

Chromatic Scheduling of 1- and 2-Processor UET Tasks on Dedicated Machines with Availability Constraints*

K. Giaro, M. Kubale

Department of Algorithms and System Modeling,
Gdańsk University of Technology, Poland,
{giaro,kubale}@eti.pg.gda.pl

Abstract. We address a generalization of the classical 1- and 2-processor UET scheduling problem on dedicated machines. In our chromatic model of scheduling machines have non-simultaneous availability times and tasks have arbitrary release times and due dates. Also, the versatility of our approach makes it possible to generalize all known classical criteria of optimality. Under these constraints we show that the problem of optimal scheduling of sparse instances can be solved in polynomial time.

1 Introduction

In recent years, a number of new approaches to the problem of parallel computer systems have been proposed. One of them is scheduling of *multiprocessor* task systems [6]. According to this model any task may require for its processing more than one processor at a time. There are two main classes of problems in multiprocessor task scheduling. In the first class of problems, it is assumed that the *number* of simultaneously required processors is important [2, 15] and a task requires a certain prespecified number of processors. In the second class of multiprocessor scheduling problems, the *set* of simultaneously required processors is assumed to be important [1, 3, 13, 14]. In this case either a certain *fixed set* of processors, or a *family* of processor sets on which the task can be executed is given. This paper is concerned with a fixed set scheduling problem.

Furthermore, we assume that for each task the above-mentioned fixed set is either 1-element or 2-element. In other words, we assume that each task is either uniprocessor (it requires a single dedicated processor) or biprocessor (it requires two dedicated procesors simultaneously). For brevity, we will name uniprocessor tasks as *1-tasks* and biprocessor tasks as *2-tasks*.

Since the problem of scheduling 2-tasks is NP-hard subject to all classical optimality criteria, in this paper we are concerned with a special case that the duration of every task is the same. We will call such tasks *unit execution time* (UET) tasks. Consequently, all data are assumed to be positive integers and

* This research was partially supported by KBN grant 4T11C 04725

deterministic. Later on we will see that such a restriction, although remaining NP-hard instances in general, does allow for polynomial-time algorithms in numerous special cases.

The third important assumption concerns availability constraints. Namely, we assume that the availability of tasks is restricted and, in addition, some machines are available only in certain intervals called *time windows*. Time windows may appear in the case of computer breakdowns or maintenance periods. Moreover, in any multitasking computer system and in hard real-time systems in particular, urgent tasks have high priority and are pre-scheduled in certain time intervals, thus creating multiple time windows of availability. Scheduling in time windows was considered in e.g. [1, 2, 8].

The model of scheduling considered in this paper is justified by various technological and efficiency reasons. For example, in fault-tolerant systems tasks may be duplicated to obtain results surely [7]. Mutual testing of processors needs two processors working in parallel [12]. The same can be said about file transfers, which require two corresponding processors simultaneously: the sender and the receiver [4]. Another example is resource scheduling in a batch manufacturing system, where jobs require two dedicated resources for processing [5].

The remaining of this paper is organized as follows. In Section 2 we set up the problem more formally and model it as a list edge-coloring problem. Section 3 is devoted to the case when there are fewer tasks than processors. We show that our problem can be solved efficiently by a tree coloring technique. In Section 4 we generalize these results to the case when the number of tasks is $O(1)$ greater than the number of processors.

2 Mathematical Model

A collection $J = \{J_1, \dots, J_j, \dots, J_n\}$ of n tasks has to be executed by m identical processors $M_1, \dots, M_i, \dots, M_m \in M$. Each task J_j requires the simultaneous use of a set fix_j of one or two prespecified (unique) processors for its execution but each processor can execute at most one such task at a time. Repetition of tasks is not allowed. Task $J_j, j = 1, \dots, n$ requires processing during a given time p_j . All tasks are independent, nonpreemptable and of the same length. For the sake of simplicity we assume that $p_j = 1$ for $j = 1, \dots, n$. Time is divided into unit-length time slots. Due to availability constraints, each task J_j has a list $L(J_j)$ of time slots during which the task is available for processing. Each slot on list $L(J_j)$ has its own weight (cost), independent of other weights of slots of all the lists. Our aim is to check whether a solution exists, and if the answer is affirmative, we are interested in finding a schedule with the minimum total cost. This criterion of optimality generalizes all known classical criteria: C_{\max} , L_{\max} , T_{\max} , \bar{F} , $\sum w_j C_j$, $\sum w_j L_j$, $\sum w_j T_j$, $\sum w_j U_j$, etc.

