Approximation algorithm for constrained coupled-tasks scheduling problem

Gilles Simonin *, Benoit Darties †, Jean-Claude König ‡ and Rodolphe Giroudeau ‡

†LIRMM UMR 5506 CNRS - Université de Montpellier II
161 rue Ada 34392 Montpellier Cedex 5, France
Email: {rodolphe.giroudeau, jean-claude.konig}@lirmm.fr

‡LE2I-CNRS UMR 6306 - University of Burgundy
8 Rue Alain Savary 21000 Dijon, France
benoit.darties@u-bourgogne.fr

* Insight Centre for Data Analytics - University College Cork - Ireland
Email: gilles.simonin@insight-centre.org

Abstract—We tackle the makespan minimization coupled-tasks problem in the presence of compatibility constraints. In particular, we focus on stretched coupled-tasks, i.e. coupled-tasks having the same sub-tasks execution time and idle time duration. In such context, we propose some complexity results according to several parameters and we design an efficient polynomial-time approximation algorithm.

I. PRESENTATION OF STRETCHED COUPLED-TASK MODEL

In this article, we consider a extended coupled-tasks scheduling problem (see [11], [12]) in which the tasks are subjected to compatibility constraints. Coupled-tasks, introduced by Shapiro [13], are an easy way to model some data acquisition processes: a coupled-task is composed by two sub-tasks of processing time \( a_i \) and \( b_i \) and whose execution must be separated by a fixed interval time \( L_i \) (called the idle time of the task). Applications from coupled-tasks include, among others, radar detection process on embedded systems: a sensor emits a radio pulse as a first sub-task, and finally listens for an echo. Between these two sub-tasks there is a fixed idle time \( L_i \) which represents the spread of the echo in the medium. We work in a non-preemptive mode: once started, a sub-task cannot be stopped and then continued later. A valid schedule implies here that for any task started at \( t \), the first sub-task is fully executed between \( t \) and \( t + a_i \), and the second between \( t + a_i + L_i \) and \( t + a_i + L_i + b_i \). We note \( A = \{ A_1, \ldots, A_n \} \) the collection of coupled-tasks to be scheduled. This paper focuses on stretched coupled-tasks, i.e. coupled-tasks for what the durations of the first sub-task, the second sub-task and the idle time are equal to a stretch-factor applied to an original task (by example \( (a_i, L_i, b_i) = (1, 1, 1) \)). Formally, a stretched coupled-task \( A_i \) is a task such that \( a_i = L_i = b_i = \alpha(A_i) \), where \( \alpha(A_i) \) is the stretch factor of the task. In the rest of the paper, coupled-tasks are always stretched coupled-tasks, and noted \( A \) when we need to refer to the values \( a_i \), \( b_i \) and \( L_i \), or with a single identifier, i.e. \( x \). Otherwise in such configuration, for two compatible tasks \( A_i \) and \( A_j \) to be scheduled in parallel, one of the following conditions must hold:

1) either \( \alpha(A_i) = \alpha(A_j) \): then the idle time of one task is fully exploited to schedule a sub-task from the other (i.e. \( b_i \) is scheduled during \( L_j \), and \( a_j \) is scheduled during \( L_i \)), and the completion of the two tasks is done without idle time.

2) or \( 3\alpha(A_i) \leq \alpha(A_j) \): then task \( A_i \) is fully executed during the idle time \( L_j \) of \( A_j \). For sake of simplify, we say we pack \( A_i \) into \( A_j \).

In follows, we focus in specific compatibility graph \( G_C \) by considering a 1-stage bipartite graph i.e. a bipartite \( G_c = (X,Y,E_c) \) graph of depth one. We introduce \#X which count up the number of different elements in the set \( X \). Moreover, \( \Delta G_c(X) \) (resp. \( \Delta G_c(Y) \)) represents the maximum degree of the set \( X \) (resp. \( Y \)). In the same way, \( d(Y)G_c \) gives the exact degree of each task in the \( Y \)-set. The results are summarized...
in Table I (ST for stage bipartite). Notice that approximation results are based on results given by [1], [2], [3] and [6].

<table>
<thead>
<tr>
<th>Complexity</th>
<th>Approximation</th>
</tr>
</thead>
<tbody>
<tr>
<td>$O(n^3)$</td>
<td>$FPTAS$ (see [5])</td>
</tr>
</tbody>
</table>

### II. Hardness and Solvable Case

We start by design a polynomial-time algorithm for the scheduling problem in which the maximum degree of incoming arcs on $Y$-tasks is at most two.

**Theorem 1:** The problem of deciding whether an instance of $\#\text{SAT} \leq n \& Y \leq n \& \Delta_{G_c}(Y) = 2 |C_{max}|$ is polynomial.

