Energetic Reasoning Revisited: Application to Parallel Machine Scheduling

Lotfi Hidri[1], Anis Gharbi[1,2], and Mohamed Haouari[1,3]
(1) Combinatorial Optimization Research Group - ROI,
Ecole Polytechnique de Tunisie, BP 743, 2078, La Marsa, Tunisia.
(2) Department of Industrial Engineering, College of Engineering, King Saud University,
PO Box 800 Riyadh 11421, Saudi Arabia.
(3) Faculty of Business Administration, Bilkent University, 06800 Ankara, Turkey.

Abstract

We consider the problem of minimizing makespan on identical parallel machines subject to release dates and delivery times. We present several new feasibility tests and adjustment techniques that consistently improve the classical energetic reasoning approach. Computational results carried out on a set of hard instances provide strong evidence that the performance of a state-of-the-art exact branch-and-bound algorithm is substantially improved through embedding the proposed enhanced energetic reasoning.

Keywords: scheduling, release dates, due dates, makespan, feasibility and adjustment procedures, energetic reasoning, branch-and-bound.

1. Introduction

We investigate the problem of scheduling identical parallel machines with release dates and delivery times. This problem is denoted by $P|r_j, q_j|C_{\max}$ and is formally described as follows: A set $\mathcal{J}$ of n jobs has to be scheduled on m identical parallel machines (with $n > m \geq 2$). Each job j ($1 \leq j \leq n$) has a processing time p_j, a release date (or head) r_j on which the job becomes available for processing, and a delivery time (or tail) q_j that must elapse between its completion on the machine and its exit from the system. Each machine processes at most one job at one time and each job cannot be processed by more than one machine at one time. We assume that preemption is not allowed and that all machines are available from time zero onwards. The objective is to find a schedule that minimizes the makespan.

This strongly $\mathcal{NP}$-hard problem has been intensely investigated in the machine scheduling literature. Carlier (1987) proposed a first exact branch-and-bound algorithm. Gharbi and Haouari (2002) improved this algorithm by embedding new lower and upper bounds as well as an efficient preprocessing algorithm. Gharbi and Haouari (2005) have proposed a new exact branch-and-bound algorithm that is based on the implementation of a strong max-flow based lower bound

as well as effective dominance rules. During the last decade, several investigations have focused on deriving lower bounding strategies for the $P|r_j, q_j|C_{\max}$. Carlier and Pinson (1998) proposed the so-called Jackson's pseudo-preemptive schedule and showed that it yields a valid lower bound for the $P|r_j, q_j|C_{\max}$. Haouari and Gharbi (2003) improved the preemptive lower bound of Horn (1974) by introducing the semi-preemptive lower bound. Lately, Tercinet et al. (2004) proposed a stronger lower bound that is both based on preemptive scheduling and energetic reasoning. Haouari and Gharbi (2004) proposed a general procedure which aims at improving the quality of a given lower bound by considering subsets of jobs and machines. Recently, Vandevelde et al. (2005) introduced the so-called adjusted lower bound which takes into account in a clever way the non-preemption constraint. Moreover, research efforts have been also devoted to the development of heuristic procedures. Gusfield (1984) and Carlier (1987) investigated the worst-case performance of Jackson's algorithm. This latter is a constructive heuristic which consists in scheduling the available job with the largest tail on the first available machine. Finally, Gharbi and Haouari (2007) developed an effective optimization-based heuristic which consistently provides very near-optimal solutions for instances with up to 2000 jobs and 100 machines.

In this paper, we consider the decision variant of the $P|r_j, q_j|C_{\max}$ which consists in checking the existence of a feasible schedule with makespan equal to a fixed value C. This decision problem is denoted by $P|r_j, d_j|-$, where $d_j = C - q_j$ for all $j \in \mathcal{J}$, and it amounts to checking the existence of a feasible schedule such that each job $j \in \mathcal{J}$ is processed nonpreemptively within its prescribed time window $[r_j, d_j]$, respectively. The main contribution of this work is the development of new feasibility tests and adjustment procedures (or, consistency tests) for this problem. Adjustment techniques aim at tightening the time windows that are associated with each job, respectively. The objective is threefold: detecting infeasibility, improving the values of the lower bounds, and more importantly to enhance the efficiency of exact algorithms. Since the pioneering work of Carlier and Pinson (1989) on the job-shop problem, the use of adjustment techniques has been extensively investigated by many authors (Brucker et al. 1994, Carlier and Pinson 1990, Carlier and Pinson 1994). In addition, Baptiste et al. (1999) and Néron et al. (2001) proposed adjustment techniques for the cumulative problem and the hybrid flow shop problem, respectively. In this paper, we propose several improvements of the so-called *energetic reasoning* which has been initially proposed for cumulative scheduling by Erschler et al. (1991). Our computational experiments show that, in contrast to the *classical* energetic reasoning, the proposed *enhanced* energetic reasoning substantially improves the performance of the best existing branch-and-bound algorithm for the $P|r_j, q_j|C_{\max}$. Also, it is worth emphasizing that although this paper mainly focuses on parallel machine scheduling, the proposed techniques can be extended to more complex scheduling problems including cumulative scheduling and job shop scheduling problems.

The paper is organized as follows. In Section 2, the energetic reasoning is briefly described. Section 3 presents the main contribution: enhanced energetic reasoning-based feasibility and adjustment procedures are detailed. In Section 4, the results of a computational study are reported. Finally, some concluding remarks are provided.

2. The energetic reasoning

Initiated by the paper of Lahrichi (1982), *Energetic Reasoning* (ER) has been originally developed by Erschler et al. (1991) to solve cumulative scheduling problems. The energetic approach which has been formalized and assessed both from a theoretical and an experimental point of view by Baptiste et al. (1999), aims at developing feasibility (or as more commonly referred to, satisfiability) tests and time-bound adjustments to ensure that either a given schedule is infeasible or to derive some necessary conditions that any feasible schedule must satisfy. Moreover, Dorndorf et al. (2000) investigate the use of energetic reasoning within the broader issue of consistency tests (i.e. domain reduction techniques). Since its inception, energetic reasoning has gained popularity and has been used for solving more complex scheduling problems including the hybrid flow shop problem (Néron et al. 2001) and the resource constrained project scheduling problem (Baptiste and Demassey, 2004) as well.

For the sake of completeness, we briefly recall the essentials of energetic reasoning. Given a time interval $[t_1, t_2]$, the energetic reasoning is based on the computation of the part of the jobs that must be processed in any feasible schedule between times t_1 and t_2. The part of job j which needs to be completed within $[t_1, t_2]$ is called its *work* in the time interval $[t_1, t_2]$. To compute this (mandatory) work, the jobs are either left-shifted or right-shifted on their time window $[r_j, d_j]$; i.e., a job either starts at r_j or finishes at d_j. The satisfiability tests and the time-bound adjustments are based on the concepts of the *left-work* and *right-work* of a job. The left-work of a job j over $[t_1, t_2]$, denoted by $W_j^l(t_1, t_2)$, is defined as the part of the job that must be processed between t_1 and t_2 if the job starts at its release date. Symmetrically, the right-work of a job j, denoted by $W_j^r(t_1, t_2)$, is defined as the part of the job that must be processed between t_1 and t_2 if it finishes at its deadline. Thus, the work of a job j over $[t_1, t_2]$, denoted by $W_j(t_1, t_2)$, is equal to the minimum between its left-work and its right-work. Néron et al. (2001) proposed the following formulae for the computation of the left, right and total work of a job j over $[t_1, t_2]$:

$$W_j^l(t_1, t_2) = \min(t_2 - t_1, p_j, \max(0, r_j + p_j - t_1)) \tag{2.1}$$

$$W_j^r(t_1, t_2) = \min(t_2 - t_1, p_j, \max(0, t_2 - d_j + p_j)) \tag{2.2}$$

$$W_j(t_1, t_2) = \min\left(W_j^l(t_1, t_2), W_j^r(t_1, t_2)\right) \tag{2.3}$$

The total work over the time interval $[t_1, t_2]$ is defined by $W(t_1, t_2) = \sum_{j \in \mathcal{J}} W_j(t_1, t_2)$. Clearly, the instance is infeasible if $W(t_1, t_2) > m(t_2 - t_1)$.

Moreover, the time bounds of a job j may be adjusted as follows. Let $s_j(t_1, t_2) = m(t_2 - t_1) - W(t_1, t_2) + W_j(t_1, t_2)$ denote the slack of a job j over $[t_1, t_2]$, i.e. the maximum amount of time that might be allocated to job j during the time interval $[t_1, t_2]$. Let $\bar{r}_j$ and $\bar{d}_j$ denote the adjusted values of r_j and d_j, respectively. Then,

- if $s_j(t_1, t_2) < W_j^l(t_1, t_2)$ then $\bar{r}_j = \max(r_j, t_2 - s_j(t_1, t_2))$
- if $s_j(t_1, t_2) < W_j^r(t_1, t_2)$ then $\bar{d}_j = \min(d_j, t_1 + s_j(t_1, t_2))$

Baptiste et al. (1999) prove that the only relevant values of t_1 and t_2 (with $t_1 < t_2$) that need to be considered are the following ones:

- $t_1 \in \{r_j; j \in \mathcal{J}\} \cup \{d_j; j \in \mathcal{J}\} \cup \{r_j + p_j; j \in \mathcal{J}\}$ and $t_2 \in \{r_j; j \in \mathcal{J}\} \cup \{d_j; j \in \mathcal{J}\} \cup \{d_j - p_j; j \in \mathcal{J}\}$
- $t_1 \in \{r_j; j \in \mathcal{J}\} \cup \{d_j; j \in \mathcal{J}\} \cup \{r_j + p_j; j \in \mathcal{J}\}$ and $t_2 \in \{r_j + d_j - t_1; j \in \mathcal{J}\}$
- $t_2 \in \{r_j; j \in \mathcal{J}\} \cup \{d_j; j \in \mathcal{J}\} \cup \{d_j - p_j; j \in \mathcal{J}\}$ and $t_1 \in \{r_j + d_j - t_2; j \in \mathcal{J}\}$

Recently, Tercinet et al. (2006) have proposed a bin packing-based idea in order to improve the energetic reasoning. It consists in considering the bin packing instance defined by the item weights $W_j(t_1, t_2)$ and the bin capacity $t_2 - t_1$. Clearly, the $P|r_j, d_j|-$ instance is infeasible if a lower bound on the optimal number of bins of the corresponding bin packing instance is larger than m. Therefore, better feasibility tests have been derived by embedding several efficient bin packing lower bounds in the energetic reasoning.