Using the three-field notation we can describe our problem as $P|win, p_j = 1, |fix_j| \leq 2|criterion$, where we add a word *win* in the second field since time windows are imposed on tasks rather than machines. The word *criterion* in the third field is used to emphasize the versatility of our approach.

The problem considered here can be modeled as list edge-coloring of a pseudograph $G = (V, E)$. There is a one-to-one correspondence between the vertex set $V = \{v_1, \dots, v_m\}$ and the processor set M as well as the edge set $E = \{e_1, \dots, e_n\}$ and the task set J , in the sense that a loop $e_j = \{v_i\} \in E$ corresponds to 1-task J_j to be executed on M_i , and edge $e_j = \{v_i, v_k\} \in E$ corresponds to 2-task J_j to be executed on machines M_i and M_k . Such a pseudograph will be called a *scheduling graph*. To each edge e of G there is assigned a list $L(e) \subseteq N$ (where N is the set of positive integers) specifying which slots (colors) are available to e . Moreover, to each $e \in E$ there is assigned an arbitrary (not necessary increasing) *cost function* $f_e: L(e) \rightarrow N \cup \{0\}$ specifying the cost of coloring edge e with a particular color $i \in N$. The functions f_e are assumed to be computable in $O(1)$ time. The sum of costs among all edges is the *cost of coloring* of graph G . We will attempt to minimize this parameter. It is easy to see that any solution to the $P|win, p_j = 1, |fix_j| \leq 2|$ - is equivalent to a list edge-coloring of the associated scheduling graph G . Therefore, from here on we will speak interchangeably of colorings and schedules.

Illustration of the above concepts is given in Fig. 1. We show over there an example of problem instance, the corresponding scheduling graph and its optimal coloring as well as Gantt diagram of the optimal solution. As already mentioned, we will consider sparse scheduling graphs, i.e. with $|E| = |V| + O(1)$, since such graphs allow polynomial-time scheduling algorithms. In terms of scheduling this means that we restrict instances to those in which the number of tasks n is $O(1)$ greater than the number of processors m . More precisely, the scheduling graphs that we consider may have two kinds of cycles: short 1-edge cycles (loops), and long cycles of 3 or more edges. In the next section we consider scheduling graphs without long cycles. Section 4 is devoted to scheduling graphs with few long cycles.

3 Scheduling Graphs without Long Cycles

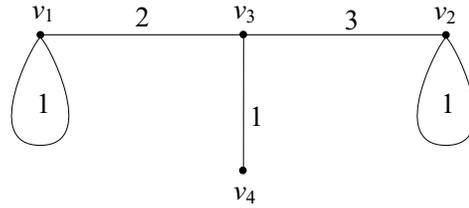
In the following by $\gamma(G)$ we denote the *cyclomatic number* of G , i.e. $|E| - |V| - \lambda + l$, where λ is the number of loops and l is the number of connected components of G . In this section we consider *forests*, i.e. scheduling graphs with $\gamma(G) = 0$. The case of graphs with $\gamma(G) \leq k$ will be considered in the next section. Suppose that a scheduling graph G is in shape of forest, i.e. it is a collection of trees T_1, T_2, \dots . In this case we can color the forest by coloring each T_i separately. For this reason it suffices to consider the problem of coloring the edges of a tree T . Accordingly, let T be a tree on m vertices with $\lambda \leq m$ loops $e_{i_1}, e_{i_2}, \dots, e_{i_\lambda}$. We replace each loop $e_{i_j} = \{v_l\}$ by a pendant edge $e_{i_j} = \{v_l, v_{m+j}\}$, $j = 1, \dots, \lambda$, thus obtaining a new tree with the same number of edges and λ new vertices, but now there are no loops at vertices. Such a transformation is polynomial and does not change the problem, i.e. optimal coloring of the new tree is equivalent to optimal coloring of T .

