

COMPLEXITY RESULTS ON OPEN SHOP SCHEDULING TO MINIMIZE TOTAL COST OF OPERATIONS¹

K Giaro, M Kubale, K Piwakowski

Foundations of Informatics Department, Gdańsk University of Technology, Poland
{giaro, kubale, coni}@eti.pg.gda.pl

Received: 06/11/2001

Accepted in the final form: 15/06/2002

Abstract. In this paper we present a collection of results on the complexity of open shop scheduling problem with the total cost of operations as an optimization criterion. In general, our problem is NP-hard even in a degenerated 1-processor case. Therefore we are interested in finding as many restricted instances as possible for which the problem becomes polynomial. These are, in particular, special cases determined by the length of operations and the structure of associated scheduling graph.

Keywords: Bipartite Graph, Chromatic Sum, Edge Coloring, Open Shop, Polynomial Algorithm, NP-Hard Problem

1 Introduction

Suppose there is a set $J = \{J_1, J_2, \dots, J_n\}$ of n jobs and a set $M = \{M_1, M_2, \dots, M_m\}$ of m processors. Each job J_i consists of m operations $o_{i1}, o_{i2}, \dots, o_{im}$. Any operation o_{ij} may be executed by processor M_j only, and for each i the operations of job J_i can be performed in any order. Each processor can simultaneously execute only one operation and at most one operation of a job can be performed at a time. The processing time p_{ij} of each operation o_{ij} is known and is greater or equal to 0. Operations cannot be interrupted (nonpreemptive model). The completion time of the operation o_{ij} is denoted by C_{ij} and the completion time of a job J_i equals $C_i = \max\{C_{ij} : 1 \leq j \leq m\}$. The classical model of total completion time is to minimize $\sum C_i$. In this paper, however, we assume that $\sum_{i=1}^n \sum_{j=1, p_{ij}>0}^m f_{ij}(C_{ij})$ is the optimization criterion, where the summation is restricted to non-empty operations, i.e. such that $p_{ij} > 0$, and function $f_{ij} : N \rightarrow N$, for $j = 1, \dots, n$ is an arbitrary increasing function which describes the cost $f_{ij}(C_{ij})$ of completing operation o_{ij} in time C_{ij} .

Consider the following example: n patients (jobs) come to the hospital and each of them has to undergo m surgeries (operations) one in each of m operating rooms (processors) and the order of surgeries is immaterial. Until the i -th surgery is done the patient must be connected to a special piece of equipment (monitor) responsible for the i -th body action (at the beginning each patient is immediately connected to m monitors and then, successively, after each surgery one monitor is disconnected). The cost of renting a monitor is proportional to the time of monitoring. In this case the optimal schedule minimizes the total cost of renting equipment.

The problem considered here can be modeled as an edge-weighted bipartite graph $G = (V_1, V_2; E)$ with cost functions on the edges. In this model the vertices of V_1 stand for jobs, the vertices of V_2 stand for processors and there is a one-to-one correspondence between the edge set E and the set

¹A preliminary version of this article appeared in the Proceedings of the 10th Int. Conf. "System-Modelling-Control"

of nonempty operations, i.e. edge $ij \in E$ iff there is a nonzero operation o_{ij} . The weight of edge ij denotes the processing time p_{ij} of the corresponding operation. Note that any coloring of the scheduling graph with intervals of p_{ij} colors on the edges corresponds to a solution of our scheduling problem. Such a solution need not to be optimal in the sense of the number of colors used, because we have to take into account the cost of it.

In general, *Total Cost Open Shop* (TCOS) problem is NP-hard and remains so even if the shop has just one processor. Thus we are interested in finding possibly numerous instances for which the problem becomes polynomial. In this paper we discuss the complexity of TCOS problem in its special cases determined by the length of operations and the cyclicity of scheduling graph. According to this we consider two cases: the case of arbitrary time operations (Section 2) and the case of unitary time operations (Section 3). In the last section we summarize the obtained results in a table of complexity.

2 Arbitrary Time Operations

In this section we do not introduce any restriction on the length of operations, but strong restriction on the structure of G .