Remark: It is worth noting that formula (2.1) that has been proposed by Néron et al. (2001) for the computation of $W_j^l(t_1, t_2)$ is incorrect in the particular cases where $t_1 < r_j < t_2 < r_j + p_j$ or $r_j \geq t_2$. Indeed, in these cases it yields $W_j^l(t_1, t_2) = \min(t_2 - t_1, p_j)$, whereas the part of job j that must be processed between t_1 and t_2 if it starts at its release date is equal to $t_2 - r_j$ and 0, respectively. A correct formula would be the following

$$W_j^l(t_1, t_2) = \max\{0, \min(r_j + p_j, t_2) - \max(r_j, t_1)\} \tag{2.4}$$

However, the computation of $W_j(t_1, t_2)$ remains valid since, in these two cases, we have $W_j(t_1, t_2) = W_j^r(t_1, t_2)$. Also, the subsequent adjustment of r_j remains valid.
Similarly, the computation of $W_j^r(t_1, t_2)$ over (2.2) is not correct if $d_j - p_j < t_1 < d_j < t_2$ or $d_j \leq t_1$, and could be corrected by setting

$$W_j^r(t_1, t_2) = \max\{0, \min(d_j, t_2) - \max(d_j - p_j, t_1)\} \tag{2.5}$$

Here again, this does not affect the computation of $W_j(t_1, t_2)$ nor the adjustment of d_j. Nevertheless, this does not remain true when alternative energetic reasoning-based adjustment techniques are implemented (as shown in Section 3.4.)

In the sequel, and for the sake of simplicity, $W_j^l(t_1, t_2), W_j^r(t_1, t_2)$, $W_j(t_1, t_2), W(t_1, t_2)$ and $s_j(t_1, t_2)$ will be replaced by W_j^l, W_j^r, W_j, W and s_j, respectively.

3. The energetic reasoning revisited

The goal of this section is to describe new improved feasibility conditions and adjustment procedures that can be performed for a given time interval $[t_1, t_2]$. In this way, only a reduced subset of time intervals may need to be checked. In our experiments, we show that using an $O(n)$-number of time intervals within the enhanced energetic reasoning yields substantially better results than those obtained with the classical energetic reasoning.

3.1. New simple feasibility conditions and adjustment rules

Prior to discussing our enhanced energetic reasoning, we start by describing some new feasibility conditions and define some concepts. A first (trivial) feasibility test is the following.

Condition 0: If there exists a job j satisfying $r_j + p_j > d_j$, then the instance is infeasible.

Now, let Δ denote the time interval $[t_1, t_2]$, and define $J_\Delta = \{j \in J : r_j + p_j > t_1 \text{ and } d_j - p_j < t_2\}$. Note that, for all $j \in J_\Delta$, we have $W_j = \min\{t_2 - t_1, p_j, r_j + p_j - t_1, t_2 - d_j + p_j\}$. Clearly, since J_Δ contains all of the jobs having $W_j > 0$, then $\sum_{j \in \mathcal{J}} W_j(t_1, t_2) = \sum_{j \in J_\Delta} W_j(t_1, t_2)$. Therefore, the energetic reasoning could be restricted to J_Δ.

Let $J_0 = \{j \in J_\Delta : r_j + p_j \geq t_2 \text{ and } d_j - p_j \leq t_1\}$. Clearly, any job $j \in J_0$ has to be processed by one machine during all the interval Δ. We state the following immediate results which may be viewed as a particular case of the energetic reasoning.

Condition 1: If $|J_0| > m$ or $|J_0| = m$ and $J_\Delta \backslash J_0 \neq \emptyset$ then the instance is infeasible.

Rule 0: If $|J_0| = |J_\Delta| = m$, then

- $\bar{r}_j = \min_{j \in J_0}(r_j + p_j)$ for all $j \in \mathcal{J} \backslash J_0$ such that $[r_j, r_j + p_j] \cap (\max_{j \in J_0}(d_j - p_j), \min_{j \in J_0}(r_j + p_j)) \neq \emptyset$
- $\bar{d}_j = \max_{j \in J_0}(d_j - p_j)$ for all $j \in \mathcal{J} \backslash J_0$ such that $[d_j - p_j, d_j] \cap (\max_{j \in J_0}(d_j - p_j), \min_{j \in J_0}(r_j + p_j)) \neq \emptyset$

Clearly, an upper bound on the number of available machines for processing $J_\Delta \backslash J_0$ during the time interval Δ is $m - |J_0|$. For the sake of simplicity, in the sequel, the subset $J_\Delta \backslash J_0$ will be denoted by J, $|J_\Delta \backslash J_0|$ will be denoted by n, and $m - |J_0|$ will be denoted by m.

Now, we introduce the following definitions.

Definition 1: A job $j \in J$ such that $r_j \leq t_1$ is said to be at the *left position* if it is processed during the time interval $[t_1, t_1 + 1]$.

Definition 2: A job $j \in J$ such that $d_j \geq t_2$ is said to be at the *right position* if it is processed during the time interval $[t_2 - 1, t_2]$.

Definition 3: A job $j \in J$ is said to be *inside* if it is processed within the time interval $[t_1 + 1, t_2 - 1]$.

According to the three positions defined above, one can consider the following partition of J:

- The subset of jobs that are at the left position, namely

$$J_L = \{j \in J : r_j + p_j < t_2, \text{ and } d_j - p_j \leq t_1\}$$

- The subset of jobs that are at the right position, namely

$$J_R = \{j \in J : r_j + p_j \geq t_2, \text{ and } d_j - p_j > t_1\}$$

- The subset of jobs that are inside, namely

$$J_I = \{j \in J : t_1 < r_j < d_j < t_2\}$$

- The subset of jobs that are either at the left position or inside, namely

$$J_{LI} = \{j \in J : r_j \leq t_1,\, d_j < t_2, \text{ and } d_j - p_j > t_1\}$$

- The subset of jobs that are either at the right position or inside, namely

$$J_{RI} = \{j \in J : r_j > t_1,\, d_j \geq t_2, \text{ and } r_j + p_j < t_2\}$$

- The subset of jobs that are either at the left or at the right position, namely

$$J_{LR} = \{j \in J : r_j \leq t_1,\, d_j \geq t_2,\, r_j + p_j < t_2,\; d_j - p_j > t_1, \text{ and } p_j \geq t_2 - t_1 - 1\}$$

- The subset of jobs that can be at any position, namely

$$J_{LIR} = \{j \in J : r_j \leq t_1,\, d_j \geq t_2,\, r_j + p_j < t_2,\; d_j - p_j > t_1, \text{ and } p_j < t_2 - t_1 - 1\}$$

The two following feasibility conditions can therefore be stated:

Condition 2: If $\max\{|J_L|, |J_R|\} > m$, then the instance is infeasible.

Condition 3: If $|J_L| + |J_R| + |J_{LR}| > 2m$, then the instance is infeasible.

The following three feasibility conditions take into account in a more effective way the nonpreemption constraint.

Condition 4: If $|J_L| = m$ and $d_{j_0} - p_{j_0} < \min_{j \in J_L}(r_j + p_j)$ for some $j_0 \in J \backslash J_L$, then the instance is infeasible.

Proof. Since the m machines are loaded during the time interval $\left[t_1, \min_{j \in J_L}(r_j + p_j)\right]$ by the jobs of J_L, then j_0 cannot start processing before $\min_{j \in J_L}(r_j + p_j)$. Thus, the minimum finishing time of j_0 is $\min_{j \in J_L}(r_j + p_j) + p_{j_0} > d_{j_0}$, which yields the infeasibility. ■

Condition 5: If $|J_R| = m$ and $r_{j_0} + p_{j_0} > \max_{j \in J_R}(d_j - p_j)$ for some $j_0 \in J \backslash J_R$, then the instance is infeasible.

Condition 6: If $|J_L| + |J_R| + |J_{LR}| = 2m$ and, for some $j_0 \in J_I \cup J_{LI} \cup J_{RI} \cup J_{LIR}$, $p_{j_0} > \max\left\{\max_{j \in J_R}(d_j - p_j), \max_{j \in J_{LR}}(d_j - p_j)\right\} - \min\left\{\min_{j \in J_L}(r_j + p_j), \min_{j \in J_{LR}}(r_j + p_j)\right\}$, then the instance is infeasible.

The proofs of Conditions 5 and 6 are similar to that of Condition 4 and are therefore omitted.

Moreover, the following three simple adjustment rules hold.

Rule 1: If $|J_L| = m$, then

- For all $j \in \mathcal{J}\backslash(J_0 \cup J_L)$ such that $[r_j, r_j + p_j] \cap (\max\limits_{j \in J_0 \cup J_L}(d_j - p_j), \min\limits_{j \in J_L}(r_j + p_j)) \neq \varnothing$, we have $\bar{r}_j = \min\limits_{j \in J_L}(r_j + p_j)$.
- For all $j \in \mathcal{J}\backslash(J_0 \cup J_L)$ such that $[d_j - p_j, d_j] \cap (\max\limits_{j \in J_0 \cup J_L}(d_j - p_j), \min\limits_{j \in J_L}(r_j + p_j)) \neq \varnothing$, we have $\bar{d}_j = \max\limits_{j \in J_0 \cup J_L}(d_j - p_j)$.