**Proof:** Let $G_c = (X, Y, E)$ be a 1-stage bipartite compatibility graph. Y-tasks will always be scheduled sequentially. The aim is to fill their idle time with a maximum of tasks of $X$, while the remaining tasks will be executed after the Y-tasks. We just have to minimize the length of the remaining tasks. Note that $\Delta_{G_c}(y) = 2, \forall y \in Y$. The algorithm use three steps:

1. For each task $y \in Y$ such that $3 \times \alpha(x_1) + 3 \times \alpha(x_2) \leq \alpha(y)$ where $x_1$ and $x_2$ are the only two neighbors of $Y$, we add $y$ to the schedule and execute $x_1$ and $x_2$ sequentially during the idle time of $y$. Then we remove $x_1$ and $x_2$ from the instance.

2. Each remaining task $y \in Y$ admits at most two incoming arcs $(x_1, y)$ or $(x_2, y)$. We add a weight $\alpha(x)$ to the arc $(x, y), \forall x \in N(y)$, then perform a maximum weight matching on $G_c$ in order to minimize the length of the remaining tasks of $X$. Thus, the matched coupled-tasks are executed, and these tasks are removed from $G_c$.

3. Then, remaining $X$-tasks are allotted sequentially after the other tasks.

The complexity of an algorithm is $O(n^3)$ using the Hungarian method [9].

**Theorem 2:** The problem of deciding whether an instance of $\#\text{SAT} \leq n |C_{max}|$ is polynomial.

**Proof:** Assume that the arcs are oriented $X \rightarrow Y$. Let $I$ be a instance with $d_G(y_i) \geq k_i + 1, \forall y_i \in Y$. Indeed, if $d_G(y_i) \leq k_i$, the schedule of Y-tasks and the $I^-(y_i)$ are obvious. We assume that $G$ is connected. The scheduling problem is transformed into a maximum flow problem in the following ways:

- We add a source $s$ and a sink $t$. We add arcs $(s, x), \forall x \in X, (y, t), \forall y_i \in Y$ with capacity $c(x, y) = 1$ (resp. $c(y, t) = k_i$ with $k_i = \frac{3\alpha(x)}{\alpha(y_i)}$).

- Finally, we add $c(xy_i) = 1, \forall (x, y_i) \in E$.

The result graph is denoted by $G' = (s, x, Y, E')$.

Now, we show that the flow $f$ admits a maximum flow with $F = \sum_{i=1}^{l} k_i$ with $l = |Y|$. Suppose that the flow $f$ is maximum with value $F = \sum_{i=1}^{l} k_i - 1$. Therefore it exists one arc $(y_i, t)$ unsaturated by the flow $f$. Consequently, we have one arc $(x^*, y_i) \in E, f(x^*, y_i) = 0$. Then it exists one vertex $y_i^* \neq y_i, f(x^*, y_i^*) = 1$ otherwise the task $x^*$ must be packed into $y_i$. Moreover, we assume that it exists one arc $(x, y_i)$ with $f(x, y_i) = 0$ and $f(x, y_i) = 1$.

Since $G_c$ is connected, the chain $x^*, y_1, x_1, \ldots, y_k, x_k \geq 1$ of length even exists. We change the flow value $f(x, y) = z \rightarrow f(x, y) = z$, for all pair of vertices in this chain, with $z \in \{0, 1\}$ and finally we put $f(x^*, y_i) = 1$. Therefore, it exists a new feasible flow $f$ with a value strictly greater than $F$. Since it exists a $s - t$ cut with capacity $\sum_{i=1}^{l} k_i$, thus the maximum value of $f$ is necessarily $\sum_{i=1}^{l} k_i$.

The X-tasks such that $f(x, y_i) = 1$ in $G'$ are packed into the $y_i$-task., otherwise the X-tasks are executed consequently after the Y-tasks.

**Theorem 3:** The problem of deciding whether an instance of $\#\text{SAT} \leq n |C_{max}|$ is N\hspace{-.1em}P-hard.

**Sketch of Proof 1:** The proof is based on a reduction from the 3 Dimensional Matching (see [7]): let $A, B,$ and $C$ be finite, disjoint sets of the same cardinality $n$, and $T$ be a subset of $A \times B \times C$ of cardinality $t$.

The aim is to find $M \subseteq T$ with maximum cardinality, such that for two distinct triples $(u_a, u_b, u_c) \in M$ and $(v_a, v_b, v_c) \in M$ we have $u_a \neq v_a, u_b \neq v_b$, and $u_c \neq v_c$. This problem is well known to be N\hspace{-.1em}P-hard even if the occurrence of each element of $A \cup B \cup C$ is at most 2 in the $T$-subsets [4] (in this case, we have $t = 2n$).

We define a set of tasks $X \cup Y$ and model the compatibility constraint with a graph $G_c = (X, Y, E)$. For each element $x_i \in A \cup B \cup C$, we add an item coupled-task $x_i$ into $X$ with $\alpha(x_i) = 1$.