2.1 Stars

Theorem 1 *Problem TCOS with only one processor is NP-hard.*

Proof: Consider the problem $1||\Sigma w_i U_i$ of scheduling jobs with deadlines on one processor to minimize the weighted number of late jobs. We have a set of n tasks T_1, \dots, T_n with the following parameters all being natural numbers: processing times p_i , deadlines d_i and weights w_i for each $i = 1, \dots, n$. For a given number k we ask if there is a schedule with $\Sigma w_i U_i \leq k$. The NP-completeness of this problem has been shown in [1]. However, we can regard this question as an instance of the TCOS problem for open shop with one processor and n jobs composed of operations o_{i1} corresponding to tasks T_i , for $i = 1, \dots, n$ (in particular $p_{i1} = p_i$). Let the cost functions be $f_i(t) = t + 2nPw_i T(d_i < t)$, where $T(true/false) = 1/0$ and $P = p_1 + \dots + p_n$. It is easy to see that the answer to the previous question is *true* iff there is a schedule for open shop with the objective function of value at most $nP(2k+1)$. \square

Theorem 1 implies that there is no nontrivial polynomially solvable subclass of TCOS that does not introduce any restriction on the maximal number of operations constituting one job and on the maximal number of operations requiring the same processor. The only exception is 1-processor case where f_{i1} are linear functions. This case is equivalent to problem $1||\Sigma w_i C_i$, which can be solved by sorting the operations in non-decreasing order of p_i/w_i [2].

Note that if we have 1-processor open shop then the corresponding scheduling graph is a star with processor M_1 as the central vertex.

2.2 Paths

Assume that the operations form a path P_{m+n} . For the sake of simplicity we assume that $m+n = s$. In other words, we have a sequence t_1, \dots, t_{s-1} of operations, each with length p_i , the completion time C_i and $f_i(C_i)$ being the cost of C_i , for $i = 1, \dots, s-1$. The aim is to find for each t_i a starting time b_i such that

- (1) $b_j + p_j \leq b_{j+1}$ or $b_j \geq b_{j+1} + p_{j+1}$ ($j = 1, \dots, s-2$) and
- (2) the total completion time is as small as possible.

Without loss of generality we introduce two dummy operations t_0 and t_s with $p_0 = p_s = 0$ for which $b_0 = b_s = 0$ and $f_0 \equiv f_s \equiv 0$.

Given a schedule $B = (b_1, \dots, b_{s-1})$ we denote $d(B) = \sum_{i=1}^{s-1} f_i(C_i)$ and say that

- t_i is *preceded on the left* if $b_i = C_{i-1}$;
- t_i is *preceded on the right* if $b_i = C_{i+1}$;
- t_i is *initial* if $b_i = 0$.

Two initial operations t_i, t_j are said to be adjacent, if there is no operation $t_k, i < k < j$ that is initial.

Lemma 1 *For any optimal schedule and any two adjacent initial operations $t_i, t_j (i < j)$ there is $k, i < k \leq j$ such that*

- i. *all operations t_{i+1}, \dots, t_{k-1} are preceded on the left*
- ii. *all operations t_k, \dots, t_{j-1} are preceded on the right.*

Proof: By assumption none operation from t_{i+1}, \dots, t_{j-1} is initial. Thus each of them is either preceded on the left or preceded on the right. If they are all preceded on the left, the thesis follows with $k = j$. If this is not the case, let $k = \min\{r : i < r < j, t_r \text{ is preceded on the right}\}$. Property (i) is obvious. Property (ii) follows from the fact that for any two operations t_r, t_{r+1} it is impossible that t_r is preceded on the right and simultaneously t_{r+1} is preceded on the left. \square

Note that it is not true that for all indices $i < k \leq j$ there is a schedule fulfilling conditions (1) and (i), (ii) of Lemma 1. It may happen that t_{k-1} and t_k would have to perform simultaneously. This situation takes place if and only if $b_{k-1} < C_k$ and $b_k < C_{k-1}$, where

$$b_{k-1} = \sum_{r=i}^{k-2} p_r, \quad C_{k-1} = \sum_{r=i}^{k-1} p_r, \quad b_k = \sum_{r=k+1}^j p_r, \quad C_k = \sum_{r=k}^j p_r.$$

If $p_i = 0$ then $k = i + 1$, since otherwise t_{i+1} would be initial. Similarly, $p_j = 0$ implies $k = j$. Let us define a predicate P excluding these cases:

$$P(i, j, k) \Leftrightarrow \left(\sum_{r=i}^{k-2} p_r \geq \sum_{r=k}^j p_r \vee \sum_{r=k+1}^j p_r \geq \sum_{r=i}^{k-1} p_r \right) \wedge (p_i \neq 0 \vee k = i + 1) \wedge (p_j \neq 0 \vee k = j)$$

Let

$$U(a, b) = \begin{cases} \sum_{i=a}^b f_i(\sum_{j=a}^i p_j) & \text{if } a \leq b \\ \sum_{i=b}^a f_i(\sum_{j=i}^a p_j) & \text{if } a \geq b \end{cases}$$

The following lemma follows immediately from the above definition of U .

