LIBERTAS MATHEMATICA, VOL XX (2000)

COMPLEXITY OF A CLASS OF TASK SCHEDULING PROBLEMS

TONY WONG, JEAN-LOUIS HOULE, RUXANDRA MIHAELA BOTEZ

Abstract

In this paper we present a graph-theoretic procedure which shows the NP-Completeness of a class of task scheduling problems. This problem-class involves scheduling of real-time tasks with temporal constraints usually encountered in process manufacturing and parallel computing. By adopting a decision-theoretic approach, we show the NP-Completeness using a polynomial reduction procedure that is both insightful and intuitive. Since NP-Completeness implies intractability assuming that P $\neq$ NP, exposing the underlying structure of this problem-class should strengthen research efforts in the area of intelligent scheduling systems

1. INTRODUCTION

Tasks scheduling problems arise whenever there is a need to allocate resources, to establish an ordering of tasks to be performed, and to assign tasks to a set of servers for processing. The ordering may involve program tasks to be run on a cluster of computers [1, 2, 3, 4, 5], or jobs that need to be processed in a manufacturing plant [6, 7]. In this paper, we use the term 'task' to denote a manufacturing process or a program unit. We use the term 'processor' to denote individual machinery or computer.

Optimal scheduling solutions are extremely difficult to obtain, mostly because these problems are NP-Hard. We know that a problem $\mathcal{P}$ belongs to the NP-Complete class if: *i*) there exists a polynomial time algorithm that verifies $\mathcal{P}$; *ii*) we can reduce, in polynomial time, all problem $\mathcal{P}$' to $\mathcal{P}$ with $\mathcal{P}$' $\in$ NP [8]. In practice, this cumbersome procedure is never used. Instead, we can prove that $\mathcal{P}$ belongs to the NP-Complete class by showing $\mathcal{P}$', known to be NP-Complete, is reducible in polynomial time to $\mathcal{P}$. In order words, we would induce NP-Complete membership of a problem by transforming the latter to another problem known to be NP-Complete [11].

Intuitively, a problem $\mathcal{P}$ can be reduced to another problem $\mathcal{P}$' if all instances of $\mathcal{P}$ can easily be transformed into instances of $\mathcal{P}$' where solutions of $\mathcal{P}$' are also solutions to $\mathcal{P}$. Thus, if a problem $\mathcal{P}$ is reduced to another problem $\mathcal{P}$' then $\mathcal{P}$ is no harder to solve than $\mathcal{P}$'.

Definition 1.1 *A problem* $\mathcal{P} = (C, P)$ *is reducible to another problem* $\mathcal{P}' = (C', P')$ *if and only if there exists a function* $f : C \rightarrow C'$ *computable in polynomial time, such that,* $\forall c \in C$, c *verifies property* P *if and only if* $f(c)$ *verifies* $\mathcal{P}'$. *The notation* $\mathcal{P} \leq_t \mathcal{P}'$ *designates the reduction of problem* $\mathcal{P}$ *to a problem* $\mathcal{P}'$ *using a function* f *computable in polynomial time. We call* f *a reduction function and the algorithm that computes* f *a reduction algorithm.*

Definition 1.2 *When* $\mathcal{P} \leq_t \mathcal{P}'$ *and* $\mathcal{P}' \leq_t \mathcal{P}$ *then* $\mathcal{P}$ *and* $\mathcal{P}'$ *are equivalent problems. Furthermore, if* $\mathcal{P} \in$ P *then* $\mathcal{P}' \in$ P *and vice versa.*

Theorem 1.1 *The polynomial reduction* $\leq_t$ *is transitive. If* $\mathcal{P} \leq_t \mathcal{P}'$ *and* $\mathcal{P}' \leq_t \mathcal{P}''$ *then* $\mathcal{P} \leq_t \mathcal{P}''$.

