HAL
open science

Equivalence of two classical list scheduling algorithms for dependent tasks with release dates and due dates on parallel processors

Claire Hanen, Alix Munier-Kordon

To cite this version:

Claire Hanen, Alix Munier-Kordon. Equivalence of two classical list scheduling algorithms for dependent tasks with release dates and due dates on parallel processors. Project Management and Scheduling conference, Apr 2012, Leuwen, Belgium. hal-01272442

HAL Id: hal-01272442

https://hal.science/hal-01272442v1

Submitted on 12 Dec 2016

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Equivalence of two classical list scheduling algorithms for dependent tasks with release dates and due dates on parallel processors

Claire Hanen, Alix Munier-Kordon ,
LIP6, Université P. et M. Curie, France,
Claire.Hanen@lip6.fr, Alix.Munier@lip6.fr

December 28, 2011

Keywords: scheduling, release dates, due dates, list scheduling

1 Introduction

Scheduling problems with release and due dates have been considered for a long time, either in their decision version (is there a schedule meeting all the constraints ?) or in their optimization version, by minimizing the maximum lateness L_{max}. Most of the decision problems are $\mathcal{NP}$-complete once precedence and resource constraints are considered. However, some particular instances have led to polynomial algorithms. Garey and Johnson (1977) solved polynomially $P2|prec, p_i = 1, r_i, d_i|L_{max}$ and the preemptive version $P2|prec, pmtn, r_i, d_i|L_{max}$ by using a particular list scheduling algorithm. This algorithm was extended to get approximation algorithms for the L_{max} criteria with various hypothesis: parallel processors (Hanen and Zinder 2009), preemptive jobs (Hanen and Zinder 2005), communication delays (Hanen and Zinder 2005), typed tasks systems, and unitary resource constraints scheduling problems (Benabid and Hanen 2009, Benabid and Hanen 2010).

This class of scheduling problem was also studied in the context of parallel computing. Processing times are unitary (*i.e.*, $p_i = 1$ for each task i) since each task corresponds to a basic instruction. Precedence delays are introduced to model data transfers between two task: each arc (i, j) is associated to a positive integer value ℓ_{ij} and corresponds to the constraint $t_i + 1 + \ell_{ij} \leq t_j$ where t_i and t_j are starting times of respectively i and j. Leung et. al. (2001) proposed a new list scheduling algorithm that solves polynomially various class of decision instances, among them $P|int.\ order, p_i = 1, monot.\ prec, r_i, d_i|\star$ or $P2|prec, p_i = 1, r_i, d_i, \ell_{ij} \in \{-1, 0\}|\star$ (the $\star$ in the third field of the classical notation corresponds here to the decision version of the scheduling problem). This algorithm was extended to handle monotone interval orders with typed tasks systems (Dupont de Dinechin 2007) or communications delays and dedicated procesors (Munier-Kordon *et. al.* 2011).

Garey and Johnson (*in short* GJ) and Leung, Palem and Pnueli (*in short* LPP) algorithms rely both on an iterative process that modifies the due dates until either a fixed point is reached (due dates are then said to be consistent), or a contradiction is observed. These modified due dates are then used as priorities to build a feasible schedule using a list scheduling algorithm. The motivation of our study was to analyze the relationships between these two iterative process. So we considered a basic scheduling problem where both algorithms can be easily described, namely $P|prec, p_i = 1, r_i, d_i|\star$. We proved that although GJ and LPP algorithms are different, they rely on the same notion of consistency,

i.e, the modified due dates produced by one of the algorithm are consistent with respect to the other algorithm. The consequence is that all the results obtained in the literature for one of these algorithm are also valid for the other one for this basic problem.

2 Problem definition and notations

The scheduling problem tackled by this talk is the optimization problem $P|prec, p_i = 1, r_i, d_i|\ L_{max}$ and its corresponding decision problem $P|prec, p_i = 1, r_i, d_i|\star$. The set of tasks is noted $\mathcal{T}$ and $n = |\mathcal{T}|$. With each task i are given a release time r_i, a due date d_i, and its unit processing time $p_i = 1$.

The cyclic precedence graph is noted $G = (\mathcal{T}, E)$. We denote $i \to j$ if there is a path from i to j in G. $j \in Indep(i)$ if neither $i \to j$ nor $j \to i$. A schedule defines for each task i a completion time C_i. We assume without loss of generality that release times are consistent with precedence constraints, *i.e*, if $i \to j$ then $r_i < r_j$.

The question addressed by the decision problem is: "is there a schedule on m parallel processors satisfying the precedence constraints such that $\forall i \in \mathcal{T}, r_i < C_i \leq d_i$? ". The optimization problem can be stated as follows : "construct a schedule on m parallel processors satisfying the precedence constraints such that $\forall i \in \mathcal{T}, r_i < C_i$ and minimizing the maximum lateness $L_{max} = \max_{i \in \mathcal{T}} C_i - d_i$".

3 The GJ and the LPP consistency

GJ and LPP algorithms modify due dates until a fixed point $D = (D_1, \cdots, D_n) \leq (d_1, \ldots d_n)$ is reached, expressing in both cases necessary conditions for the existence of a feasible schedule. This fixed point rely on two notions of consistency of due dates described in the following.