Lemma 2 *We have $U(a, b) = f_a(C_a) + \dots + f_b(C_b)$ if t_a is initial and one of the following holds: $a < b$ and t_{a+1}, \dots, t_b are preceded on the left, or $a > b$ and t_b, \dots, t_{a-1} are preceded on the right.* \square

Let

$$W(a, b) = \begin{cases} \min\{U(a, c-1) + U(b, c) - f_b(p_b) : a < c \leq b \wedge P(a, b, c)\} & \text{for } 0 \leq a < b \leq s \wedge (p_a \neq 0 \vee p_b \neq 0) \\ \infty & \text{otherwise} \end{cases}$$

and let $D = (V, A)$ be a weighted digraph with vertex set $V = \{0, 1, \dots, s\}$ and arc set $A = V \times V$, where arc vw has weight $W(vw)$. For a given set of indices $I = \{0, a_1, \dots, a_k, s\}$, where $a_j - a_i > 1$ for $1 \leq i < j \leq k$, we define a schedule B_I as a (proper) schedule having the set of indices of initial operations equal to I and fulfilling conditions (1) and (2).

Lemma 3 *The length of path $I = (0, a_1, \dots, a_k, s)$, where $a_j - a_i > 1$ for $1 \leq i < j \leq k$, in digraph D is $d(B_I)$.*

Proof: Let $B = (b_1, \dots, b_{s-1})$ be an optimal schedule with the set of indices of initial operations $I = \{0, a_1, \dots, a_k, s\}$. Let $a_0 = 0$, $a_{k+1} = s$. By Lemmas 1 and 2 and the definition of function W we have $\sum_{j=a_i}^{a_{i+1}-1} f_j(C_j) = W(a_i, a_{i+1})$ for any two adjacent initial operations with indices a_i, a_{i+1} , $0 \leq i \leq k$. Thus

$$\sum_{i=0}^k W(a_i, a_{i+1}) = \sum_{i=1}^{s-1} f_i(C_i).$$

□

Let B be a (proper) schedule fulfilling (1) and (2) and let I be the set of indices of initial operations in B . Then $B = B_I$. For any other set C of such indices schedule B_C fulfills (1) and since B fulfills (2) we have $d(B) \leq d(B_C)$, so $d(B) = \min\{d(B_C) : S = \{0, a_1, \dots, a_k, s\}, \text{ where } a_j - a_i > 1 \text{ for } 1 \leq i < j \leq k\}$. Thus the schedule B corresponds to the shortest path from vertex 0 to vertex s in D . This leads to the following

Lemma 4 *For any optimal schedule B the set I of indices of initial operations coincides with the set of vertices of any shortest path from vertex 0 to vertex s in digraph D . Moreover, the total weighted completion time of optimal schedule $d(B)$ is equal to the length of the corresponding path I .* □

Now we are in a position to state

Theorem 2 *The TCOS problem whose scheduling graph is a path of $s - 1$ edges can be solved in time $O(s^2)$.*

Proof: Lemmas 3–5 lead us to the following algorithm for finding an optimal solution.

Step 1. Calculate the values of $U(a, b)$ for $0 \leq a, b \leq s$. This can be achieved in $O(s^2)$ time by the following iterations:

for $a = 0$ **to** s :

$$M(a, a) = p_a$$

$$U(a, a) = f_a(p_a)$$

for $b = a + 1$ **to** s :

$$M(a, b) = M(a, b - 1) + p_b$$

$$U(a, b) = U(a, b - 1) + f_b(M(a, b))$$

$$U(b, a) = U(a, b)$$

Step 2. Calculate the weights $W(a, b)$ for $0 \leq a, b \leq s$ storing in $X(a, b)$ the corresponding value x for which $W(a, b) = U(a, x - 1) + U(b, x)$. By a similar double iteration loop, as in the Step 1, it can be computed in time $O(s^2)$.

Step 3. Construct digraph D and find the shortest path I from 0 to s in D . Dijkstra's algorithm determines the shortest path in time $O(s^2)$.