1. For each 3-element subset $s = (x_a, x_b, x_c) \in T$, we add a box $\alpha(x_b)$ to $Y$ with $\alpha(y) = 9$, and an item coupled-task $x_b$ with $\alpha(x_b) = 2 + \epsilon$.

2. Also add the compatibility arcs $(x_a, y_1), (x_a, y_1), (x_b, y_1), (x_c, y_1))$ to $E$. Clearly we have $l = 2n$ box coupled-tasks (each with an idle time of 9 units) of degree 4 in $G_c$, $t = 2n$ item coupled-tasks $x_b$ with stretch factor $2 + \epsilon$ of degree 1 in $G_c$, and $3n$ item coupled-tasks with stretch factor 1 of degree $2$ in $G_c$. Moreover $G_c$ is a bipartite graph.

The construction implies that for any valid schedule, if 1 item coupled-tasks with stretch factor $2 + \epsilon$ is packed into a box coupled-task, then no other task from $X$ can be packed with it. Suppose we obtain on this instance a schedule of length $27t + 3.3(2 + \epsilon) + 3.3(n - m) = 36n - 3n(1 - \epsilon)$, with

\[\begin{array}{|c|c|c|}
\hline
\text{Topology} & \text{Complexity} & \text{Approximation} \\
\hline
G_c=\text{Star graph} & \text{NP} (see [5]) & \text{FPTAS} (see [5]) \\
G_c=\text{Chain graph} & \text{NP} (see [5]) & \text{FPTAS} (see [5]) \\
\hline
\end{array}\]
m ≤ n, then the idle time of m box coupled-tasks is used to schedule 3n item coupled-tasks with stretch factor 1, the idle time of t − m other box coupled-tasks is used to schedule item coupled-tasks with stretch factor 2+ε, and 3(n−m) remaining tasks from X are scheduled sequentially.

Then one can find a solution to 3 − DM with cardinality m.

III. NEW APPROXIMATION RESULTS

A k-stage bipartite graph is a digraph G = (V0 ∪ ⋯ ∪ Vk, E1 ∪ ⋯ ∪ Ek) where V0…Vk are disjoint vertex sets, and each arc in Ei is from a vertex in Vi to a vertex in Vi+1. The vertex of Vi are said to be at rank i, and the subgraph Gi = (Vi−1 ∪ Vi, Ei) is called the i-th stage of G, and we write G = G1 + ⋯ + Gk.

Figure 1 presents such a k-stage bipartite graph with k = 8.

In the following, we note S the cost of the solution produced by our algorithm, and S* the cost of the optimal one, using in the rest of the paper a *-character exponent to denotes the sets / values related to this optimal solution. We note S[W] (resp. S*[W]) the cost of a valid (resp. optimal) schedule on the instance induced by tasks W ⊆ V with compatibility graph G[W]. We aim to show one can produce a schedule with cost S = S[V] ≤ 41/30 S*[V] = 41/30 S* in a polynomial time.

Compatibilities existing only between tasks of two consecutive ranks and arcs-transitive are not considered: none schedule can Suppose that x is executed during idle of y, then if y is already processed during z-idle time, therefore these allocations lead to a non-feasible schedule, for any x, y, z. In any solution, V-tasks can be 3-partitioned as (1) tasks scheduled alone, noted A(V), (2) tasks with at least one other task scheduled during it idle time noted B(V), and (3) tasks scheduled during the idle time of another one, noted C(V). This notation is extendable to any subset of Vi, i.e. Xi.

Let’s consider an optimal solution S*, also depicted in Figure 2. Each stage Xi is partitioned into \{A*(Xi), B*(Xi), C*(Xi)\} represented with "tirets". "Potatoes" on the figure helps to identify the different set of tasks scheduled in parallel: C*(Xi)-tasks are allotted during the idle time of B*(Xi−1)-tasks. During the idle time of C*(Xi+1) the B*(Xi)-tasks are processed. Lastly, A*(Xi)-tasks are scheduled alone.

**Theorem 4:** The problem 1|αi = ai = Li = bi, k-stage bipartite|Cmax admits a 41/30-approximation algorithm.

**Proof:** Due the lack of place, we prove is omitted. Only the main idea is given. The idea consists to compute a 2/3-approximation on each stage Gi for each i odd thanks to previous results [5], and to merge these parts to get a 41/30-approximated solution. This requires \(\sum_{v \in V_i} \alpha(v) > \sum_{v \in V_i} \alpha(v)\). If not, use the same algorithm with i even instead of odd.

IV. CONCLUSION

We propose some news results on complexity and approximation for constrained coupled-tasks scheduling problem in presence of specific compatibility graph (bipartite). This article supplement the previous works given by [5].

**REFERENCES**


