Università degli Studi di Roma Tre
Dipartimento di Informatica e Automazione
Via della Vasca Navale, 79 – 00146 Roma, Italy

Lagrangean Bounds and Lagrangean Heuristics for Just In Time Job-Shop Scheduling

Philippe Baptiste[1], Marta Flamini[2], Francis Sourd[3]

RT-DIA-103-2005

Dicembre 2005

(1) CNRS LIX laboratory, Ecole Polytechnique,
Palaiseau, France
(Philippe.Baptiste@polytechnique.fr)

(2) Dipartimento di Informatica e Automazione,
Universit Roma Tre,
Via della Vasca Navale 79,
00146 Roma, Italy.
(flamini@dia.uniroma3.it)

(3) CNRS LIP6 laboratory,
Paris, France.
(Francis.Sourd@lip6.fr)

ABSTRACT

We study the Job-Shop Scheduling problem with Earliness and Tardiness penalties. We describe two Lagrangean relaxations of the problem. The first one is based on the relaxation of precedence constraints while the second one is based on the relaxation of machine constraints. We introduce dedicated algorithms to solve the corresponding dual problems. The second one is solved by a simple dynamic programming algorithm while the first one requires the resolution of an NP-Hard problem by Branch and Bound. In both cases, the relaxations allow us to derive lower bounds as well as "Lagrangean" heuristic solutions. We finally introduce a simple local search algorithm to improve the best solution found. Computational results show the effectiveness of our approach.

1 Introduction

The importance of "Just-in-Time" inventory management in industry has motivated the study of theoretical scheduling models able of capturing the main features of this philosophy. Among these models, a lot of research effort was devoted to earliness-tardiness problems — where both early completion (which results in the need for storage) and tardy completion are penalized. So far, due to the difficulty of these problems, most of the effort was dedicated to the one-machine problem. Here we consider the earliness-tardiness problem in a complex jobshop environment.

We consider a job-shop scheduling problem that consists of a set of n jobs, $J = \{J_1, \ldots, J_n\}$, and a set of m machines $M = \{M_1, \ldots, M_m\}$. Each job J_i is a chain of n_i ordered operations, $O_i = \{o_i^1, \ldots, o_i^{n_i}\}$ where o_i^k is the k-th operation of job J_i. For each operation o_i^k we have a due date d_i^k and we denote its completion time C_i^k. Furthermore, each operation o_i^k has a processing time p_i^k, and has to be processed by a specific machine $M(o_i^k) \in M$. Likewise, for each machine $M_u \in M$ we denote $O(M_u)$ the set of operations that have to be processed by machine M_u. Let $E_i^k = \max(0, d_i^k - C_i^k)$ and $T_i^k = \max(0, C_i^k - d_i^k)$ respectively be the earliness and the tardiness of operation o_i^k. An operation o_i^k is an early(tardy) operation if $E_i^k > 0$ ($T_i^k > 0$).

For each operation o_i^k, we have two penalty coefficients α_i^k and β_i^k penalizing its early or tardy completion. Hence a cost function is defined considering that an early operation is penalized by the cost $\alpha_i^k E_i^k$ and a tardy operation is penalized by the cost $\beta_i^k T_i^k$.

The objective to minimize is the sum of earliness and tardiness costs of all operations, hence:

$$\min \sum_{i=1}^{n} \sum_{k=1}^{n_i} (\alpha_i^k E_i^k + \beta_i^k T_i^k) \tag{1}$$

Contraints of our problem are the following:

- the first operation starts after time 0, that is for any i in $\{1, \ldots, n\}$,

$$C_i^1 - p_i^1 \geq 0 \tag{2}$$

- Precedence constraints: for each pair of consecutive operations of the same job, o_i^k and o_i^{k+1}, operation o_i^{k+1} can not be processed before operation o_i^k has completed, that is, for $i = 1, \ldots, n$ and $k = 1, \ldots, n_i - 1$:

$$C_i^k \leq C_i^{k+1} - p_i^{k+1} \tag{3}$$

- Resource constraints (also known as disjunctive constraints): for each machine $M_u \in M$ two operations of two distinct jobs, o_i^k and o_j^h, in the set $O(M_u)$, cannot be processed simultaneously, that is, for $i, j = 1, \ldots, n, i \neq j$, and for $k = 1, \ldots, n_i$ and $h = 1, \ldots, n_j$:

$$C_i^k \geq C_j^h + p_i^k \quad \text{or } C_j^h \geq C_i^k + p_j^h \tag{4}$$

Beck and Refalo [2] investigated a special case of this problem in which all the operations but the last operations of each job have null earliness and tardiness penalty cost. This model can be criticized because it produces schedules in which the first $n_i - 1$ operations of each job J_i are processed as early as possible thus inducing wait times (and storage costs) between the last two operations of the job, which contradicts the Just-in-Time philosophy. In the proposed model, this issue can be avoided by considering that $d_i = d_i^{n_i}$ is the due-date of J_i and the due-date of any operation o_i^k is $d_i^k = d_i - \sum_{j>k} p_i^j$. Therefore, in the ideal situation where each operation completes at its due date, the job is on time and there is no wait period between operations. Several researchers have used Lagrangean relaxations for the job shop problem or for more complex problems such as the Resource Constrained Project Scheduling Problem. This latter problem differs from the Job-Shop due to the structure of precedence constraints between operations (a chin for the Job-Shop, an arbitrary acyclic graph for the Resource Constrained Project Scheduling Problem). Roughly speaking, two kind of relaxations have been studied.

- Starting from a time inexed formulation, one can relax machine constraints. The subproblem to solve is then reduced to scheduling a operations subjected to precedence constraints with arbitrary cost functions. Depending on the inital scheduling constraints, authors use various techniques to solve this problem. Either dynymic programming when the structure of precedence costraints is a chain or a tree [3] or a cut computation in the general case [7].

- Chen and Luh [4] relax the precedence constraints as well as constraints linking completion times of operations to the total cost. Once all these constraints are relaxed, the subproblem to solve is decomposed into a set of polynomially solvable problems.

In this paper, we evaluate the efficency of the two relaxation techniques based respectively on the relaxation of resource constraints and on precedence constraints. While the first rfelaxation closely follows the one described in [3] (Section 3), the second one is much stronger than [4] as we only relax the precedence constraints (2). For each of these relaxatiuons, we also introduce some simple Lagrangean heuristics. We finally introduce a simple local search algorithm (Section 4) to improve the best solution found. Computational results (Section 4) show the effectiveness of our approach.

2 Relaxing Precedence Constraints

In this section we rely on the basic formulation of the problem. To strengthen the upcomming relaxation, we define release dates $r_i^k = \sum_{1 \leq h \leq k-1} p_i^h$ for each operation o_i^k. We

then have:

$$\min \sum_{i=1}^{n} \sum_{k=1}^{n_i} \alpha_i^k E_i^k + \beta_i^k T_i^k$$
$$u.c. \begin{cases} C_i^k - p_i^k \geq r_i^k & 1 \leq i \leq n, 1 \leq k \leq n_i \\ C_i^k \leq C_i^{k+1} - p_i^{k+1} & 1 \leq i \leq n, 1 \leq k \leq n_i - 1 \\ C_i^k \geq C_j^h + p_i^k \quad or \quad C_j^h \geq C_i^k + p_j^h & 1 \leq u \leq m, o_i^k, o_j^h \in O(M_u) \\ E_i^k = \max(0, d_i^k - C_i^k) & 1 \leq i \leq n, 1 \leq k \leq n_i \\ T_i^k = \max(0, C_i^k - d_i^k) & 1 \leq i \leq n, 1 \leq k \leq n_i \end{cases}$$

We associate to the precedence constraint between operaions o_i^k and o_i^{k+1} a Lagrangean multiplier, $\lambda_i^k \geq 0$. In the following C and λ denote respectively the vector of complition times and the vector of Lagrangean multipliers. The Lagrangean function $L(C, \lambda)$ can then be defined as follows.