Step 4. Construct the solution B as follows.

if $i \in I$ **then** $b_i = 0$.

if $i \notin I$ **then** find two adjacent indices $a, b \in I$ such that $a < i < b$ and set

$$b_i = \begin{cases} \sum_{r=a}^{i-1} p_r & \text{if } i < X(a, b) \\ \sum_{r=i+1}^b p_r & \text{if } i \geq X(a, b) \end{cases}$$

It is clear that this step can be implemented in time $O(s^2)$.

□

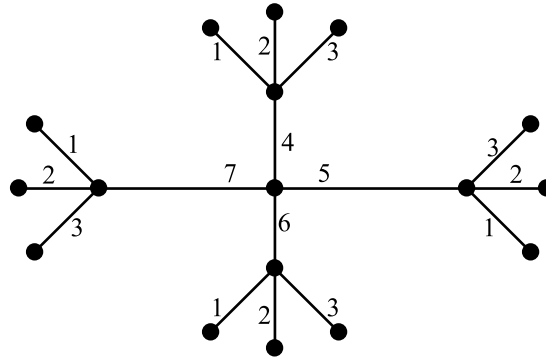


Figure 1: An example of tree.

2.3 Cycles

A straightforward generalization of the algorithm given above leads to the following

Theorem 3 *The TCOS problem whose scheduling graphs is a cycle of length s can be solved in time $O(s^3)$.*

Proof: In an optimal solution at least one of the operations must begin at the time 0. Assuming this for each operation t_k separately we obtain s instances of the above problem for a path of length $s - 1$, with slight modification that dummy tasks t_0 and t_s obtain $b_0 = b_s = 0$ and $p_0 = p_s = p_k$. \square

3 Unitary Time Operations

In this section we assume that all operations have the same execution time. Note that in this case TCOS is equivalent to the edge-cost chromatic sum problem.

Let us assume that the cost of coloring an edge $e \in E(G)$ with $i \in N$ is $f_e(i)$, where $f_e(\cdot)$ is a given increasing function. If $f_e(\cdot)$ does not depend on e , we know that every optimal coloring of a bipartite graph G uses colors not exceeding Δ , where Δ is the maximum vertex degree [3]. This time, however, we cannot formulate a similar theorem. A simple counterexample is a tree shown in Figure 1. Here all pendant edges have $f_e(i) = i$ for $i < k$ and $f_e(i) > 2k^3$ for $i \geq k$, but for central edges we have $f_e(i) = i$.

It is obvious that every optimal coloring uses colors $\{k, \dots, 2k - 1\}$ at the central vertex, so the number of colors used is $2\Delta - 1$. On the other hand, any edge uv has only $\deg(u) + \deg(v) - 2$ incident other edges, where $\deg(u)$ is the degree of vertex u . Therefore, we have

Fact 1 *In any optimal edge coloring c of a graph G :*

- *edge uv has a color not greater than $\deg(u) + \deg(v) - 1$,*
- *c uses colors form $\{1, \dots, 2\Delta - 1\}$ only.*

3.1 Trees

In the following we show a polynomial time algorithm for optimal coloring of a tree G . We will explain only a part of it that computes the optimal color sum, but the extension to a version that computes also an optimal coloring is simple. First, let us orient the tree G in such a way that any vertex $u \in V(G)$ with $\deg(u) = 1$ is the root. For any vertex v let $\text{next}(v)$ be a set of all immediate successors of v in our orientation and for any $w \in \text{next}(v)$ let $T(v, w)$ be a subtree of G induced by v, w and all successors of w (not necessarily immediate). For example $T(u, u') = G$ if $\text{next}(u) = \{u'\}$.

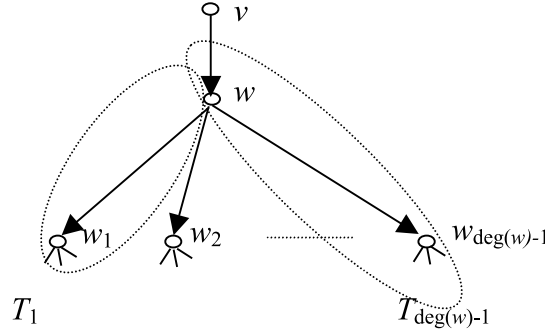


Figure 2: A tree $T(v, w)$ and its subtrees.

Furthermore, let $L = \{1, \dots, 2\Delta - 1\}$ and $\text{Val}(v, w)[i]$ for $i \in L$ be the minimum possible color sum of colorings of a tree $T(v, w)$ with colors from L , where an edge vw is colored with i . We will show, how to obtain $\text{Val}(u, u')[i]$ for all $i \in L$, so the minimal of these numbers is the optimal sum that we are looking for.