The authors showed in [10] the following

(a) $M = \{M_1, M_2, M_3, M_4\}$
 $J = \{J_1, J_2, J_3, J_4, J_5\}$

$J_1 \rightarrow M_1$	$L(J_1) = (1,2)$	$f_1(x) = x$
$J_2 \rightarrow M_2$	$L(J_2) = (1,3)$	$f_2(x) = 5x$
$J_3 \rightarrow M_1, M_3$	$L(J_3) = (1,2,3)$	$f_3(x) = 5 x-2 $
$J_4 \rightarrow M_2, M_3$	$L(J_4) = (2,3,4)$	$f_4(x) = x$
$J_5 \rightarrow M_3, M_4$	$L(J_5) = (1,3)$	$f_5(x) = 4x$

(b)



optimal cost = 13

(d)

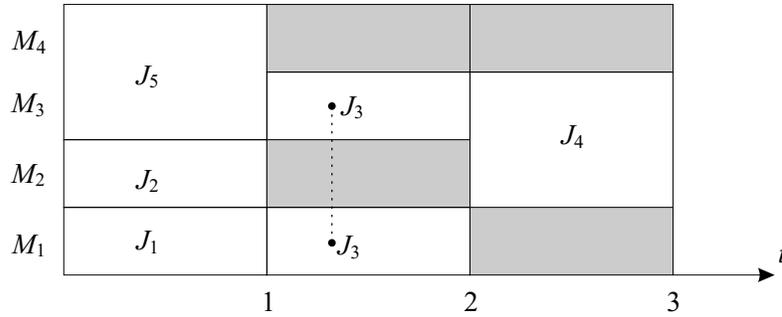


Fig. 1. Example: (a) problem instance; (b) scheduling graph; (c) Gantt diagram.

Lemma 1. For every fixed k the cost edge-coloring problem with increasing cost functions for graphs G with the cyclomatic number at most k can be solved in time $O(m\Delta^{1.5+k} \log(mC))$, where $C = \max_{e \in E(G)} f_e(2\Delta - 1)$ and $\Delta \leq m$ stands for the maximum vertex degree in G . \square

We will show that Lemma 1 can be generalized to list cost coloring of such graphs with arbitrary cost functions. Our approach develops a technique for finding optimal edge sum coloring of trees [9] and uses an improvement useful in reducing its time complexity [16].

Theorem 1. *An optimal list cost coloring of the edges of tree T can be found in time $O(m\Delta^2 \log(mC))$, where $C = \max_{e \in E(G), i \in L(e)} f_e(i)$.*

Proof. First of all, we say that a collection of lists L is *exact* for a graph G if $\forall_{\{u,v\} \in E} |L(\{u,v\})| = \deg(u) + \deg(v) - 1$. It is easy to see that there is a simple polynomial time reduction of a general list-cost coloring problem to its subcase of exact lists L (in this reduction we only have to delete the most expensive colors from lists "too long" and add appropriate new "very expensive" colors to lists "too short"). So in the following we can assume that L is exact.

We shall sketch a procedure for determining the minimum cost of list coloring of a tree T . One can easily extend it to finding the corresponding coloring. For a pair of adjacent vertices v and v' let H stand for the largest subtree of T such that $\{v, v'\}$ belongs to H and v is a leaf. Function $Val(v, v') : L(\{v, v'\}) \rightarrow \{0\} \cup N$ is defined so that $Val(v, v')(i)$ is the minimal cost of $L|_{E(H)}$ -list coloring of the edges of H with the same cost functions as in T , in which edge $\{v, v'\}$ gets color i . If we recursively find the values of $Val(u, u')$ for a leaf u of T , then the requested optimal cost will be equal to $\min_{i \in L(\{u, u'\})} Val(u, u')(i)$.

If H contains only one edge $\{v, v'\}$ then obviously $Val(v, v') = f_{\{v, v'\}}$. So suppose that in the succeeding step of the procedure we have a subtree H as shown in Fig. 2 and the functions $Val(v', w_l), l = 1, \dots, k$ are already known.

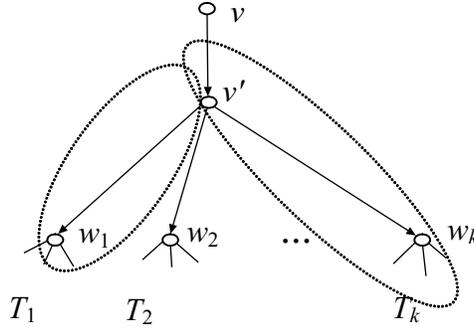


Fig. 2. Tree H and its subtrees.