Proof. By definition 1.1, the reduction function f is surjective. The relation $\mathcal{P} \leq_t \mathcal{P}''$ implies the existence of a function f such that for all entries $u \in \mathcal{P} \rightarrow f(u) \in \mathcal{P}'$. So,

if $\forall u \in \mathcal{P} \rightarrow f(u) \in \mathcal{P}'$;

if $\forall u' \in \mathcal{P}' \rightarrow f'(u') \in \mathcal{P}''$;

then $\forall u \in \mathcal{P} \rightarrow f'(f(u)) \in \mathcal{P}''$.

Using the functional composition properties, $f'(f(u))$ is also surjective. |

Definition 1.3 *A problem belongs to the* NP-*Complete class if it satisfies the following conditions:*

(i) $\mathcal{P} \in$ NP

(ii) $\mathcal{P}' \leq_t \mathcal{P}, \forall \mathcal{P}' \in$NP.

Definition 1.4 *A problem is* NP-*Hard if it satisfies condition* (ii) *but not necessarily condition* (i) *of definition* 1.3.

2. METHOD OF PROOF

To satisfy the first condition of definition 1.3, it is necessary to find an algorithm that verifies in polynomial time the solution of $\mathcal{P}$ so that $\mathcal{P} \in$ NP. The second condition consists in showing that $\mathcal{P}$ can be reduced in polynomial time to all problems in NP. The latter condition is rather laborious to satisfy. Instead, it is possible to show NP-Hardness by reducing a problem $\mathcal{P}'$ known to be NP-Complete to our problem $\mathcal{P}$ [8, 13]. Once we proved NP-Hardness we must also proceed to show the existence of an algorithm that verifies the solutions of $\mathcal{P}$ in polynomial time. The method of proof is given by the following steps.

- show that $\mathcal{P} \in$ NP;
- choose a problem $\mathcal{P}'$ known to belong in the NP-Complete class;
- describe the computation of the reduction function f which associates all instances of $\mathcal{P}'$ to $\mathcal{P}$;
- show that f satisfies $u \in \mathcal{P}'$ if and only if $f(u) \in \mathcal{P}$;
- show that the computation time of f is polynomial.

3. PROBLEM MODELING

An undirected graph $G = (S, A)$ is defined by a set vertices S and by a set of edges A. In this paper, we consider only simple graphs. There are neither self-loops nor parallel links between two vertices in simple graphs.

Definition 3.1 *The set of vertices $S' \subseteq S$ of graph $G = (S, A)$ is independent if its elements are mutually disjointed.*

Definition 3.2 *The set of vertices $S' \subseteq S$ of graph $G = (S, A)$ is complete if every pair of vertices in S' is linked by an edge $a \in A$. These edges are called complete edges.*

We are interested in graphs whose vertices can be $PARTITION$ed into subsets $S = S_1 \cup S_2$ where S_1 is the set of independent vertices and S_2 is the set of complete vertices. Note that S_2 forms a subgraph also known as a clique [14]. For our problem modeling, we do not impose constraints on the edges linking vertices in S_1 to vertices in S_2.

Definition 3.3 *A graph is complete if there exists an edge linking each node in S_1 to each node in S_2. Edges linking S_1 and S_2 are called link edges.*

Such a graph can model the scheduling of real-time tasks. The set of independent vertices S1 represent the set of processors, and the tasks set is represented by the complete vertices. Figure 1 shows a graphical representation of the problem model [15].

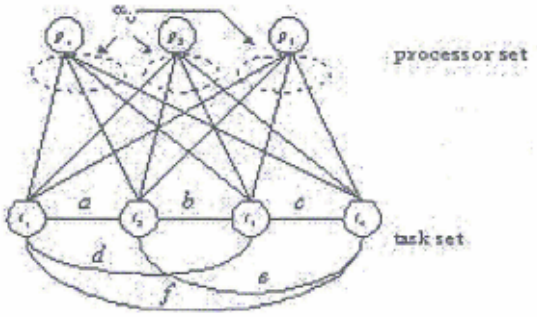

Figure 1. Scheduling problem

A complete edge, in this model, represents the communication cost between two tasks. The communication cost is a value $c \in \aleph$. A link edge represents the execution cost of a task (in S_2) on a processor (in S_1). The execution cost is a value $c \in \aleph^+$. The labels $\{a, b, ..., f\}$ represent the communication cost between tasks. A null value means there is no communication between the tasks. The labels $\{\alpha_{ij}\}$ represent the execution cost of task j on processor i.