The following procedure uses recursion. If $w \in \text{next}(v)$ is a leaf, then $T(v, w)$ is an edge and $\text{Val}(v, w)[i] = f_{vw}(i)$. So let us assume that $\text{next}(w) = \{w_1, \dots, w_{\text{deg}(w)-1}\}$ and that we have already computed vectors $\text{Val}(w, w_l)$ for $l \in \{1, \dots, \text{deg}(w) - 1\}$. How can one obtain $\text{Val}(v, w)[i]$? We create a complete bipartite graph B with partitions $V_1 = \{w_1, \dots, w_{\text{deg}(w)-1}\}$ and $V_2 = L$, in which an edge between w_l and j has weight $\text{Val}(w, w_l)[j]$ (the cost of coloring $T(w, w_l)$ with ww_l colored j). Now the matching of size $\text{deg}(w) - 1$ in B can be treated as legal assignment of colors to edges ww_l . So we find the minimum weighted matching in $B - \{i\}$. If its weight is W , then $\text{Val}(v, w)[i] = W + f_{vw}(i)$. The above facts are sufficient to observe that we can obtain a vector $\text{Val}(u, u')$ in a polynomial time of $O(n\Delta^4)$ using a cubic algorithm for minimum weighted matching. However, we will prove some properties of vectors $\text{Val}(v, w)$ needed to speed this procedure up.

First observe, that in any considered coloring c of $T(v, w)$ with colors from L we have $c(ww_l) \leq \text{deg}(w) + \text{deg}(w_l) - 1$ so we might assume that our weighted graph B has $V_2 = \{1, \dots, \text{deg}(w) + \max_{w' \in \text{next}(w)} \text{deg}(w') - 1\} \subset L$ only. Also for not all $i \in L$ we need a minimum weighted matching algorithm to compute $\text{Val}(v, w)[i]$.

Lemma 5 *For every $j > i \geq \text{deg}(w) + \max_{w' \in \text{next}(w)} \text{deg}(w') - 1$ we have $\text{Val}(v, w)[j] - \text{Val}(v, w)[i] = f_{vw}(j) - f_{vw}(i)$.*

Proof: We consider two colorings of $T(v, w)$, namely: c_1 with $c_1(vw) = i$ and $c_2(vw) = j$, both using colors from L and having color sums equal to $\text{Val}(v, w)[i]$ and $\text{Val}(v, w)[j]$, respectively. Let T be a tree $T = T(w, w_1) \cup \dots \cup T(w, w_{\text{deg}(w)-1})$. The reader may check that $\text{Val}(v, w)[j] - \text{Val}(v, w)[i] - (f_{vw}(j) - f_{vw}(i)) = \sum_{e \in E(T)} (c_2(e) - c_1(e))$. Let us assume that $\sum_{e \in E(T)} c_2(e) < \sum_{e \in E(T)} c_1(e)$. We have $c_2(ww_l) \leq \text{deg}(w) + \text{deg}(w_l) - 2$, so we can construct a coloring c with $c(vw) = i$ by taking $c(e) = c_2(e)$ for all $e \in E(T)$. This coloring has the sum equal to $f_{vw}(i) + \sum_{e \in E(T)} c_2(e) < \text{Val}(v, w)[i]$, a contradiction. Similarly, $\sum_{e \in E(T)} c_2(e) > \sum_{e \in E(T)} c_1(e)$ is impossible. \square

Lemma 6 [4] *For a given bipartite graph B with nonnegative integer weights on the edges we can find maximum weighted matchings in all subgraphs of B induced by $V(B) - \{v\}$, $v \in V(B)$ in time $O(|V(B)|^{1/2} |E(B)| \log(|V(B)| W_{\max}))$, where W_{\max} is the maximum edge weight.* \square

Table 1: Complexity classification for the TCOS problem.