To find the value of $Val(v, v')$ we construct a bipartite graph $K^{v, v'}$ whose vertices in the first partition are w_1, \dots, w_k and the vertices in the second partition are colors $L(\{w_1, v'\}) \cup \dots \cup L(\{w_k, v'\})$. There is an edge $\{w_l, x\}$ in $K^{v, v'}$ if and only if $x \in L(\{w_l, v'\})$ and then we put the weight $Val(v', w_l)(x)$ on it. In that case $Val(v, v')(j)$ is equal to the weight of the lightest k -edge matching in subgraph $K^{v, v'} - \{j\}$ (i.e. the cost of coloring of T_1, \dots, T_k) plus $f_{\{v, v'\}}(j)$, since edge $\{v, v'\}$ has to be given color j .

Finding the value of $Val(v, v')$ requires prior calculating the weights of k -edge matchings in all graphs $K^{v, v'} - \{j\}$ for $j \in L(\{v, v'\})$. By using the algorithm

mentioned in [11] for graph $K^{v,v'}$, for which the numbers of vertices and edges are $O(\deg_T(v')\Delta(T))$, we can do this in polynomial time $O(\deg_T(v')^{3/2}\Delta(T)^{3/2}\log(m\Delta(T)C))$. We get the overall complexity of the coloring algorithm by summing up among all vertices v' , which results in $O(m\Delta^2\log(mC))$. \square

Reformulating Theorem 1 in terms of scheduling and introducing the symbol $G = \text{forest}$ to mean that the associated scheduling graph of the system is a forest, we obtain the following easy instance of our basic problem.

Theorem 2. *The $P|win, p_j = 1, |fix_j| \leq 2, G = \text{forest}|criterion$ problem can be solved in polynomial time, where ‘criterion’ stands for any of the following: $C_{\max}, \sum w_j C_j, L_{\max}, \sum w_j L_j, T_{\max}, \sum w_j T_j, \sum w_j U_j$, etc.*

Proof. We will show how to minimize the solution subject to the first two criteria: $C_{\max}, \sum w_j C_j$. Finding an optimal solution with respect to the remaining criteria can be accomplished similarly.

Case 1: criterion = $\sum w_j C_j$. For each edge $e \in E(G)$ we set $f_e(i) = iw_e, i = 1, 2, \dots$. Next we find an optimal coloring of the forest G by repeatedly finding an optimal coloring of its trees.

Case 2: criterion = C_{\max} . For each edge $e \in E(G)$ we set $f_e(i) = i, i \in N$. We find an optimal coloring of the forest G . Let l be the length of this solution. We remove colors $s \geq l$ from all lists $L(e_1), L(e_2), \dots$ and invoke the procedure for optimal list edge-coloring of G once more. If it fails, the previous solution is the best possible. If not, we continue the process of shrinking a schedule, possibly using a binary search to speed up the procedure. \square

4 Scheduling Graphs with Few Long Cycles

In this section we assume that $n - m = O(1)$. As previously, suppose that graph G is loopless and connected, and the cyclomatic number $\gamma(G) \leq k$. Under these assumptions we have the following

Theorem 3. *For every fixed k there is a polynomial-time algorithm of complexity $O(m\Delta^{2+k}\log(mC))$, where $C = \max_{e \in E(G), i \in L(e)} f_e(i)$ for a list cost coloring of the edges of G with the cyclomatic number at most k .*

Proof. Let $G(V, E)$ be a graph with $\gamma(G) = k$, L its exact lists and let $A = \{e_1, \dots, e_k\}$ be a set of the edges whose deletion from G results in a spanning tree. Moreover, let $U = \cup_{e \in A} e$ be the set of all vertices incident with the edges of A . All we need is an $O(m\Delta(G)^2\log(mC))$ -time procedure which for a given $L|_A$ -list coloring $d: A \rightarrow N \cup \{0\}$ of graph (U, A) finds its cost optimal extension to an L -list edge-coloring of G . We will use the algorithm of Theorem 1. On the basis of graph G , the collection of lists L and cost functions we build a tree T , as follows.

Step 1: Delete the edges of A from G .

Step 2: For each deleted edge $\{u, v\} \in A$ introduce two new pendant edges $\{u, u_{new}\}$, $\{v, v_{new}\}$ with 1-element lists of the form: $L^T(\{v, v_{new}\}) = L^T(\{u, u_{new}\}) = \{d(\{u, v\})\}$. Next, define two cost functions for them, namely: one $f_{\{v, v_{new}\}}^T = f_{\{v, u\}}$, and the other $f_{\{u, u_{new}\}}^T = 0$.

Step 3: Leave costs and lists of the remaining edges unchanged.