In this way, the scheduling problem corresponds to the $PARTITION$ing of a graph G into disjoint cliques. Each clique of graph G consists of a vertex $p \in S_1$ and a subset of vertices $S' \in S_2$. The subset S' represents the tasks assigned to the processor p. Also, the number of disjoint cliques must equal to $|S_1|$ meaning that the number of cliques equal to the number of processors.

As in classical job-shop problems, an optimization criterion is used to measure the efficiency of task-processor assignments. This criterion should be a function of the link edges and complete edges from the resulting cliques. One of the usual metric is the total execution cost C_T given by

$$C_T = C_E + C_C \leq d \tag{1}$$

where C_E, C_C are task execution and task communication costs and d is the task completion deadline.

In our problem model, the execution cost of all tasks on a processor is the sum of link edges in all resulting cliques. The communication cost is given by all complete edges that do not belong to any clique. Suppose an assignment corresponding to a $PARTITION$ of

$G = (S_1 \cup S_2, A)$ into a set of cliques $\{\sigma_1, \sigma_2, ..., \sigma_m\}$ where $m = |S_1|$. Each clique contains at most one vertex $p_i \in S_1$. For each assignment $S = \sigma_1 \cup \sigma_2 \cup ... \cup \sigma_m$ with $p_i \in S_1$ in clique i there is an active link edge (s, p_i) if $s \in \sigma_i$ otherwise it is an inactive link edge. We also have an active complete edge (s_i, s_j) if $s_i, s_j \in \sigma_k$ otherwise it is an inactive complete edge. In this way, the execution cost C_E is the sum of all active link edges and the communication costs is the sum of all inactive complete edges.

In order to minimize the total cost (1), we should maximize the values of active complete edges and minimize the values of active link edges. This can be easily formulated as shown below.

(2) $$\max\{g\} = \max\{c(s_i, s_j)|_{i \neq j} + [\beta - c(s, p_i)]\}$$

where $c(s_i, s_j), i \neq j$ represents the communication cost, $c(s, p_i)$ is the execution cost of a task on processor p_i and β is a large positive constant. Thus, the maximization of (2) is equivalent to the max-clique problem.

4. NP-COMPLETENESS

Using a decision-theoretic approach [10], we describe an instance of the real-time scheduling problem as a 4-tuple and a question as follows.

$RTTR = \{< T, P, W, d >: T = \{t_1, t_2, ..., t_n\}$ is the set of tasks, $P = \{p_1, p_2, ..., p_m\}$ is the set of processors, $W = [w_{ij}]$ is the communication cost matrix and $d > 0$ an integer. Is there a $PARTITION$ing of T onto P such that $C_E + C_C \leq d$ where $C_E = \sum x_{ij}$ is the execution cost of task t_i on p_j and $C_C = \sum w_{ij}$ is the communication cost between t_i and $t_j\}$.

As outlined in section 2, we need a problem belonging to NP-Complete class in order to proceed with our polynomial reduction. The general problem of graph $PARTITION$ing is known to be NP-Complete [8, 9]. The $PARTITION$ problem can be described as a 5-tuple and a question.

$PARTITION = \{< G', c', l', K, J > : G' = (V', E')$ is a simple graph, $c' : V' \rightarrow 1$ is an application giving the weight of vertices in V', $l' : E' \rightarrow \theta$ is an application giving the value of edges in E', K and J are positive integers. Is there a $PARTITION$ of V' into m disjoint subsets $V_1', V_2', ..., V_m'$ such that $\sum_{v \in V_i'} c'(v) \leq K$ for $1 \leq i \leq m$? Furthermore, if $E'' \subseteq E'$ is the set of edges where both ends are in distinct sets V_i' then $\sum_{e \in E''} l'(e) \leq J\}$.

To show that $RTTR \in$NP-Complete, we shall use an intermediary step, mainly to simplify our proof. This intermediary step is another problem concerning the $PARTITION$ of a graph into cliques of maximum weight. We shall name this problem as $MAXCLIQUE$.

