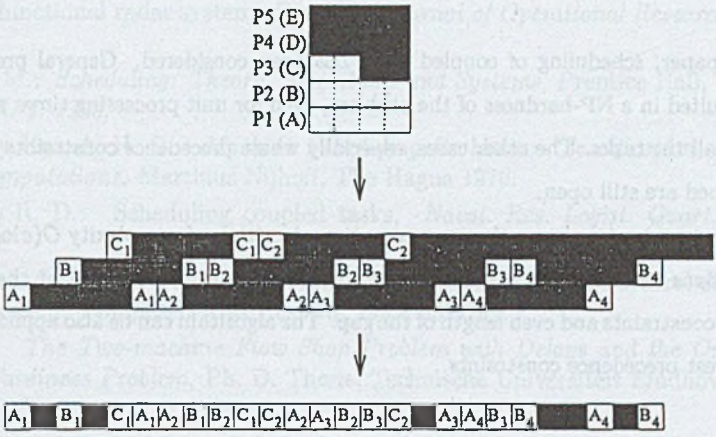

Fig. 5. Conversion to schedule type (segment) S3
Rys. 5. Konstrukcja segmentu S3

- A part $S2$ followed by a part $S1'$ followed by a part $S3$. In this case $b_{2k-1} = b_1$, so the schedule contains no more than $2 + 2k + 2k$ units of idle time, so

$$C_{max} - C_{max}^{opt} = 4k + 2$$

- A part $S3$ only. In this case the instance contains at least one chain of length b_1, so

$$C_{max}^{opt} \geq (2k + 2)b_1$$

The length of schedule generated by this algorithm is

$$C_{max} = (2k + 2)b_1 + 2\max\{i : 2 \leq i \leq 2k - 1 \quad \wedge \quad b_i = b_1\}$$

so

$$C_{max} - C_{max}^{opt} \leq 4k - 2$$

To summarise

$$C_{max} - C_{max}^{opt} \leq 4k + 2$$

So, the solution generated by this algorithm can be longer than the optimal one by a constant number of time units (the k is fixed) regardless of the size of the instance.

5. Conclusions

In the paper, scheduling of coupled tasks has been considered. General precedence constraints resulted in a NP-hardness of the problem, even for unit processing times and equal gap lengths for all the tasks. The other cases, especially where precedence constraints are chain-like or tree-shaped are still open.

In this paper is also shown the approximation algorithm of complexity $O(c \log c)$ that generates solutions with fixed error comparative to the optimal ones is given for the strict chains precedence constraints and even length of the gap. The algorithm can be also applied for in-forest and out-forest precedence constraints.

REFERENCES

1. Baker K.: *Introduction to Sequencing and Scheduling*, J. Wiley, New York 1974.
2. Błażewicz J., Cellary W., Słowiński R. and Węglarz J.: *Scheduling under Resource Constraints: Deterministic Models*, J. C. Baltzer, Basel 1986.
3. Błażewicz J., Ecker K. H., Pesch E., Schmidt G. and Węglarz J.: *Scheduling Computer and Manufacturing Processes* (2nd edition), Springer, Berlin, New York 2001.
4. Błażewicz J., Ecker K. H., Plateu B., Trystam D.: *Handbook of Parallel and Distributed Processing*, Springer, Berlin, New York 2000.
5. Conway R. W., Maxwell W. L. and Miller L. W.: *Theory of Scheduling*, Addison-Wesley, Reading, Mass. 1967.
6. Gupta J. N. D.: *Single facility scheduling with two operations per job and time-lags*, preprint, 1994.
7. Graham L. R., Lawler E. L., Lenstra J. K., Rinnooy Kan A. H. G.: Optimization and approximation in deterministic sequencing and scheduling theory: a survey, *Ann. Discrete Math. 5*, 1979, 287-326.
8. Farina A., Neri P.: Multitarget interleaved tracking for phased radar array, *IEE Proceedings F* **27**, 1980, 312-318.
9. Kern W., Nawijn W. M.: *Scheduling multi-operations jobs with time lags on a single machine*, preprint, 1991.
10. Lawler E. L., Lenstra J. K., Rinnoy Kan A. H. G., Shmoys D. B: Sequencing and scheduling: algorithms and complexity, in S. C. Graves et al, *Handbooks in Operations Research and Management Science* 4, North-Holland, Amsterdam 1993.
11. Milojevic D. J., Popovic B. M.: Improved algorithm for the interleaving of radar pulses, *IEE Proceedings F* **139**, 1992, 98-104.
12. Orman A. J., Potts C. N.: On the complexity of coupled tasks scheduling, *Discrete Applied Mathematics* **72**, 1997, 141-154.