Rule 2: If $|J_R| = m$, then

- For all $j \in \mathcal{J}\backslash(J_0 \cup J_R)$ such that $[r_j, r_j + p_j] \cap (\max\limits_{j \in J_R}(d_j - p_j), \min\limits_{j \in J_0 \cup J_R}(r_j + p_j)) \neq \varnothing$, we have $\bar{r}_j = \min\limits_{j \in J_0 \cup J_R}(r_j + p_j)$.
- For all $j \in \mathcal{J}\backslash(J_0 \cup J_R)$ such that $[d_j - p_j, d_j] \cap (\max\limits_{j \in J_R}(d_j - p_j), \min\limits_{j \in J_0 \cup J_R}(r_j + p_j)) \neq \varnothing$, we have $\bar{d}_j = \max\limits_{j \in J_R}(d_j - p_j)$.

Rule 3: If $|J_L| + |J_R| + |J_{LR}| = 2m$, then

- For all $j \in \mathcal{J}\backslash(J_0 \cup J_L \cup J_{LR})$ such that $[r_j, r_j + p_j] \cap (t_1, \min\{\min\limits_{j \in J_L}(r_j + p_j), \min\limits_{j \in J_{LR}}(r_j + p_j)\}) \neq \varnothing$, we have $\bar{r}_j = \min\{\min\limits_{j \in J_L}(r_j + p_j), \min\limits_{j \in J_{LR}}(r_j + p_j)\}$.
- For all $j \in \mathcal{J}\backslash(J_0 \cup J_L \cup J_{LR})$ such that $[d_j - p_j, d_j] \cap (t_1, \min\{\min\limits_{j \in J_L}(r_j + p_j), \min\limits_{j \in J_{LR}}(r_j + p_j)\}) \neq \varnothing$, we have $\bar{d}_j = t_1$.
- For all $j \in \mathcal{J}\backslash(J_0 \cup J_R \cup J_{LR})$ such that $[r_j, r_j + p_j] \cap (\max\{\max\limits_{j \in J_R}(d_j - p_j), \max\limits_{j \in J_{LR}}(d_j - p_j)\}, t_2) \neq \varnothing$, we have $\bar{r}_j = t_2$.
- For all $j \in \mathcal{J}\backslash(J_0 \cup J_R \cup J_{LR})$ such that $[d_j - p_j, d_j] \cap (\max\{\max\limits_{j \in J_R}(d_j - p_j), \max\limits_{j \in J_{LR}}(d_j - p_j)\}, t_2) \neq \varnothing$, we have $\bar{d}_j = \max\{\max\limits_{j \in J_R}(d_j - p_j), \max\limits_{j \in J_{LR}}(d_j - p_j)\}$.

Now, we introduce a feasibility condition which takes into account the maximum number of jobs that can be processed on one machine. Let $\bar{p}_i(J)$ denote the i^{th} smallest processing time in J, and let J_k $(k = 1, ..., n-1)$ denote the largest subset of J such that the smallest k processing times are at least $t_2 - t_1 - 1$, i.e. $\sum\limits_{i=1}^{k} \bar{p}_i(J_k) \geq t_2 - t_1 - 1$. Let $F(j) = \sum\limits_{i=1}^{j} \bar{p}_i(J)$ denote the sum of the j smallest processing times of J $(j = 1, ..., n)$, and $F(0) = 0$. For presorted jobs according to the nondecreasing order of their processing times and a given value of k, the subset J_k can be computed in linear-time in the following way.

For $j = 0, ..., n-k$
If $F(j+k) - F(j) \geq t_2 - t_1 - 1$, *then* $J_k = \{j+1, ..., n\}$ *Stop.*
End (for)

Note that the maximum number of jobs of J_{k-1} that can be processed on a machine is equal to k. Indeed, since the first job ends processing at least at $t_1 + 1$ (by definition of J), and processing the other $k-1$ jobs requires at least $t_2 - t_1 - 1$ units of time, then any set of k jobs of J_{k-1} will finish processing at least at t_2. Which leaves no room for a $(k+1)^{th}$ job. Therefore, the following feasibility condition holds:

Condition 7: If, for some $k = 2, ..., \lfloor n/m \rfloor$, we have $|J_{k-1}| > km$, then the instance is infeasible.

Example 1: Consider the 5 job-2 machine instance defined by $r_j = 2$, $p_j = 2$ and $d_j = 7$ for all $j = 1, ..., 5$. Consider the time interval $\Delta = [3, 6]$. For $k = 2$, we have $J_{k-1} = J$ and $m = 2$. The instance is infeasible since $|J_{k-1}| > km$.

For each time interval, these simple procedures are reiterated until an infeasibility is detected or no adjustment is performed. The whole procedure is outlined by the following algorithm.

Simple_Feasibility_and_Adjustment_Procedure

While (an adjustment occured) Do
Check Condition 0. If an infeasibility is detected, then Stop
Compute J_0 *and Check Condition 1*
If an infeasibility is detected, then Stop. Otherwise, Apply Rule 0
Compute $J_L, J_R, J_I, J_{LI}, J_{RI}, J_{LR}$, *and* J_{LIR}
Check Conditions 2-6
If an infeasibility is detected, then Stop. Otherwise, Apply Rules 1-3
Check Condition 7. If an infeasibility is detected, then Stop
End (while)

Clearly, the partition of J, Conditions 0-6 and Rules 0-3 can be computed in linear time, while the computational complexity of Condition 7 is $O(n \lfloor n/m \rfloor)$ for presorted jobs. Therefore, each iteration of the whole procedure requires $O(n \lfloor n/m \rfloor)$ time.

3.2. Network flows based energetic reasoning

Actually, a job $j \in J$ such that $r_j \leq t_1$ requires an amount of work equal to $r_j + p_j - t_1$ if and only if it is processed within the time interval $[r_j, r_j + p_j]$. That is, there is necessarily one machine loaded with job j during the time interval $[t_1, t_1 + 1]$. Consequently, there are at most m jobs having $r_j \leq t_1$ that would require an amount of work equal to $r_j + p_j - t_1$, respectively. Similarly, there are at most m jobs having $d_j \geq t_2$ that would require an amount of work equal to $t_2 - d_j + p_j$, respectively.

An immediate consequence of this observation is that the minimum total amount of work over Δ could be more accurately estimated in the following way. Since we have $W_j \neq t_2 - t_1$ for all $j \in J$, then $W_j = \min\{r_j + p_j - t_1, t_2 - d_j + p_j, p_j\}$. For all $j \in J$, define

$$x_j = \begin{cases} 1, \text{ if } W_j = r_j + p_j - t_1 \\ 0, \text{ otherwise} \end{cases}$$

$$y_j = \begin{cases} 1, \text{ if } W_j = t_2 - d_j + p_j \\ 0, \text{ otherwise} \end{cases}$$

$$z_j = \begin{cases} 1, \text{ if } W_j = p_j \\ 0, \text{ otherwise.} \end{cases}$$

Hence, a valid lower bound on the total amount of work over Δ is equal to the optimal value of the following 0-1 program.

$$(P1): \text{Minimize } \sum_{j \in J}(r_j + p_j - t_1)x_j + \sum_{j \in J}(t_2 - d_j + p_j)y_j + \sum_{j \in J} p_j z_j \tag{3.1}$$

subject to :

$$x_j + y_j + z_j = 1, \qquad \forall j \in J, \tag{3.2}$$

$$\sum_{j \in J: r_j \leq t_1} x_j \leq m, \tag{3.3}$$

$$\sum_{j \in J: d_j \geq t_2} y_j \leq m, \tag{3.4}$$

$$x_j, y_j, z_j \in \{0, 1\}, \qquad \forall j \in J. \tag{3.5}$$

The objective function (3.1) is to minimize the total amount of work. Constraints (3.2) require that each job should be either at the left, right, or inside position. Constraints (3.3) and (3.4) require that at most m jobs should be assigned at the left and right position, respectively. Finally, Constraints (3.5) state that the decision variables are binary-valued.

Proposition 1: The value of the optimal solution of the relaxed version of (P_1) that is obtained by relaxing Constraints (3.3) and (3.4) corresponds to the total amount of work over Δ computed using the classical energetic reasoning.

Proof. Obvious. ■

It is worth noting that $(P1)$ could be enhanced by setting:

$$x_j = 0, \quad \forall\, j \in J_R \cup J_I \cup J_{RI}$$

$$x_j = 1, \qquad \forall\, j \in J_L$$

$$y_j = 0, \quad \forall\, j \in J_L \cup J_I \cup J_{LI}$$

$$y_j = 1, \qquad \forall\, j \in J_R$$

$$z_j = 0, \quad \forall\, j \in J_L \cup J_R \cup J_{LR}$$

$$z_j = 1, \qquad \forall\, j \in J_I$$

Also, let m_L (resp. m_R) denote the maximum number of available machines for processing the jobs of $J_{LI} \cup J_{LR} \cup J_{LIR}$ (resp. $J_{RI} \cup J_{LR} \cup J_{LIR}$) that may be at the left (resp. the right) position. We have $m_L = \min\{|J_{LI} \cup J_{LR} \cup J_{LIR}|, m - |J_L|\}$ and $m_R = \min\{|J_{RI} \cup J_{LR} \cup J_{LIR}|, m - |J_R|\}$. Thus, we get the following formulation.