Graph Classes	<i>TCOS with arbitrary time tasks</i>	<i>TCOS with unitary time tasks</i>
general case	NP-hard	NP-hard
$\deg_M \leq 3, \deg_J \leq 3$	NP-hard	NP-hard
scheduling graph is unicyclic	NP-hard	$O((m+n)^{3.5} \log((m+n)C))$
scheduling graph is acyclic	NP-hard	$O((m+n)^{2.5} \log((m+n)C))$
scheduling graph is a star	NP-hard	$O((m+n)^{2.5} \log((m+n)C))$
scheduling graph is acyclic and $\deg_M \leq 3, \deg_J \leq 3$?	$O(m+n)$
scheduling graph is unicyclic and $\deg_M \leq 2, \deg_J \leq 2$	$O((m+n)^3)$	$O(m+n)$
scheduling graph is acyclic and $\deg_M \leq 2, \deg_J \leq 2$	$O((m+n)^2)$	$O(m+n)$

Using this algorithm we can compute $\text{Val}(v, w)[i]$ for all $i \in L$ from $\text{Val}(w, w_l)$, $l \in \{1, \dots, \deg(w) - 1\}$, performing the number of operations $O((\deg(w) + \max_{w' \in \text{next}(w)} \deg(w') - 1)^{2.5} \log(\Delta C) + \Delta)$ for $C = \max_{e \in E(G)} f_e(2\Delta - 1)$. Therefore the overall time complexity is:

$$\begin{aligned}
O(\sum_{w \in V(G)} ((\deg(w) + \max_{w' \in \text{next}(w)} \deg(w') - 1)^{2.5} \log(\Delta C) + \Delta)) &\leq \\
O(\Delta^{1.5} \log(\Delta C) \sum_{w \in V(G)} (\deg(w) + \max_{w' \in \text{next}(w)} \deg(w')) + |V|\Delta) &\leq \\
O(\Delta^{1.5} |V| \log(\Delta C)). &
\end{aligned}$$

Expressing this result in terms of task scheduling we obtain

Theorem 4 *Problem TCOS whose scheduling graph G is a tree can be solved in time $O((m+n)^{2.5} \log((m+n)C))$, where $C = \max_{e \in E(G)} f_e(2\Delta - 1)$.*

3.2 Unicyclic Graphs

If G is a connected scheduling graph with $m+n$ vertices and the same number of edges, then G is unicyclic. Of course, the only cycle must be even. By cutting any edge vw of the cycle into two pendant edges vv_1 and ww_1 and attaching to v_1 and w_1 two bunches of pendant edges v_1x_i and w_1y_i , where $x_i, y_i, i \in \{1, \dots, 2\Delta - 1\}$, we obtain a series of new graphs G_i which are acyclic. For each i we try to color G_i in such a way that edges vv_1 and ww_1 get color i . Now, all we need is proper assignment of cost functions to new edges. Going along the same lines as in [5] we obtain the following result

Theorem 5 *Problem TCOS whose scheduling graph is unicycle can be solved in time $O((m+n)^{3.5} \log((m+n)C))$, where $C = \max_{e \in E(G)} f_e(2\Delta - 1)$.*

4 Summary

From Section 2 it follows that the TCOS problem, where each job consists of at most two operations and at most two operations have to be executed on each processor, can be solved in time $O((n+m)^3)$.

If we replace two by three in the above sentence then the problem becomes NP-hard even if one assumes that the execution time of each operation is the same. This follows from Theorem 1 of [5], where the authors proved that the edge chromatic sum problem is NP-hard even for bipartite graphs of maximum degree 3. If, however, the scheduling graph is a tree with maximum degree 3 and arbitrary weights on the edges then the problem remains open. Finally, if the scheduling graph is acyclic then the problem can be solved in $O((n + m)^2)$ time.

Let deg_M denote the maximal number of operations executed by one processor and deg_J be the maximal number of operations of a job. Table 1 summarizes the main results presented in the previous sections in terms of deg_M and deg_J . Entries in the table are either “NP-hard” or “ $O(\cdot)$ ” for an upper bound on the complexity of the corresponding subproblem, or “?” for an open subproblem.

References

- [1] RM Karp, “Reducibility Among Combinatorial Problems,” in *Complexity of Computer Computations*, Plenum Press, New York, pp 85–103, 1972.
- [2] WE Smith, “Various Optimizers for Single-Stage Production,” *Naval Res. Logist. Quart.* 3, pp 59–66, 1956.
- [3] J Mitchem, P Morriss, E Schmeichel, “On the Cost Chromatic Number of Outerplanar, Planar and Line Graphs,” *Discussiones Mathematicae — Graph Theory*, vol 17, pp 1–13, 1997.
- [4] M Kao, T Lam, W Sung, H Ting, “All-Cavity Maximum Matchings,” in *Proc. ISAAC'97, Lect. Notes in Comp. Sci.*, vol 1350, pp 364–373, 1997.
- [5] K Giaro, M Kubale, “Edge-Chromatic Sum of Trees and Bounded Cyclicity Graphs,” *Inf. Process. Lett.*, vol 75, pp 65–69, 2000.