$MAXCLIQUE = \{< G, l, B > : G = (V_1 \cup V_2, E)$ a graph with V_1 the set of independent vertices and V_2 the set of complete vertices, $l : E \rightarrow \theta$ an application giving the value of edges in E, $B \geq 0$ an integer. Is there a $PARTITION$ of V into m disjoint cliques σ_i such that $\sum_{e \sigma_i} l(e) \leq B$, for $1 \leq i \leq m$ where $e \in E$ and $m = |V_1|\}$.

In order to show $RTTR \in$ NP-Complete, we should first establish that $PARTITION \leq MAXCLIQUE$ and then $MAXCLIQUE \leq RTTR$.

Theorem 4.1 *The problem* $MAXCLIQUE \in$NP.

Proof. The solution to this problem is a set of disjoint cliques $s_1, s_2, ..., s_m$. To verify $\sum_{e\sigma_i} l(e) \leq B$, for $1 \leq i \leq m$, simply place all edges from resulting cliques in m lists. For each list we sum the edge values and compare the total to B. Assuming a complete graph G with n vertices, there will be $n(n-1)/2$ edges. The summing procedure involves $n^2 - n$ additions. The list creation procedure is a linear function proportional to the number of vertices in G. Thus, we can verify a solution of $MAXCLIQUE$ problem in polynomial time. ■

Theorem 4.2 *The problem $PARTITION$ is reducible to an instance of $MAXCLIQUE$ in polynomial time.*

Proof. We give a polynomial reduction procedure that is capable of reducing $PARTITION$ to $MAXCLIQUE$. Incidentally, we know that $PARITION \in$ NP-Complete for $K \geq 3$ even when$l'(e) = 1, \forall e \in E'$ [8]. The $MAXCLIQUE$ problem $PARTITION$s the set of vertices into two sets V_1 and V_2 where V_1 is the set of independent vertices and V_2 is the set of complete vertices.

A. Install a new set of vertices in the $MAXCLIQUE$ problem whose cardinality is $|V_1| = \begin{pmatrix} m \\ K \end{pmatrix}$. This cardinality guarantees that there is always at least K link edges such that $U \subseteq V_2$, $|U| \leq K$.

B. Assign $V_2 = V'$.

C. Assign to E the set of edges of $MAXCLIQUE$ by the following rules:

$$E = E_1 \cup E_2 \cup E_3 \tag{3}$$

where $E_1 = E'$, $E_2 = \{(u,v)|u,v \in V_2 \text{ and } (u,v) \notin E_1\}$ and $E3 = \{(u,v)|u \in V_1 \text{ and } v \in V_2\}$.

D. Assign to application $l(\cdot)$ of $MAXCLIQUE$ the following values:

$$l(e) = l'(e), \forall e \in E_1, \tag{4a}$$
$$l(e) = 0, \forall e \in E_2, \tag{4b}$$
$$l(e) = a, a \in \{1, \alpha\}, \forall e \in E_3, \tag{4c}$$

where $\alpha \leq -\left(\sum_{e \in E_1} l(e) + m\right)$ is a small negative integer.

Since there is $\begin{pmatrix} m \\ K \end{pmatrix}$ vertices in V_1, there will be $|V_1|$ values given by (4c). We arrange these values in $m \times m$ matrix.

$$M = \begin{pmatrix} 1 & 1 & ... & 1 & \alpha & \alpha & ... & \alpha \\ 1 & ... & 1 & \alpha & \alpha & \alpha & ... & \alpha \\ 1 & ... & \alpha & \alpha & \alpha & \alpha & ... & \alpha \\ \vdots & ... & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \alpha & \alpha & ... & \alpha & 1 & 1 & ... & 1 \end{pmatrix} \tag{5}$$

The goal is to produce a combination of values in M with exactly K value 1 and $m-K$ value of α so that for any subset of complete vertices $V \subseteq V_2$, there exists a vertex u in V_1 such that $l((u,v)) = 1, \forall v \in V, |V| \leq K$.