$$(P2): \text{Minimize} \sum_{j \in J_{LI} \cup J_{LR} \cup J_{LIR}} W_j^l x_j + \sum_{j \in J_{RI} \cup J_{LR} \cup J_{LIR}} W_j^r y_j + \sum_{j \in J_{LI} \cup J_{RI} \cup J_{LIR}} p_j z_j \tag{3.6}$$

subject to :

$$x_j + z_j = 1, \qquad \forall j \in J_{LI}, \tag{3.7}$$

$$y_j + z_j = 1, \qquad \forall j \in J_{RI}, \tag{3.8}$$

$$x_j + y_j = 1, \qquad \forall j \in J_{LR}, \tag{3.9}$$

$$x_j + y_j + z_j = 1, \qquad \forall j \in J_{LIR}, \tag{3.10}$$

$$\sum_{j \in J_{LI} \cup J_{LR} \cup J_{LIR}} x_j \leq m_L, \tag{3.11}$$

$$\sum_{j \in J_{RI} \cup J_{LR} \cup J_{LIR}} y_j \leq m_R, \tag{3.12}$$

$$x_j \in \{0,1\}, \qquad \forall j \in J_{LI} \cup J_{LR} \cup J_{LIR}, \tag{3.13}$$

$$y_j \in \{0,1\}, \qquad \forall j \in J_{RI} \cup J_{LR} \cup J_{LIR}, \tag{3.14}$$

$$z_j \in \{0,1\}, \qquad \forall j \in J_{LI} \cup J_{RI} \cup J_{LIR}. \tag{3.15}$$

Formulation (3.6)-(3.15) is self-explanatory and is a straightforward enhancement of Formulation (3.1)-(3.5). Let Z_2^* denote the optimal value of $(P2)$. Therefore, an improved value of W_Δ is

$$\tilde{W}_\Delta = Z_2^* + \sum_{j \in J_L} W_j^l + \sum_{j \in J_R} W_j^r + \sum_{j \in J_I} p_j$$

The following feasibility condition clearly holds:

Condition 8: The instance is infeasible if $\tilde{W}_{\Delta} > m(t_2 - t_1)$

Interestingly, the following lemma shows that $(P2)$ can be solved in polynomial-time.

Lemma 1: $(P2)$ is equivalent to a minimum-cost flow problem.

Proof. Consider the network $N = (V, A)$ which is depicted in Figure 1, where the set of nodes V is the union of the following node subsets:

- $\{J_1, J_2, ..., J_{|J_{LI} \cup J_{RI} \cup J_{LR} \cup J_{LIR}|}\}$ (where node J_j represents job $j \in J_{LI} \cup J_{RI} \cup J_{LR} \cup J_{LIR}$)
- $\{L, I, R\}$ representing respectively the left, inside and the right positions
- $\{s, t\}$ where s is the source node, and t is the sink node

The set of arcs A is constructed in the following way:

- For each job node J_i $(i = 1, ..., |J_{LI} \cup J_{RI} \cup J_{LR} \cup J_{LIR}|)$, there is an arc (s, J_i) with unit capacity and zero cost.
- For each job node J_i $(i \in J_{LI} \cup J_{LR} \cup J_{LIR})$, there is an arc (J_i, L) with unit capacity and cost W_i^l.
- For each job node J_i $(i = 1, ..., |J_{LI} \cup J_{RI} \cup J_{LIR}|)$, there is an arc (J_i, I) with unit capacity and cost p_i.
- For each job node J_i $(i \in J_{RI} \cup J_{LR} \cup J_{LIR})$, there is an arc (J_i, R) with unit capacity and cost W_i^r.
- There are three zero-cost arcs $(L, t), (I, t)$ and (R, t) having capacities $m_L, |J_{LI} \cup J_{RI} \cup J_{LIR}|$, and m_R, respectively.

Insert Figure 1 here

One can readily check that there is a strict one-to-one correspondence between feasible $s - t$ flows having a value $v = |J_{LI} \cup J_{RI} \cup J_{LR} \cup J_{LIR}|$ and valid job assignments. Moreover, the cost of such a flow is equal to the total workload of the corresponding job assignment. Hence, the result follows. ■

The resulting network contains $|J_{LI} \cup J_{RI} \cup J_{LR} \cup J_{LIR}| + 2|J_{LI}| + 2|J_{RI}| + 2|J_{LR}| + 3|J_{LIR}| + 3 = O(n)$ arcs and $|J_{LI} \cup J_{RI} \cup J_{LR} \cup J_{LIR}| + 5 = O(n)$ nodes. Therefore, by using the enhanced capacity scaling algorithm for the min-cost flow computation (Orlin, 1988), $\tilde{W}_{\Delta}$ can be computed in $O((n \log n)^2)$ time.

Example 2: Consider the following 5 job-2 machine instance defined by $p_j = 4$ for all $j = 1, ..., 5$ and

$$r_1 = r_2 = r_3 = 1,\ r_4 = r_5 = 2$$

$$d_1 = d_2 = d_3 = 11,\ d_4 = d_5 = 9$$

If we consider the time interval $\Delta = [2, 9]$, then we get

$$W_1^l = W_2^l = W_3^l = 3, W_4^l = W_5^l = 4$$

$$W_1^r = W_2^r = W_3^r = 2, W_4^r = W_5^r = 4$$

$$W_1 = W_2 = W_3 = 2, W_4 = W_5 = 4$$

Using the classical energetic reasoning, we obtain $W_\Delta = 14 = m(t_2 - t_1)$ and no infeasibility is detected. However, the improved energetic reasoning yields $x_1 = 1$, $x_4 = 1$, $y_2 = y_3 = 1$, and $z_5 = 1$, which corresponds to $\tilde{W}_\Delta = 15 > m(t_2 - t_1)$ and therefore permits to declare that the instance is infeasible.

Although, $(P2)$ is solvable in polynomial-time, it might be useful for some applications to have available a faster approximate solution. To that aim, we introduce a relaxed version of $(P2)$ which consists in slightly underestimating $\tilde{W}_\Delta$ while consistently reducing the computational burden from $O((n \log n)^2)$ to $O(n \log m)$. Firstly, remark that:

$$z_j = 1 - x_j \qquad \text{for} \quad j \in J_{LI}$$

$$z_j = 1 - y_j \qquad \text{for} \quad j \in J_{RI}$$

$$z_j = 1 - x_j - y_j \qquad \text{for} \quad j \in J_{LIR}$$

Let $\overline{W}_j^l = p_j - W_j^l$ for $j \in J_{LI} \cup J_{LR} \cup J_{LIR}$ and $\overline{W}_j^r = p_j - W_j^r$ for $j \in J_{RI} \cup J_{LR} \cup J_{LIR}$. Thus, we can eliminate the z variables from $(P2)$ and get the following 0-1 programming formulation.

$$(P3): \text{Maximize} \sum_{j \in J_{LI} \cup J_{LR} \cup J_{LIR}} \overline{W}_j^l x_j + \sum_{j \in J_{RI} \cup J_{LR} \cup J_{LIR}} \overline{W}_j^r y_j \tag{3.16}$$

subject to :

$$x_j + y_j = 1, \qquad \forall j \in J_{LR}, \tag{3.17}$$

$$x_j + y_j \leq 1, \qquad \forall j \in J_{LIR}, \tag{3.18}$$

$$\sum_{j \in J_{LI} \cup J_{LR} \cup J_{LIR}} x_j \leq m_L, \tag{3.19}$$

$$\sum_{j \in J_{RI} \cup J_{LR} \cup J_{LIR}} y_j \leq m_R, \tag{3.20}$$

$$x_j \in \{0, 1\}, \qquad \forall j \in J_{LI} \cup J_{LR} \cup J_{LIR}, \tag{3.21}$$

$$y_j \in \{0, 1\}, \qquad \forall j \in J_{RI} \cup J_{LR} \cup J_{LIR}. \tag{3.22}$$

Let Z_3^* denote the optimal value of $(P3)$. We have

$$Z_2^* = \sum_{j \in J_{LI} \cup J_{RI} \cup J_{LIR} \cup J_{LR}} p_j - Z_3^*$$

Now, let $(P3')$ denote the problem that is obtained by relaxing (3.17) and (3.18). Let $\overline{J}_L$ denote the subset of $J_{LI} \cup J_{LR} \cup J_{LIR}$ that contains the m_L jobs having the largest $\overline{W}_j^l$. Similarly, let $\overline{J}_R$ denote the subset of $J_{RI} \cup J_{LR} \cup J_{LIR}$ that contains the m_R largest $\overline{W}_j^r$. Clearly, an optimal solution of $(P3')$ is obtained by setting

$$x_j = \begin{cases} 1 & \forall\ j \in \overline{J}_L \\ 0 & \text{otherwise.} \end{cases}$$

$$y_j = \begin{cases} 1 & \forall\ j \in \overline{J}_R \\ 0 & \text{otherwise.} \end{cases}$$

Let $\overline{Z}_3^* = \sum\limits_{j \in \overline{J}_L} \overline{W}_j^l + \sum\limits_{j \in \overline{J}_R} \overline{W}_j^r$ denote the optimal objective value of $(P3')$, and

$$W'_\Delta = \sum_{j \in J_{LI} \cup J_{RI} \cup J_{LIR} \cup J_{LR}} p_j - \overline{Z}_3^* + \sum_{j \in J_I} p_j + \sum_{j \in J_L} W_j^l + \sum_{j \in J_R} W_j^r$$

Thus, a valid lower bound on $\tilde{W}$ is

$$\overline{W}_\Delta = \max\left\{W_\Delta, W'_\Delta\right\}$$

Obviously, the following feasibility condition holds:

Condition 9: The instance is infeasible if $\overline{W}_\Delta > m(t_2 - t_1)$.

Example 3: Consider the 6 job-2 machine instance defined by $r_j = 2$, $p_j = 3$ and $d_j = 9$ for all $j = 1, ..., 6$, and let $\Delta = [3, 7]$. We have $W_j^l = 2, W_j^r = 1, W_j = 1$ for all $j = 1, ..., 6$. Thus, $\overline{W}_j^l = 1, \overline{W}_j^r = 2$ for all $j = 1, ..., 6$. Then, we have $\overline{Z}_3^* = (1 + 1) + (2 + 2) = 6$. Consequently, we have $W_\Delta = 6, W'_\Delta = 18 - 6 = 12$, and $\overline{W}_\Delta = 12$ (Note that $W_\Delta \leq m(t_2 - t_1) < \overline{W}_\Delta$).