13. Orman A. J., Potts C. N., Shahani A. K., Moore A. R.: Scheduling for the control of a multifunctional radar system, *European Journal of Operational Research* 90, 1996, 13-25.
14. Pinedo M.: *Scheduling: Theory, Algorithms and Systems*, Prentice Hall, Englewood Cliffs, N. J. 1995.
15. Rinnooy Kan A. H. G.: *Machine Scheduling Problems: Classification, Complexity and Computations*, Martinus Nijhoff, The Hague 1976.
16. Shapiro R. D.: Scheduling coupled tasks, *Naval. Res. Logist. Quart.* 27, 1980, 477-481.
17. Orman A. J., Shahani A. K. and Moore A. R.: Modelling for the control of a complex radar system, *Computers Ops Res.* 25, 1998, 239-249.
18. Yu W.: *The Two-machine Flow Shop Problem with Delays and the One-machine Total Tardiness Problem*, Ph. D. Thesis, Technische Universiteit Eindhoven, 1996.

Recenzent: Prof. dr hab. Józef Grabowski

Streszczenie

W referacie zaprezentowano problem szeregowania zadań, zwanych zadaniami sprzężonymi, składających się z dwóch operacji, przy czym wykonywanie drugiej z nich może zostać rozpoczęte po upływie określonego czasu od momentu zakończenia pierwszej operacji. Problem szeregowania zadań sprzężonych jest często spotykany w praktyce w systemach sterowania urządzeniami radarowymi, gdzie pierwszą operacją jest wysłanie impulsu radarowego, a drugą odbiór powracającego echa.

W referacie skupiono się na problemie szeregowania zadań sprzężonych na jednym procesorze, gdzie czasy wykonywania wszystkich operacji są jednostkowe, długość przerwy pomiędzy operacjami jest stała, a celem jest minimalizacja długości uszeregowania.

Najpierw skupiliśmy się na przypadku ogólnych ograniczeń kolejnościowych i wykazaliśmy, że dla ogólnego grafu ograniczeń kolejnościowych problem szeregowania zadań sprzężonych jest NP-trudny, nawet przy stałej długości przerwy i jednostkowych czasach wykonywania operacji. W dowodzie wykorzystaliśmy jako pośredni etap problem sprawiedliwego kolorowania grafów z częściowo uporządkowanymi wierzchołkami. Najpierw pokazaliśmy pseudowielomianową transformację problemu trójpodziału zbioru do problemu sprawiedliwego kolorowania grafów, a następnie wielomianową transformację problemu kolorowania grafów do danego problemu szeregowania.

W dalszej kolejności przedstawiliśmy szybki algorytm znajdujący przybliżone rozwiązanie naszego problemu szeregowania zadań sprzężonych dla przypadku, gdy graf ograniczeń kolejnościowych ma postać łańcuchów, a długość przerwy pomiędzy operacjami jest parzysta. Dokonaliśmy analizy najgorszego przypadku i wykazaliśmy, że błąd bezwzględny rozwiązania generowanego przez ten algorytm jest nie większy od pewnej stałej niezależnie od wielkości instancji.