E. Assign to B the positive integer of $MAXCLIQUE$ the value

$$B = \sum_{e \in E'} l'(e) + m - J \tag{6}$$

Now we show that the reduction procedure produces equivalent solutions for $PARTITION$ and $MAXCLIQUE$ starting with the second part of the question which is part of the definition of $PARTITION$. If we answer YES to the question defining $MAXCLIQUE$, it means that it is possible to obtain m cliques $\sigma_1, \sigma_2, \ldots, \sigma_m$ such that $\sum_{e \sigma_i} l(e) \geq B$ for $1 \leq i \leq m$. Since B is a positive integer and that α is a small negative integer, no clique should contain link edges e with value $l(e) = \alpha$. Consequently, there are m link edges with $l(e) = 1$ in all cliques belonging to E_3. That enables us to write

$$\sum_{e \in \sigma_i \wedge e \notin E_3} l(e) \geq B - m \tag{7}$$

Note that each subgraph obtained from $PARTITION$ is a clique in $MAXCLIQUE$ with one less vertex since we augmented the set V_1 when transforming G'. Also, each subgraph obtained from $PARTITION$ will not contain edges added during the transformation. The edges in E_2 and E_3 are not in the subgraphs of $PARTITION$. Let $\sigma_i = (V(\sigma_i), E(\sigma_i))$ the i^{th} clique obtained from $MAXCLIQUE$. The set of subgraph vertices V_i' of $PARTITION$ are given by $V_i' = V(\sigma_i) \backslash \{v\}$ where $v \in V_1$. That is also true for the set of subgraph edges of $PARTITION$, $E_i' = E(\sigma_i) \backslash (E_2 \cup E_3)$. Finally, the subgraphs obtained from $PARTITION$ are $G_i' = (V_i', E_i')$. Thus, we can rewrite (7) as

$$\sum_{e \in G_i' \wedge e \in E'} l'(e) \geq B - m \tag{8}$$

The sum of values from edges not in the subgraphs is

$$\begin{aligned} \sum_{e \notin G_i'} l'(e) &= \sum_{e \in E'} l'(e) - \sum_{e \in G_i'} l'(e), \\ \sum_{e \notin G_i'} l'(e) &\leq \sum_{e \in E'} l'(e) - (B - m) \end{aligned} \tag{9}$$

Using (6) of step E, we have

$$\sum_{e \notin G_i'} l'(e) \leq \sum_{e \in E'} l'(e) - \left(\sum_{e \in E'} l'(e) + m - J - m \right)$$

which gives

$$\sum_{e \notin G_i'} l'(e) \leq J \tag{10}$$

Since $e \notin G_i'$ are edges not belonging to subgraphs from $PARTITION$, equation (10) is equivalent to the original formulation

$$\sum_{e \in E''} l'(e) \leq J$$

where E'' is the set of edges with both ends in two different V_i'. This solves the second part of the question in $PARTITION$.

We now show the answer to the first part of the question. From (7), we stated that no clique contains link edges e where $l(e) = \alpha$ since $\alpha < 0$. Thus, only link edge with $l(e) = 1$ are included in all cliques. From step D, we have exactly K link edges for each vertex in V_1. Thus, each clique from $MAXCLIQUE$ will contain one vertex in V_1 and at most K vertices in V_2. Since each subgraph vertex from $PARTITION$ is of value 1, K vertices in a subgraph are at most $\sum_{v \in V_i'} c'(v) \leq K$.

The reduction procedure transforms the $PARTITION$ problem into an instance of $MAXCLIQUE$. The transformation steps A to E are polynomial time. Since $PARTITION$ $\in$ NP-Complete and that $MAXCLIQUE \in$ NP (theorem 4.1), we conclude that $MAXCLIQUE$ $\in$ NP-Complete. |