The computation of $\overline{Z}_3^*$ requires sorting m_L jobs of $J_{LI} \cup J_{LR} \cup J_{LIR}$ and m_R jobs of $J_{RI} \cup J_{LR} \cup J_{LIR}$. Therefore, $\overline{W}_\Delta$ can be computed in $O(n \log m)$ time.

3.3. Lifting procedures

Haouari and Gharbi (2004) proposed a so-called *lifting procedure* in order to improve the lower bounds for the $P|r_j, q_j|C_{\max}$.In this section, we show how to use this procedure in order to improve the energetic reasoning. First, we recall the following lemma.

Lemma 2 (Haouari and Gharbi, 2004): In any feasible schedule of a parallel machine problem with n jobs and m machines, there is at least a set of k machines ($1 \leq k \leq m$) which must process

at least $\lambda_k(n)$ jobs, where $\lambda_k(n) = k\lfloor n/m \rfloor + \min(k, n - \lfloor n/m \rfloor m)$.

An interesting consequence of the above lemma is that m energetic feasibility tests can be derived as follows:

Lemma 3: For a given value of k ($k = 1, ..., m$), let $J_{n,k}$ denote the jobset that is obtained by considering the $\lambda_k(n)$ jobs of J with smallest works W_j. The instance is infeasible if $\sum_{j \in J_{n,k}} W_j > k(t_2 - t_1)$.

More interestingly, additional energetic feasibility tests can be derived by considering different subsets of J. Define $J_l \subseteq J$ as the subset of jobs that contains the l jobs of J with largest works, and $J_{l,k} \subseteq J_l$ as the subset of jobs that is obtained by considering the $\lambda_k(l)$ jobs of J_l with smallest works. We get the following result.

Lemma 4: The instance is infeasible if, for a given value of k ($k = 1, ..., m$) and a given value of l ($l = 1, ..., n$), we have $\sum_{j \in J_{l,k}} W_j > k(t_2 - t_1)$.

It has been shown by Haouari and Gharbi (2004) that only the values of l such that $l \equiv k \pmod m$ have to be considered. Which yields a total number of n energetic feasibility tests. Consequently, we have the following feasibility condition.

Condition 10: The instance is infeasible if $\sum_{j \in J_{l,k}} W_j > k(t_2 - t_1)$ for a given value of k ($k = 1, ..., m$) and $l \equiv k \pmod m$.

Example 4: Consider the 4 job-2 machine instance defined in Table 1 and let $\Delta = [4, 12]$.

insert Table 1 here

We have $W_1 = 5$, $W_2 = 4$, $W_3 = 1$ and $W_4 = 5$. For $k = 1$ and $l = 3$, we get $\lambda_k(l) = 2$ and $J_{l,k} = \{1, 2\}$. Therefore, $\sum_{j \in J_{l,k}} W_j = 9 > k(t_2 - t_1) = 8$ and the instance is infeasible. Note that the classical energetic reasoning does not detect any infeasibility.

In the following, we show how additional feasibility tests can be performed by using Lemma 2 together with the computation of $\tilde{W}_\Delta$ and $\overline{W}_\Delta$.

Consider any set of k machines ($k = 1, ..., m$) and let $J_L^k \subseteq J_L$ denote the subset of $\max(0, k - m_L)$ jobs with minimal W_j^l. Clearly, the amount of work required by the jobs of J_L on the k machines is at least equal to $\sum_{j \in J_L^k} W_j^l$. Similarly, the amount of work required by the jobs of J_R on the k machines is at least equal to $\sum_{j \in J_R^k} W_j^r$, where $J_R^k \subseteq J_R$ denotes the subset of $\max(0, k - m_R)$ jobs with minimal W_j^r. Also, the minimal amount of work required by the jobs of J_I is at least equal to $\max(0, \sum_{j \in J_I} p_j - (m - k)(\max_{j \in J_I} d_j - \min_{j \in J_I} r_j))$.

Now, let $n' = n - |J_L \cup J_R \cup J_I|$ and denote by Z_k a lower bound on the minimum value of $\tilde{W}_\Delta$ yielded by $\lambda_k(n')$ jobs on k machines. The value of Z_k is equal to the optimal value of the following generalized version of $(P2)$:

$$(P_k) : \text{Minimize} \sum_{j \in J_{LI} \cup J_{LR} \cup J_{LIR}} W_j^l x_j + \sum_{j \in J_{RI} \cup J_{LR} \cup J_{LIR}} W_j^r y_j + \sum_{j \in J_{LI} \cup J_{RI} \cup J_{LIR}} p_j z_j \tag{3.23}$$

subject to :

$$\begin{aligned}
x_j + z_j &\leq 1, && \forall j \in J_{LI}, && (3.24)\\
y_j + z_j &\leq 1, && \forall j \in J_{RI}, && (3.25)\\
x_j + y_j &\leq 1, && \forall j \in J_{LR}, && (3.26)\\
x_j + y_j + z_j &\leq 1, && \forall j \in J_{LIR}, && (3.27)\\
\sum_{j \in J_{LI} \cup J_{LR} \cup J_{LIR}} x_j &\leq \min(m_L, k), && && (3.28)\\
\sum_{j \in J_{RI} \cup J_{LR} \cup J_{LIR}} y_j &\leq \min(m_R, k), && && (3.29)
\end{aligned}$$

$$\sum_{j \in J_{LI}} (x_j + z_j) + \sum_{j \in J_{RI}} (y_j + z_j) + \sum_{j \in J_{LR}} (x_j + y_j) + \sum_{j \in J_{LIR}} (x_j + y_j + z_j) = \lambda_k(n') \tag{3.30}$$

$$\begin{aligned}
x_j &\in \{0,1\}, && \forall j \in J_{LI} \cup J_{LR} \cup J_{LIR}, && (3.31)\\
y_j &\in \{0,1\}, && \forall j \in J_{RI} \cup J_{LR} \cup J_{LIR}, && (3.32)\\
z_j &\in \{0,1\}, && \forall j \in J_{LI} \cup J_{RI} \cup J_{LIR}. && (3.33)
\end{aligned}$$

(P_k) consists in selecting a subset $J^k \subseteq J \backslash (J_L \cup J_R \cup J_I)$ containing $\lambda_k(n')$ jobs, such that $\tilde{W}_\Delta(J^k)$ is minimized. Constraints (3.24)-(3.27) are generalizations of (3.7)-(3.10), respectively, since a job j may not be selected in J^k. Constraints (3.28) and (3.29) ensure that there are no more than $\min(m_L, k)$ available left positions and no more than $\min(m_R, k)$ available right positions, respectively. Finally, constraint (3.30) enforce the total number of selected jobs has to be equal to $\lambda_k(n')$.

The value of Z_k can be computed using a minor modification of the min-cost flow reformulation of $(P2)$ (see Lemma 1). Indeed, it suffices to set the flow value $v = \lambda_k(n')$ and the capacities of the arcs (L,t) and (R,t) equal to $\min(m_L, k)$ and $\min(m_R, k)$, respectively (see Figure 1). Consequently, a lower bound on the minimum value of $\tilde{W}_\Delta$ computed on k machines is equal to

$$\tilde{W}_\Delta^k = Z_k + \sum_{j \in J_L^k} W_j^l + \sum_{j \in J_R^k} W_j^r + \max(0, \sum_{j \in J_I} p_j - (m-k)(\max_{j \in J_I} d_j - \min_{j \in J_I} r_j)) \tag{3.34}$$

Therefore, we have the following feasibility condition which can be computed in $O(m(n \log n)^2)$ time:

Condition 11: The instance is infeasible if $\tilde{W}_\Delta^k > k(t_2 - t_1)$ for a given value of $k = 1, ..., m-1$.

Moreover, a lower bound $\overline{W}_{\Delta}^{k}$ on the minimum value of $\overline{W}_{\Delta}$ computed on k machines can be computed in the following way: Let (P'_k) denote the problem obtained by relaxing (3.24)-(3.27) in (P_k). Let $\mathcal{L}$ denote the list of all the values of W_j^l ($j \in J_{LI} \cup J_{LR} \cup J_{LIR}$), W_j^r ($j \in J_{RI} \cup J_{LR} \cup J_{LIR}$) and p_j ($j \in J_{LI} \cup J_{RI} \cup J_{LIR}$) sorted in nondecreasing order. The optimal value Z'_k of (P'_k) can be computed by taking the minimal $\lambda_k(n')$ values of $\mathcal{L}$ while selecting no more than $\min(m_L, k)$ values of W_j^l, and no more than $\min(m_R, k)$ values of W_j^r. We have

$$\overline{W}_{\Delta}^{k} = Z'_k + \sum_{j \in J_L^k} W_j^l + \sum_{j \in J_R^k} W_j^r + \max(0, \sum_{j \in J_I} p_j - (m-k)(\max_{j \in J_I} d_j - \min_{j \in J_I} r_j))$$

This yields the following feasibility condition which can be computed in $O(nm \log m)$ time:

Condition 12: The instance is infeasible if $\overline{W}_{\Delta}^{k} > k(t_2 - t_1)$ for a given value of $k = 1, ..., m-1$.

3.4. Adjustment rules

Recall that the slack of a job $j \in \mathcal{J}$ is equal to $s_j = m(t_2 - t_1) - W_{\Delta}(J \backslash \{j\})$. In Baptiste et al. (1999), the slack of every job $j \in \mathcal{J}$ is used for adjusting its head and tail in the following way:

- if $s_j < W_j^l$ then $\overline{r}_j = \max(r_j, t_2 - s_j)$
- if $s_j < W_j^r$ then $\overline{d}_j = \min(d_j, t_1 + s_j)$

where $\overline{r}_j$ and $\overline{d}_j$ refer to the adjusted values of r_j and d_j, respectively.