3.1 GJ consistency

Let consider, for each task $i \in \mathcal{T}$ and each tuple (s, d) such that $r_i \leq s \leq D_i \leq d$, the set

$$S(i, s, d) = \{j \in \mathcal{T}, j \neq i, D_j \leq d,\ and\ (r_j \geq s\ or\ i \to j)\}.$$

Obviously, all tasks from $S(i, s, d)$ should end before d. Now, let suppose that task i ends exactly at its due date D_i. If $D_i = s$, all tasks from $S(i, s, d)$ have to be performed in the time interval $[s, d)$. Otherwise, $D_i > s$ and $S(i, s, d) \cup \{i\}$ have to be performed in the time interval $[s, d)$.

The notion of GJ consistency derives from this intuition. The due dates $D_1, \cdots, D_n$ are *GJ-consistent* if $\forall i \in \mathcal{T},\ r_i < D_i$ and either $|S(i, s, d)| < m(d - s)$ or $|S(i, s, d)| = m(d - s)$ and $s = D_i$.

3.2 LPP consistency

Let $i \in \mathcal{T}$ and $\mathcal{T}_i = Indep(i) \cup \{j \in \mathcal{T}, i \to j\}$. Let also release and due dates vectors r and d and a value $t \in \{r_i + 1, \cdots, d_i\}$. Then, $Existence_i(t, r, d)$ defined as follows is a necessary condition for the existence of a feasible schedule such that i ends at time t.

$Existence_i(t, r, d)$: is there a schedule on m processors of $\mathcal{T}_i \cup \{i\}$ considered as independent tasks meeting release dates r' and due dates d' defined as follows:

1. $d'_i = t$ and $\forall j \in \mathcal{T}_i,\ d'_j = d_j$
2. $r'_i = t - 1$; $\forall j \in Indep(i),\ r'_j = r_j$

3. $\forall j \in \mathcal{T}$ with $i \to j$, $r'_j = max(r_j, t+1)$.

Let us denote by $BS_i(r,d)$ the maximal integer $t \leq d_i$ for which the answer for $Existence_i(t,r,d)$ is yes. The due dates $D = (D_1, \cdots, D_n)$ are *LPP-consistent* if for any task $i \in \mathcal{T}$, $BS_i(r,D) = D_i$. Notice that list scheduling can be used to decide wether $Existence_i(t,r,d)$ is satisfiable.

Lemma 1 *The problem $Existence_i(t,rd)$ can be decided by fixing the starting time of i to be $t-1$ and then scheduling the tasks of $\mathcal{T}_i$ as if they were independent with their modified due-dates and release times using a list scheduling algorithm based on the priorities of the due-dates in increasing order.*

4 Equivalence of due dates consistency and consequences

Our main theorem establishing the equivalence between GJ and LPP consistencies is first presented. Its consequences are then detailed from both an algorithmic and a theoretical point of view.

4.1 Equivalence between GJ and LPP consistencies

The next theorem establishes the equivalence between the two due dates consistencies and constitutes the central result of our talk. The main idea of its proof is to show that any vector D verifying one property also fulfills the other.

Theorem 1 *Let consider an instance of the problem $P|prec, p_i = 1, r_i, d_i|\star$ and let a vector $D_i \leq d_i, \forall i \in \mathcal{T}$. D is GJ-consistent if and only if D is LPP-consistent.*

To prove this theorem, we proceed step by step, by first showing that any LPP-consistent due dates are GJ-consistent.

Lemma 2 *Let D be a vector of LPP-consistent due-dates. Then D is GJ-consistent.*

Proof. Let us assume that D is not GJ-consistent. Then, there exists a task i, and two integers s, d with $r_i \leq s \leq D_i \leq d$, such that the set $S(i,s,d)$ is two large. Assume first that $s < D_i$. then $|S(i,s,d)| \geq m(D_i - s)$. Now, let us notice that $S(i,s,d) \subset \mathcal{T}_i$. Moreover, if we consider the modified due dates and release times for checking $Existence_i(D_i, r, D)$ (whose answer is yes), all successors j of i have a modified release time $r'_j = D_i > s$. Hence, in the schedule build to check $Existence_i(D_i, r, D)$, all tasks from $S(i,s,d)$ are scheduled in the interval $[s,d]$. Moreover, $i \notin S(i,s,d)$ is also scheduled during $[s,d]$. As there are at least $m(d-s)+1$ such tasks, we get a contradiction. ■

Lemma 3 *Let D be a vector of GJ-consistent due-dates. Then D is LPP-consistent.*

Proof. ■ Both GJ and LPP consistencies are sufficient to obtain optimization and approximation results. Thus, a simple outcome of Theorem 1 is that any result obtained for one of these priority list is still valid for the other one. They are listed in the following.