Theorem 4.3 *The problem* $RTTR \in$NP

Proof. The solution to this problem is a set of task assignments A_1, A_2, ..., A_m. Each assignment A_i contains one and only one processor. To verify that A_i satisfies $C_E + C_C \leq d$, we compute $\sum x_{ij}^k + \sum w_{ij}^k$ for $1 \leq k \leq m$ where m is the number of assignments and ascertain that the sum is less than or equal to d. x_{ij}^k is the execution cost of task i onto processor j for the k^{th} assignment. w_{ij}^k is the communication cost between task i and task j for the k^{th} assignment. Assume a system with n tasks and m processors with $m \leq n$. We will need n summations to compute C_E and at most $n(m-1)$ to compute C_C. The verification of $C_E + C_C \leq d$ involves at most nm summations. Thus, we can verify $RTTR$ in polynomial time. |

Theorem 4.4 *The problem* $MAXCLIQUE$ *is reducible to an instance of* $RTTR$ *in polynomial time.*

Proof. We give a polynomial reduction procedure that is capable of reducing $MAXCLIQUE$ to an instance of $RTTR$.

A. Assign the set of complete vertices V_2 of $MAXCLIQUE$ to the task set T of $RTTR$. That is $T = V_2$.

B. Assign the set of independent vertices V_1 of $MAXCLIQUE$ to the processor set P of $RTTR$. That is $P = V_1$.

C. Assign the application $l(e)$, $e = (v_i, v_j)$ with v_i, $v_j \in V_2$ of $MAXCLIQUE$ to the communication cost w_{ij} of RTTR. That is $w_{ij} = l(e)$.

D. Assign the second term of the optimization criterion (2) to the execution cost x_{ij} of $RTTR$. $x_{ij} = \beta - l(e')$ where $e' = (v_i, v_j)$ with $v_i \in V_1$ and $v_j \in V_2$.

E. Assign to d, the task deadline of $RTTR$, the value

$$d = -B + nB + C_C, \tag{11}$$

where B is the positive integer from $MAXCLIQUE$, n is the number of tasks, C_C is the communication cost given by $C_C = \sum_{i,j} w_{ij}$ for $1 \leq i, j \leq n$.

Now we show that the reduction procedure produces equivalent solutions for both problem instances. If we answer YES to the question defining $RTTR$, it means that there exists a scheduling of task set T onto the processor set P such that $C_E + C_C \leq d$. This scheduling gives a $PARTITION$ of cliques for $MAXCLIQUE$ with $\sigma i = \{p_j\} \cup \{t_i | t_i$ are tasks assigned to $p_j\}$.

The total cost of a schedule must not exceed $C_E + C_C \leq d$ where d is the task deadline. Thus, we can rewrite it using the transformation performed in steps A to E.

$$\begin{aligned} \sum_{i,j} x_{ij}^k + \sum_{i,j} w_{ij}^k &\leq d, \\ \sum_{n} \left(\beta - l(e^{'})\right) + \sum_{n} l(e) &\leq d, \\ n\beta - \sum_{n} l(e^{'}) + C_C - \sum_{n} l(e^{''}) &\leq d, \end{aligned} \tag{12}$$

where e are complete edges in different cliques, e' and $e^{''}$ are link and complete edges from the same clique, C_C is the communication cost and n is the number of tasks in the system. By rearranging the last equation in (12) and by defining $\tilde{e}$ as the set of edges in a clique $\tilde{e} = \{(u,v)|u,v \in \sigma_i\}$ representing the total execution and communication costs inside a clique, we have

$$\begin{aligned} -\sum_{n} l(e^{'}) - \sum_{n} l(e^{''}) &\leq d - C_C - n\beta, \\ \sum_{n} l(\tilde{e}) &\geq -d + C_C - n\beta. \end{aligned} \tag{13}$$

Since the value of d has been transformed in step E to $d = -B + n\beta + C_C$, we now have

$$\sum_{n} l(\tilde{e}) \geq -d + C_C - n\beta \Leftrightarrow \sum_{n} l(\tilde{e}) \geq B \tag{14}$$

Equation (14) shows the equivalence of $MAXCLIQUE$ and $RTTR$. The reduction procedure transforms the $MAXCLIQUE$ problem into an instance of $RTTR$. The transformation steps A to E are polynomial times. Since $MAXCLIQUE \in$ NP-Complete (theorem **4.2**) and that $RTTR \in$NP (theorem **4.3**), we conclude that $RTTR \in$NP-Complete. ■