Note that these adjustments are still valid if W_{Δ} is replaced by $\tilde{W}_{\Delta}$ or $\overline{W}_{\Delta}$ in the slack computation. Interestingly, we show that the heads and tails of only few jobs need to be adjusted.

Proposition 2: Let $j_0 \in \mathcal{J}$ and $s_{j_0} = m(t_2 - t_1) - \tilde{W}_{\Delta}(J \backslash \{j_0\})$.
If in the optimal solution of $(P2)$ we have $z_{j_0} = 1$, then $\overline{r}_{j_0} = r_{j_0}$ and $\overline{d}_{j_0} = d_{j_0}$.

Proof. Consider the optimal solution of $(P2)$ defined on the set J, and let $Z_2^*(J)$ denote its optimal objective value. By removing the variables x_{j_0}, y_{j_0} and z_{j_0}, we obtain a feasible solution of $(P2)$ defined on the set $J \backslash \{j_0\}$, with objective value equal to $Z_2^*(J) - p_{j_0}$. Therefore, we have $Z_2^*(J \backslash \{j_0\}) \leq Z_2^*(J) - p_{j_0}$. That is, $\tilde{W}_{\Delta}(J \backslash \{j_0\}) \leq \tilde{W}_{\Delta}(J) - p_{j_0}$. Since $\tilde{W}_{\Delta}(J) \leq m(t_2 - t_1)$, then $p_{j_0} \leq m(t_2 - t_1) - \tilde{W}_{\Delta}(J \backslash \{j_0\}) = s_{j_0}$. Thus, $s_{j_0} \geq \max\left(W_{j_0}^r, W_{j_0}^l\right)$. ■

An interesting consequence of Proposition 2 is that the value of $\tilde{W}_{\Delta}(J \backslash \{j_0\})$ has to be computed for at most $m_L + m_R$ jobs. Therefore, the adjustments which are based on the computation of $\tilde{W}_{\Delta}$ can be performed in $O(m(n \log n)^2)$ time.

In the sequel, we show how to deduce in constant time $\overline{W}_{\Delta}(J \backslash \{j\})$ from $\overline{W}_{\Delta}(J)$ for any job $j \in J$.

Proposition 3: Let a_k and b_k denote the k^{th} largest $\overline{W}_j^l$ and $\overline{W}_j^r$ in $J_{LIR} \cup J_{LR} \cup J_{LI}$ and $J_{LIR} \cup J_{LR} \cup J_{RI}$, respectively. Define:

$$\lambda_l = \begin{cases} a_{m_L+1} & \text{if } |J_{LIR} \cup J_{LR} \cup J_{LI}| > m_L \\ 0 & \text{otherwise} \end{cases}$$

$$\lambda_r = \begin{cases} b_{m_L+1} & \text{if } |J_{LIR} \cup J_{LR} \cup J_{RI}| > m_R \\ 0 & \text{otherwise} \end{cases}$$

Thus, we have

$$\overline{W}_\Delta(J\backslash\{j\}) =$$

$$\begin{cases} \overline{W}_\Delta(J) - W_j & \text{for } j \in J_L \cup J_R \\ \max\{W_\Delta(J) - W_j, W'_\Delta(J) - p_j\} & \text{for } j \notin J_L \cup J_R \cup \overline{J}_L \cup \overline{J}_R \\ \max\{W_\Delta(J) - W_j, W'_\Delta(J) - W_j^l - \lambda_l\} & \text{for } j \in \overline{J}_L \backslash \overline{J}_R \\ \max\{W_\Delta(J) - W_j, W'_\Delta(J) - W_j^r - \lambda_r\} & \text{for } j \in \overline{J}_R \backslash \overline{J}_L \\ \max\{W_\Delta(J) - W_j, W'_\Delta(J) - p_j + \overline{W}_j^r + \overline{W}_j^l - \lambda_l - \lambda_r\} & \text{for } j \in \overline{J}_L \cap \overline{J}_R \end{cases} \tag{3.35}$$

Proof. Since we clearly have $W_\Delta(J\backslash\{j\}) = W_\Delta(J) - W_j$, then the proof is restricted to the computation of $W'_\Delta(J\backslash\{j\})$. Recall that

$$W'_\Delta(J) = \sum_{j \in J_{LI} \cup J_{RI} \cup J_{LIR} \cup J_{LR} \cup J_I} p_j - \sum_{j \in \bar{J}_L} \overline{W}_j^l - \sum_{j \in \bar{J}_R} \overline{W}_j^r + \sum_{j \in J_L} W_j^l + \sum_{j \in J_R} W_j^r$$

- If $j \in J_L$, then the contribution of j in $W'_\Delta(J)$ is W_j^l. Since $W_j = W_j^l$ for all $j \in J_L$, then $W'_\Delta(J\backslash\{j\}) = W'_\Delta(J) - W_j$. The same reasoning holds if $j \in J_R$.

- If $j \notin J_L \cup J_R \cup \overline{J}_L \cup \overline{J}_R$, then j contributes by p_j. Therefore, $W'_\Delta(J\backslash\{j\}) = W'_\Delta(J) - p_j$.

- If $j \in \overline{J}_L \backslash \overline{J}_R$ then the contribution of j in $W'_\Delta(J)$ is $p_j - \overline{W}_j^l$. That is, removing j from J yields a decrease of $W'_\Delta(J)$ by W_j^l. Moreover, if $|J_{LIR} \cup J_{LR} \cup J_{LI}| > m_L$, then there exists a job $k \in \overline{J}_L$ such that $\overline{W}_k^l = a_{m_L+1}$ which will replace j in $\overline{J}_L$. Which yields an additional decrease of $W'_\Delta(J)$ by a_{m_L+1}. Otherwise, no additional decrease of $W'_\Delta(J)$ occurs. Consequently, $W'_\Delta(J\backslash\{j\}) = W'_\Delta(J) - W_j^l - \lambda_l$. In a similar way, one can prove that $W'_\Delta(J\backslash\{j\}) = W'_\Delta(J) - W_j^r - \lambda_r$ for $j \in \overline{J}_R \backslash \overline{J}_L$, and $W'_\Delta(J\backslash\{j\}) = W'_\Delta(J) - p_j + \overline{W}_j^r + \overline{W}_j^l - \lambda_l - \lambda_r$ for $j \in \overline{J}_L \cap \overline{J}_R$. ■

An immediate consequence of Proposition 3 is that the adjustments which are based on the computation of $\overline{W}_\Delta$ can be computed in $O(n \log m)$ time. Interestingly, the following proposition enables more efficient implementation of these adjustments.

Proposition 4: Let $s_j = m(t_2 - t_1) - \overline{W}_\Delta(J\backslash\{j\})$. Then, for all job $j \in \mathcal{J}\backslash(\overline{J}_L \cap \overline{J}_R)$, we have

i) If $W_j = W_j^l$, then $\overline{r}_j = r_j$
ii) If $W_j = W_j^r$, then $\overline{d}_j = d_j$

Proof. Firstly, note that the adjustments are performed only if no infeasibility has been detected. That is, $\overline{W}_{\Delta}(J) \leq m(t_2 - t_1)$. Now, assume that $W_j = W_j^l$. Let us prove that $s_j \geq W_j^l$. Several cases have to be considered:

- For $j \in J_L \cup J_R$, we have $s_j = m(t_2 - t_1) - \overline{W}_{\Delta}(J) + W_j \geq W_j^l$.
- For $j \notin J_L \cup J_R \cup \overline{J}_L \cup \overline{J}_R$, we have $s_j = m(t_2 - t_1) - \max\{W_{\Delta}(J) - W_j, W'_{\Delta}(J) - p_j\}$ $= \min\{m(t_2 - t_1) - W_{\Delta}(J) + W_j^l, m(t_2 - t_1) - W'_{\Delta}(J) + p_j\}$. Note that $m(t_2 - t_1) - W_{\Delta}(J) \geq 0$ and $m(t_2 - t_1) - W'_{\Delta}(J) \geq 0$. Since $p_j \geq W_j^l$, then $s_j \geq W_j^l$.
- For $j \in \overline{J}_L \backslash \overline{J}_R$, we have $s_j = m(t_2 - t_1) - \max\{W_{\Delta}(J) - W_j, W'_{\Delta}(J) - W_j^l - \lambda_l\}$ $= \min\{m(t_2 - t_1) - W_{\Delta}(J), m(t_2 - t_1) - W'_{\Delta}(J) + \lambda_l\} + W_j^l$. Since $\lambda_l \geq 0$, then $s_j \geq W_j^l$.
- For $j \in \overline{J}_R \backslash \overline{J}_L$, we have $s_j = m(t_2 - t_1) - \max\{W_{\Delta}(J) - W_j, W'_{\Delta}(J) - W_j^r - \lambda_r\}$ $= \min\{m(t_2 - t_1) - W_{\Delta}(J) + W_j^l, m(t_2 - t_1) - W'_{\Delta}(J) + W_j^r + \lambda_r\}$. Note that since $W_j = \min(W_j^l, W_j^r)$, then $W_j^r \geq W_j^l$. Which yields $s_j \geq W_j^l$.

Similarly, one can prove that if $W_j = W_j^r$, then $s_j \geq W_j^r$. ■

The following result shows how the lifting procedure can be useful to obtain better adjustments.

Proposition 5:

- If Condition 10 detects an infeasibility by setting $W_{j_0} = W_{j_0}^l$ ($j_0 \in \mathcal{J}$), then $\bar{r}_{j_0} = \max(r_{j_0}, t_2 - W_{j_0}^l + 1)$.
- If Condition 10 detects an infeasibility by setting $W_{j_0} = W_{j_0}^r$ ($j_0 \in \mathcal{J}$), then $\bar{d}_{j_0} = \min(d_{j_0}, t_1 + W_{j_0}^r - 1)$.