The optimal cost of list edge-coloring of T and the extension of function d to list cost coloring of G are equal: edges $\{u, u_{new}\}$ and $\{v, v_{new}\}$ correspond to the 'halves' of split edge $\{u, v\}$ and 1-element lists of these pendant edges enforce the validity of the extension of d .

As far as the complexity considerations are concerned note that the number of edge-colorings of graph (U, A) is less than $(2\Delta)^k$, which is $O(\Delta^k)$ since k is fixed. Moreover, $|V(T)| = |V(G)| + 2k$ and $\Delta(T) = \Delta(G)$, which completes the proof. \square

As previously, on the basis of Theorem 3 we get the following easy instance of our scheduling problem.

Theorem 4. *For every fixed k the $P|win, p_j = 1, |fix_j| \leq 2, G = k\text{-cyclic}|criterion$ problem can be solved in polynomial time. \square*

Can the chromatic approach presented in the paper be extended to other instances of the UET multiprocessor tasks scheduling? Consider the same problem but with 1-tasks, 2-tasks and 3-tasks. Now 3-tasks can be modeled by hyperedges consisting of three vertices. Thus our pseudograph becomes a hypergraph with 3 kinds of hyperedges: loops, edges and triangles. Clearly, no two hyperedges having a vertex in common can be colored the same. If, in addition to general sparseness of the system, the number of such triangles is constant, then we may apply the same technique as that used in the proof of Theorem 3. Consequently, we arrive at a polynomial-time algorithm for optimal scheduling of k -processor ($k \leq 3$) UET tasks on dedicated processors with arbitrary availability constraints.

References

1. L. Bianco, J. Błażewicz, P. Dell'Olmo, M. Drozdowski, "Preemptive multiprocessor task scheduling with release times and time windows", *Ann. Oper. Res.* 70 (1997) 43-55.
2. J. Błażewicz, P. Dell'Olmo, M. Drozdowski, P. Mączka, "Scheduling multiprocessor tasks on parallel processors with limited availability", *Euro. J. Oper. Res.* 149 (2003) 377-389.
3. J. Błażewicz, P. Dell'Olmo, M. Drozdowski, M.G. Speranza, "Scheduling multiprocessor tasks on three dedicated processors", *Infor. Process. Lett.* 41 (1992) 275-280; Corrigendum IPL 49 (1994) 269-270.
4. E.G. Coffman, Jr., M.R. Garey, D.S. Johnson, A.S. LaPaugh, "Scheduling file transfers", *SIAM J. Comput.* 14 (1985) 744-780.
5. G. S. Dobson, U.S. Karmarkar, "Simultaneous resource scheduling to minimize weighted flow times", *Oper. Res.* 37 (1989) 592-600.

6. M. Drozdowski, "Scheduling multiprocessor tasks - An overview", *J. Oper. Res.* 94 (1996) 215-230.
7. E.F. Gehringer, D.P. Siewiorek, Z. Segall, "*Parallel Processing: The Cm* Experience*", Digital Press, Bedford (1987).
8. A. Gharbi, M. Haouari, "Optimal parallel machines scheduling with availability constraints", *Disc. Appl. Math.* 148 (2005) 63-87.
9. K. Giaro, M. Kubale, "Edge-chromatic sum of trees and bounded cyclicity graphs", *Infor. Process. Lett.* 75 (2000) 65-69.
10. K. Giaro, M. Kubale, K. Piwakowski, "Complexity results on open shop scheduling to minimize total cost of operations", *Intern. J. Comp. Sys. Sign.* 3 (2002) 84-91.
11. M. Kao, T. Lam, W. Sung, H. Ting, "All-cavity maximum matchings", *Proc. Inf. Syst.* E-87 (2004) 364-373.
12. H. Krawczyk, M. Kubale, "An approximation algorithm for diagnostic test scheduling in multicomputer systems", *IEEE Trans. Comp.* 34 (1985) 869-872.
13. M. Kubale, "The complexity of scheduling independent two-processor tasks on dedicated processors", *Infor. Process. Lett.* 24 (1987) 141-147.
14. M. Kubale, "Preemptive versus nonpreemptive scheduling of biprocessor tasks on dedicated processors", *Euro. J. Oper. Res.* 94 (1996) 242-251.
15. E.L. Lloyd, "Concurrent task systems", *Oper. Res.* 29 (1981) 189-201.
16. X. Zhou, T. Nishizeki, "Algorithms for the cost edge-coloring of trees", *LNCS* 2108 (2001) 288-297.