5. CONCLUSION

We have given, in this paper, a graph-theoretic procedure showing the NP-Completeness of a class of scheduling problems. The polynomial reduction is an algorithm that computes the reduction function f in polynomial time, which satisfies $MAXCLIQUE \leq_{'} RTTR$. The reduction was based on a construction associating the task set T and the processor set P of $RTTR$ to the complete set V_2 and the independent set V_1 of $MAXCLIQUE$. We also conferred to w_{ij}, the communication cost of $RTTR$, the application $l : E \rightarrow \theta$ of $MAXCLIQUE$ where $e \in E = (v_i, v_j)$ and $v_i, v_j \in V_2$. In other words, the communication cost is represented by the complete edges. Finally, the execution cost of a task t_i on processor p_j, is assigned to a function $g(e') = \beta - l(e')$ where $e' = (v_i, v_j)$ with $v_i \in V_1$ and $v_j \in V_2$. Thus, the execution cost of a task is, within a constant, the value given by the link edges. Consequently, we transformed the problem $MAXCLIQUE$ into an instance of $RTTR$ by including the optimization criterion given by (2). The NP-Completeness of $RTTR$ implies a possible void in the search for polynomial time solution for this kind of problem [12]. This is a reasonable motivation for further research into heuristic or meta-heuristic scheduling methods.

References

[1] **A. Burns,** "*Scheduling hard real-time systems: A review,*" Software Engineering Journal, vol. 6, no. 3, pp. 116-128, 1990.

[2] **M. L. Dertouzos, A. Mok,** "*Multiprocessor on-line scheduling of hard real-time tasks,*" IEEE Transactions on Software Engineering., vol. 15, no. 12, pp. 1497-1506, Dec. 1989.

[3] **C. Lee, D. Lee, M. Kim,** "*Optimal task assignment in linear array networks,*" IEEE Transactions on Computers, vol. 41, no. 7, pp. 877-880, 1992.

[4] **J. Stankovic,** "*Real-Time computing: a critical enabling technology,*" Technical Report UM-CS-1994-058, University of Massachusetts, 1994.

[5] **J. Xu, D. L. Parnas,** "*On satisfying timing constraints in hard real-time systems,*" IEEE Transactions on Software Engineering, vol. 19, no. 1, pp.70-84, Jan. 1994.

[6] **J. R. Jackson,** "*Scheduling a production line to minimize maximum tardiness,*" Rapport Technique 43, Management Science Research Project, University of California, 1955.

[7] **E. L. Lawler,** "*Optimal sequencing of a single machine subject to precedence constraints,*" Management Science, vol. 19, pp. 544-546, 1973.

[8] **M. R. Garey, D. S. Johnson,** *Computers and Intractability: A Guide to the Theory of NP-Completeness.* San Francisco: W. H. Freeman and Company, 1979.

[9] **S. Evan,** *Graph Algorithms.* Computer Science Press, 1979.

[10] **B. M. Moret,** *The Theory of Computation.* Berkeley: Addison-Wesley, 1998.

[11] **L. G. Valiant, V.V. Vazirani,** "*NP is as easy as detecting unique solutions,*" Proceedings of 17-th Annual ACM Symposium on Theory of Computing, pp. 458-463, 1985.

[12] **I. Pohl,** "*The avoidance of (relative) catastrophe, heuristic competence, genuine dynamic weighting and computational issues in heuristic problem solving,*" Proceedings of the 3-th International Joint Conference on Artificial Intelligence (IJCAI-73), Standford, Carlifornia, pp. 20-23, 1973.

[13] **T. H. Corman, C. E. Leiserson, R. L. Rivest,** *Introduction to Algorithms.* Cambridge: MIT Press, 1993.

[14] **C. Berge,** *Graphes.* Paris: Bordas, 1983.

[15] **T. Wong,** *Répartition automatiques des tâches paralleles: application dans la simulation de réseaux électriques en temps réel,* Ph. D. Thesis, Department of Electrical and Computer Engineering, École Polytechnique de Montréal, 1999.