Proof. Assume that setting the starting time of job j_0 at r_{j_0} (i.e. $W_{j_0} = W_{j_0}^l$) yields $\sum_{j \in J_{l,k}} W_j > k(t_2 - t_1)$ for a given value of k and l. Then, the starting time of job j_0 can be adjusted to $r_{j_0} + 1$. Now, setting the starting time of j_0 at any value in the interval $[r_{j_0}, t_2 - W_{j_0}^l]$ would yield $W_{j_0} \geq W_{j_0}^l$ and the infeasibility condition would continue to hold. Consequently, job j_0 cannot start processing before $t_2 - W_{j_0}^l + 1$. The deadline adjustment can be proven in a similar way. ■

Example 5: Consider the 4 job-2 machine instance which data is depicted in Table 2 and let $\Delta = [6, 12]$.

insert Table 2 here

We have $W_1 = W_2 = 3$, $W_3 = 1$, and $W_4 = 4$. By setting $W_1 = W_1^l = 4$, $k = 1$ and $l = 3$, we obtain $\lambda_k(l) = 2$ and $J_{l,k} = \{1, 2\}$. Therefore, we get $\sum_{j \in J_{l,k}} W_j = 3 + 4 > k(t_2 - t_1) = 6$ and an infeasibility is detected according to Condition 10. Consequently, the release date of job 1 is

adjusted to $\bar{r}_1 = t_2 - W_1^l + 1 = 9$. It is worth noting that no adjustments can be performed using the classical energetic reasoning. Also, the reader should have noticed that the values of W_j^l and W_j^r are computed using the corrected formulas (2.4) and (2.5). Indeed, no adjustment can be performed if W_1^l is computed using the formulas proposed by Néron et al. (2001).

In our experiments, the only considered sets of time intervals for the enhanced energetic reasoning are $\{[r_j, d_j] : j \in \mathcal{J}\} \cup \{[d_j - p_j, r_j + p_j] : j \in \mathcal{J} \text{ and } d_j - p_j < r_j + p_j\}$. The overall procedure is reiterated until an infeasibility is detected or no adjustment is performed. Each iteration requires $O(n)$ computations of the enhanced feasibility conditions and adjustment rules.

4. Computational results

The performance of the proposed procedures has been assessed through an empirical comparison with the feasibility and adjustment techniques which are based on the classical energetic reasoning. The test problems are generated in the following way: The processing times, heads and tails are drawn from the discrete uniform distribution on $[1, n]$. The number of jobs n is taken equal to $60, 70, 80$ and 90. The number of machines m is taken equal to 3, 5 and 7. For each combination of n and m, 10 instances are generated. As it has been observed by Gharbi and Haouari (2007), this test generation yields very challenging instances. Indeed, in our experiments, we found that one of the best existing branch-and-bound algorithms (Gharbi and Haouari, 2002), and denoted hereafter by B & B_1, fails to solve all of the instances with 5 and 7 machines within a time limit of 300 seconds. All the procedures were coded in C and implemented in Visual C++ 6.0 on a Pentium IV 3.2 GHz Personal Computer with 1.5 GB RAM.

We have embedded the proposed feasibility and adjustment procedures in the branch-and-bound algorithm (B & B_2) proposed by Gharbi and Haouari (2005) to obtain several variants which are described as follows:

- *Classical_ER:* the classical energetic reasoning-based procedure is embedded
- *Exact_ER:* the embedded feasibility and adjustment procedure is based on the computation of $\tilde{W}_\Delta$
- *Approximate_ER*: the embedded feasibility and adjustment procedure is based on the computation of $\overline{W}_\Delta$
- *Lifted_Classical_ER*: the classical energetic reasoning-based feasibility procedure is lifted using Condition 10
- *Lifted_Exact_ER:* the exact energetic reasoning-based feasibility procedure is lifted using Condition 11
- *Lifted_Approximate_ER:* the approximate energetic reasoning-based feasibility procedure is lifted using Condition 12

We report in Table 3 a summary of the results that were obtained with the eight branch-and-bound variants. For each variant, we report the average CPU time (*Time*) as well as the average percentage of unsolved instances within the time limit of 300 sec. (*Unsolved*). Table 3 provides strong evidence of the value of embedding the proposed techniques. We observe that

embedding the classical energetic reasoning in B & B_2 makes its performance considerably worse. However, if the classical energetic reasoning is replaced by the enhanced approximate one, then we observe a dramatic decrease of the percentage of unsolved instances (90.56%) and CPU time (82.24%). Moreover, the obtained algorithm (*Approximate_ER)* largely outperforms the two best algorithms of the literature. Also, we observe that although the lifting procedure does not improve the approximate and the exact energetic reasoning, it consistently improves the classical energetic reasoning. Indeed, the *Lifted_ Classical_ ER* solves 67.16% more instances than *Classical _ER* does, and requires on average 4.77 times less CPU time. At this point, it should be mentioned that, in our experiments, no improvement has been observed by embedding the adjustment based on Proposition 5.

insert Table 3 here

Tables 4-6 depict the detailed results of the performance of the best existing branch-and-bound algorithm (B & B_2), the classical energetic reasoning-based branch-and-bound algorithm (*Classical_ ER*) and our best algorithm (namely, *Approximate_ ER*). From these tables, we observe that instances with $m = 3$ are easy to solve since all of them have been solved by both B & B_2 and *Approximate_ ER* within an average CPU time of about 2 seconds. Yet, the *Classical_ ER* fails to solve 12.50% of these instances and requires on average a CPU time of 120.92 seconds. Also, it is worth noting that only 20% of the largest instances ($n = 90$ and $m = 7$) have been solved by *Classical_ ER* within the time limit of 300 seconds, while *Approximate_ ER* was able to solve all of these instances within an average CPU time of about 35 seconds.

insert Tables 4-6 here

A better picture of the relative performance of the three algorithms is shown in Tables 7 and 8. In these tables, we restrict the analysis to those instances that were solved by all of the three algorithms. We observe that *Approximate_ ER* clearly outperforms the best existing algorithm since it is on average 6.29 times faster than B & B_2 and explores about 12 times less nodes than B & B_2 does. This performance is even more dramatic if the enhanced approximate energetic reasoning is compared to the classical one. Indeed, we observe that *Approximate_ ER* is on average about 60.22 times faster than *Classical_ ER*, and explores 34.3% less nodes than *Classical_ ER* does.

insert Tables 7 and 8 here

5. Conclusion

In this paper, we have proposed new feasibility conditions and time bounds adjustments for the $P|r_j, q_j|C_{\max}$. The proposed feasibility conditions provide necessary conditions for the existence of

a feasible nonpreemptive schedule with a given makespan for a $P|r_j, q_j|C_{\max}$ instance. We have shown how to enhance the energetic reasoning through formulating a binary program that yields an accurate estimate of the total work. The resulting problem amounts to solving a minimum-cost network flow problem and is therefore solvable exactly in polynomial time. In addition, we have shown that a relaxed version of this problem could be used to provide a fairly good estimate of the total work value, but requires very low computational burden. Moreover, we have proposed new adjustment procedures that aim at efficiently tightening the time bounds. Our computational experiments carried out on a set of hard instances show that the performance of a state-of-the-art branch-and-bound algorithm has been substantially improved through embedding the proposed feasibility and adjustment procedures. Consequently, we found that many previously unsolved hard $P|r_j, q_j|C_{\max}$ instances were solved to optimality using the revisited energetic reasoning.

Future research needs to be focused on investigating a similar enhanced energetic reasoning to more complex scheduling problems such as the resource constrained project scheduling problem. Moreover, an interesting issue which deserves further investigation is the determination of the set of relevent time points for the enhanced energetic reasoning.

Acknowledgement:

The second author would like to thank Princess Fatimah Alnajras's Research Chair of Advanced Manufacturing Technology for supporting this research.

References

[1] **Baptiste P, Demassey S.** *Tight LP bounds for resource constrained project scheduling.* OR Spektrum 2004; 26, 251-262.

[2] **Baptiste P, Le Pape C, Nuijten W.** *Satisfiability tests and time bound adjustments for cumulative scheduling problems.* Annals of Operations Research 1999; 92, 305-333.

[3] **Brucker P, Jurisch B, Kramer A.** *The job-shop problem and immediate selection.* Annals of Operation Research 1994; 50, 73-114

[4] **Carlier J.** *Scheduling jobs with release dates and tails on identical machines to minimize the makespan.* European Journal of Operational Research 1987; 29, 298-306.

[5] **Carlier J, Pinson E.** *An algorithm for solving the job-shop problem.* Management Science 1989; 35,164-176.

[6] **Carlier J, Pinson E.** *A practical use of Jackson's Preemptive Schedule for solving job-shop problem.* Annals of Operation Research 1990; 26, 269-287.

[7] **Carlier J, Pinson E.** *Adjustment of heads and tails for the job-shop problem*, European Journal of Operation Research 1994; 78 ,146–161.

[8] **Carlier J, Pinson E.** *Jackson's Pseudo Preemptive Schedule for the $Pm/r_j, q_j/C_{\max}$ scheduling problem.* Annals of Operations Research 1998; 83, 41-58.

[9] **Dorndorf U, Pesch E, Phan-Huy T.** *A branch-and-bound algorithm for the resource-constrained project scheduling problem.* Mathematical Methods of Operations Research 2000; 52, 413-439.

[10] **Erschler J, Lopez P, Thuriot C.** *Raisonnement Temporel sous Contraintes de Ressources et Problèmes d'Ordonnancement.* Revue d'Intelligence Artificielle 1991; 5, 7-32.

[11] **Gharbi A, Haouari M.** *Minimizing Makespan on Parallel Machines Subject to Release Dates and Delivery Times.* Journal of Scheduling 2002; 5, 329-355.

[12] **Gharbi A, Haouari M.** *Optimal Parallel Machines Scheduling with Availability Constraints.* Discrete Applied Mathematics 2005; 148, 63-87.