4.2 Complexity of GJ and LPP list scheduling algorithms for solving a decision problem or minimizing the maximum lateness

From a complexity point of view, LPP algorithm is slightly better than GJ one. Indeed, it computes consistent modified due dates in a time complexity $O(n^2 logn\alpha(n) + ne)$ where e is the number of edges of G and $\alpha(n)$ functional inverse of the Ackermann function. GJ algorithm computes consistent modified due dates in $O(n^3 logn)$.

Concerning the optimization problem, several authors (Leung et. al. 2001, Hanen and Zinder 2009) observed that a polynomial-time algorithm minimizing the maximum lateness can be obtained using a due date modification and a list scheduling algorithm as follows:

step 1: Compute by binary search the minimum Δ such that a set of consistent due dates D can be computed from due dates $d + \Delta$;

step 2: Perform a list schedule of the graph G according to the priorities given by the due dates D.

According to Theorem 1, GJ or LPP consistencies may be considered without influence on the quality of the solution obtained. Now, LPP should be preferred for implementations issues as noted before.

4.3 Theoretical consequences of Theorem 1

GJ consistency was exploited by several authors to obtain approximation algorithms for minimizing the maximum lateness, using the scheme described previously. Theorem 2 was proved by Hanen and Zinder (2009) and is true for the two optimization algorithms by Theorem 1.

Theorem 2 *An upper-bound of the maximum lateness L_{max} of the solution obtained using a LPP or GJ consistency-based algorithm for the problem $P|prec, p_i = 1, r_i, d_i|\ L_{max}$ is*

$$L_{max} \leq (2 - \frac{2}{u(m)})L_{max}(\sigma^\star) + (1 - \frac{2}{u(m)}) \max_{i \in N} d_i - (1 - \frac{2}{u(m)})$$

where $L_{max}(\sigma^\star)$ is an optimum schedule and $u(m) = m$ if m is even, $m + 1$ if m is odd.

Notice that as $L_{max}(\sigma^\star)$ may be null; the performance guarantee of an algorithm always involves an additive term depending on the maximum due date.

On the other hand, LPP algorithm was considered to get polynomial-time algorithms minimizing the maximum lateness or the makespan. Theorem 3 is a consequence of Leung et. al. (2001) who proved that the LPP list scheduling algorithm can be used to solve optimally various sub cases of $P|prec, p_i = 1, r_i, d_i|\ L_{max}$ and of Theorem 1.

Theorem 3 *Scheduling problems $P|p_i = 1, interval-order, r_i|L_{max}$, $P|p_i = 1, interval-order, r_i|C_{max}$, $P1|p_i = 1, prec, r_i|L_{max}$, $P|p_i = 1, intree|L_{max}$ and $P|p_i = 1, out-tree, r_i|C_{max}$ are solved polynomially using a GJ or a LPP consistency-based list scheduling algorithm.*

5 Conclusion and further approach

We show that the structural equivalence between the two consistencies leads to a unification of the results proved for each of them. First consequences are a more efficient approximation algorithm with the same worst case performance guarantee to minimize the maximum lateness on parallel processors and some new optimality results for the GJ list scheduling algorithm.

This equivalence should be further investigated for variants of the initial problem: precedence delays, communication delays, typed tasks systems, and preemptive tasks. If, as we conjecture, the equivalence can be proved also in such contexts, other optimality and approximation results will be derived for both algorithms.

We also hope that a careful investigation of the two algorithms may lead to a more efficient algorithm taking the best of both.

One can also think that the due date modification algorithm could be used reversely to modify release times on the reverse graph, and iterate until a fixed point is reached. The question is then to evaluate if a better worst case performance ratio may be achieved.

References

Benabid A. and Hanen C., 2009, "Performance of Garey Johnson algorithm for pipelined type tasks systems", *International Transactions on Operational Research*, Vol. 17, pp. 797-808.

Benabid A. and Hanen C., 2010, "Minimizing lateness for precedence graphs with constant delays on dedicated pipelined processors", *Electronic Notes in Discrete Mathematics*, Vol. 36, pp. 791-798.

Dupont de Dinechin B., 2007, "Scheduling Monotone Interval Orders on Typed Task Systems", *PLANSIG 2007, 26th Worshop of the UK Planning and Scheduling Special Interest Group*, pp. 25–31.

Garey M. R., Johnson D. S, 1977, "Two-processor scheduling with start-time and deadlines", *SIAM Journal on Computing*, Vol. 6, pp. 416-426.

Hanen C. and Zinder Y., 2005, "Scheduling UET-UCT Task Systems under the Out-forest precedence constraints", *Multidisciplinary International Conference on Scheduling: Theory and Applications*, Vol. 2, pp. 445-452.

Hanen C. and Zinder Y., 2009, "The worst-case analysis of the Garey-Johnson Algorithm", *Journal of Scheduling*, Vol. 12, pp. 389-400.

Leung, A., Palem, K. V. and Pnueli, A., 2001, "Scheduling time-constrained instructions on pipelined processors", *ACM Trans. Program. Lang. Syst.*, Vol. 23, pp. 73-103.

Munier-Kordon A., Fadi K., Dupont de Dinechin B. and Finta L., 2011, "Scheduling an interval ordered precedence graph with communication delays and a limited number of processors", *submitted*.