[13] **Gharbi A, Haouari M.** *An Approximate Decomposition Algorithm for Scheduling on Parallel Machines with Heads and Tails.* Computers and Operations Research 2007; 34, 868-883.

[14] **Graham R.L., Lawler E.L., Lenstra J.K., Rinnooy Kan A.H.G**. *Optimization and approximation in deterministic sequencing and scheduling: a survey.* Annals of Discrete Mathematics 1979; 5, 287-326.

[15] **Gusfield D.** *Bounds for naive multiple machine scheduling with release times and deadlines.* Journal of Algorithms 1984; 5, 1-6.

[16] **Haouari M, Gharbi A.** *An Improved Max-Flow Based Lower Bound for Minimizing Maximum Lateness on Identical Parallel machines.* Operations Research Letters 2003; 31, 49-52.

[17] **Haouari M, Gharbi A.** *Lower Bounds for Scheduling on Identical Parallel Machines with Heads and Tails.* Annals of Operations Research 2004; 129, 187-204.

[18] **Horn WA.** *Some simple scheduling algorithms.* Naval Research Logistics Quarterly 1974; 21, 177-185.

[19] **Lahrichi A.** *Ordonnancements: La Notion de "Parties Obligatoires" et son Application aux Problèmes Cumulatifs*, RAIRO-RO 1982; 16, 241-262.

[20] **Néron E, Baptiste Ph, Gupta JND.** *Solving hybrid flow shop problem using the energetic reasoning and global operations.* Omega 2001; 29, 501-511.

[21] **Orlin J. B.** *A Faster Strongly Polynomial Minimum Cost Flow Algorithm.* In Proceedings 20th ACM Symposium on Theory of Computing, 1988; 377–387.

[22] **Tercinet F, Lenté C, Néron E.** *Mixed satisfiability tests for multiprocessor scheduling with release dates and deadlines.* Operations Research Letters 2004; 32, 326-330.

[23] **Tercinet F, Néron E, Lenté C.** *Energetic reasoning and bin-packing problem for bounding a parallel machine scheduling problem.* 4OR 2006; 4, 297-318.

[24] **Vandevelde A, Hoogeveen H, Hurkens C, Lenstra J.K.** *Lower Bounds for the Head-Body-Tail Problem on Parallel Machines: A Computational Study of the Multiprocessor Flow Shop.* INFORMS Journal on Computing 2005; 17, 305-320.

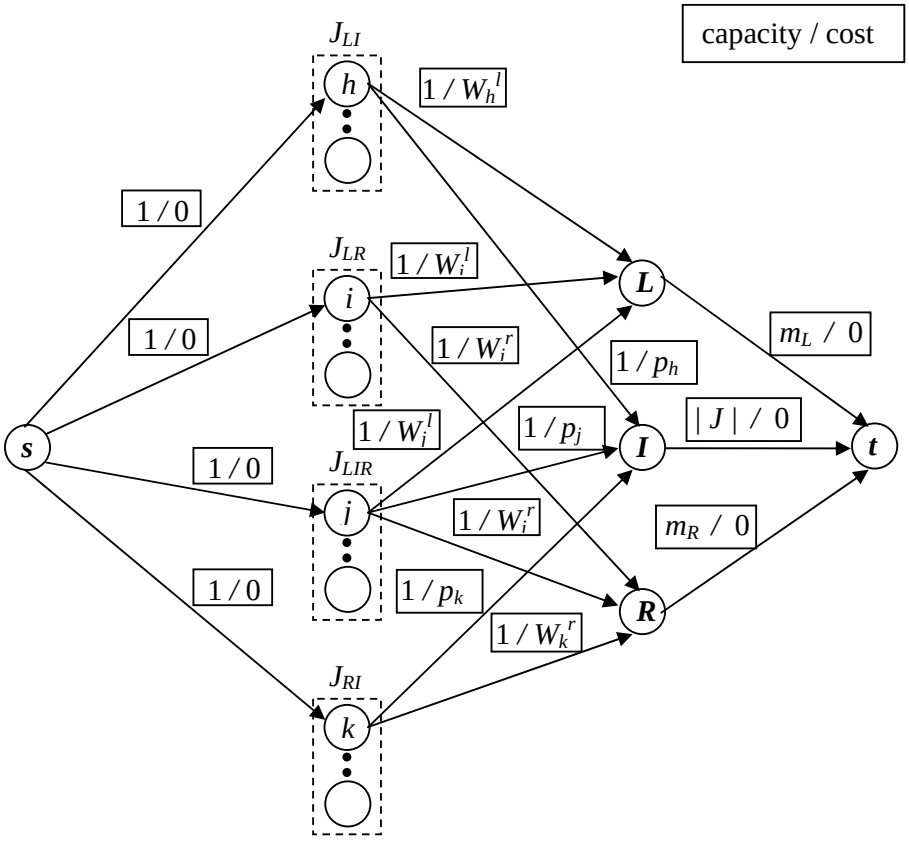

Figure 1. The network associated to (*P2*)

j	1	2	3	4
r_j	4	4	1	5
p_j	5	4	4	5
d_j	12	12	13	10

Table 1. Data of Example 4

j	1	2	3	4
r_j	8	6	4	6
p_j	5	3	3	4
d_j	14	12	7	11

Table 2. Data of Example 5

	Time	*Unsolved*
B&B$_1$	280.11	93.34
B&B$_2$	73.34	16.67
Classical_ ER	193.46	44.17
Exact_ER	145.75	25.84
Approximate_ER	**34.34**	**4.17**
Lifted_Classical_ER	40.57	6.67
Lifted_Exact_ER	148.08	35.00
Lifted_Approximate_ER	39.08	5.83

Table 3. Performance of the branch-and-bound algorithms

m	**3**		**5**		**7**		***Average***	
n	*Time*	*Unsolved*	*Time*	*Unsolved*	*Time*	*Unsolved*	*Time*	*Unsolved*
60	0.27	0.00	52.43	10.00	108.67	20.00	53.79	10.00
70	4.61	0.00	56.03	10.00	147.59	40.00	69.41	16.67
80	1.39	0.00	178.07	50.00	140.45	30.00	106.64	26.67
90	1.91	0.00	60.09	10.00	128.58	30.00	63.52	13.33
Average	2.04	0.00	86.66	20.00	131.32	30.00	73.34	16.67

Table 4. Detailed performance of $B\&B_2$

m	**3**		**5**		**7**		***Average***	
n	*Time*	*Unsolved*	*Time*	*Unsolved*	*Time*	*Unsolved*	*Time*	*Unsolved*
60	28.36	0.00	128.80	20.00	192.77	40.00	116.65	20.00
70	81.47	0.00	168.33	40.00	273.49	70.00	174.43	36.67
80	146.91	10.00	282.40	80.00	266.00	80.00	231.77	56.67
90	226.93	40.00	267.51	70.00	258.57	80.00	251.00	63.33
Average	120.92	12.50	211.76	52.50	247.71	67.50	193.46	44.17

Table 5. Detailed performance of *Classical_ER*

m	**3**		**5**		**7**		***Average***	
n	*Time*	*Unsolved*	*Time*	*Unsolved*	*Time*	*Unsolved*	*Time*	*Unsolved*
60	0.26	0.00	35.58	10.00	52.87	0.00	29.57	3.33
70	2.93	0.00	28.38	0.00	89.38	20.00	40.23	6.67
80	0.96	0.00	59.85	0.00	84.32	20.00	48.38	6.67
90	1.58	0.00	20.93	0.00	35.06	0.00	19.19	0.00
Average	1.43	0.00	36.19	2.50	65.41	10.00	34.34	4.17

Table 6. Detailed performance of *Approximate_ER*

m	**3**			**5**			**7**			***Average***		
n	*B&B*$_2$	*Classi-cal_ER*	*Approxi-mate_ER*	*B&B*$_2$	*Classi-cal_ER*	*Approxi-mate_ER*	*B&B*$_2$	*Classi-cal_ER*	*Approxi-mate_ER*	*B&B*$_2$	*Classi-cal_ER*	*Approxi-mate_ER*
60	0.27	28.36	0.35	10.52	86.01	2.60	36.15	121.28	3.48	15.65	78.55	2.14
70	4.61	81.47	3.00	5.41	80.55	2.26	15.60	211.62	4.49	8.54	124.55	3.25
80	1.47	129.90	1.38	2.09	211.98	1.56	33.58	130.02	1.27	12.38	157.30	1.40
90	2.86	178.21	1.88	47.84	191.71	0.58	0.85	92.87	2.80	17.18	154.26	1.75
Average	2.30	104.49	1.65	16.47	142.56	1.75	21.55	138.95	3.01	13.44	128.67	2.14

Table 7. Average CPU time

m	**3**			**5**			**7**			***Average***		
n	*B&B*$_2$	*Classi-cal_ER*	*Approxi-mate_ER*	*B&B*$_2$	*Classi-cal_ER*	*Approxi-mate_ER*	*B&B*$_2$	*Classi-cal_ER*	*Approxi-mate_ER*	*B&B*$_2$	*Classi-cal_ER*	*Approxi-mate_ER*
60	2775.00	1284.9	446.80	78841.63	23176.00	10247.13	164133.80	12792.67	6788.33	81916.82	12417.86	5827.42
70	62942.50	28833.4	21945.20	39575.67	3049.17	7351.83	77034.33	18478.67	10422.67	59850.83	16787.08	13239.90
80	18063.00	3385.22	2971.89	9792.00	4611.5	2721.50	102278.00	3176.50	1747.00	43377.67	3724.41	2480.13
90	34070.50	5109.33	4280.00	259818.70	1798.33	1.00	3355.50	2398.50	2093.00	99081.56	3102.06	2124.67
Average	29462.75	9653.21	7410.97	97006.99	8158.75	5080.36	86700.42	9211.58	5262.75	71056.72	9007.85	5918.03

Table 8. Average number of explored nodes