$$L(C, \lambda) = \sum_{i=1}^{n} \sum_{k=1}^{n_i} \alpha_i^k E_i^k + \beta_i^k T_i^k + \sum_{i=1}^{n} \sum_{k=1}^{n_i - 1} \lambda_i^k \left(C_i^k - C_i^{k+1} - p_i^{k+1} \right) \tag{5}$$

Under the assumption $\lambda_i^0 = \lambda_i^{n_i} = 0$ we can write (5) in a more compact form:

$$L(C, \lambda) = \sum_{i=1}^{n} \sum_{k=1}^{n_i} \alpha_i^k(\lambda) E_i^k + \beta_i^k(\lambda) T_i^k - K(\lambda) \tag{6}$$

where

$$\begin{cases} K(\lambda) & = & \sum_{i=1}^{n} \sum_{k=1}^{n_i} \lambda_i^k (p_i^{k+1} + d_i^k - d_i^{k+1}) \\ \alpha_i^k(\lambda) & = & \alpha_i^k - \lambda_i^k + \lambda_i^{k+1} \\ \beta_i^k(\lambda) & = & \alpha_i^k + \lambda_i^k - \lambda_i^{k+1} \end{cases}$$

For a given value of vector λ we search for the minimum $L(\lambda)$ of $C \rightarrow L(\lambda, C)$ under the remaining constraints.

$$L(\lambda) = \min \sum_{i=1}^{n} \sum_{k=1}^{n_i} \alpha_i^k(\lambda) E_i^k + \beta_i^k(\lambda) T_i^k - K(\lambda)$$
$$u.c. \begin{cases} C_i^k - p_i^k \geq r_i^k & 1 \leq i \leq n, 1 \leq k \leq n_i \\ C_i^k \geq C_j^h + p_i^k \quad or \quad C_j^h \geq C_i^k + p_j^h & 1 \leq u \leq m, o_i^k, o_j^h \in O(M_u) \\ E_i^k = \max(0, d_i^k - C_i^k) & 1 \leq i \leq n, 1 \leq k \leq n_i \\ T_i^k = \max(0, C_i^k - d_i^k) & 1 \leq i \leq n, 1 \leq k \leq n_i \end{cases}$$

This problem can be decomposed into m independant single-machine scheduling subproblems with eraliness and tardiness penalty costs, i.e., $L(\lambda) = \sum_{u=1}^{m} L_u(\lambda) - K(\lambda)$, where $L_u(\lambda)$ is exactly

$$\min \sum_{o_i^k \in O(M_u)} \alpha_i^k(\lambda) E_i^k + \beta_i^k(\lambda) T_i^k$$
$$u.c. \begin{cases} C_i^k - p_i^k \geq r_i^k & o_i^k \in O(M_u) \\ C_i^k \geq C_j^h + p_i^k \quad or \quad C_j^h \geq C_i^k + p_j^h & o_i^k, o_j^h \in O(M_u) \\ E_i^k = \max(0, d_i^k - C_i^k) & o_i^k \in O(M_u) \\ T_i^k = \max(0, C_i^k - d_i^k) & o_i^k \in O(M_u) \end{cases}$$

It is easy to see that $P_u(\lambda)$ is a single machine scheduling problem with Earliness/Tardiness penalties. This problem is NP-Hard in the strong sense but can be solved efficiently by an advanced Branch and Bound procedure of Sourd and Kedad-Sidhoum [10]. Then we look for the vector λ^* that maximizes the function $L(\lambda)$ by applying a standard subgradient procedure. Note that this relaxation strictly dominates the relaxation of Chen and Luh [4] as we only relax the precedence constraints while they also relax the constraints $E_i^k = \max(0, d_i^k - C_i^k)$ and $T_i^k = \max(0, C_i^k - d_i^k)$.

A weaker but faster lower bound, $L^-(\lambda) = \sum_{u=1}^{m} L_u^-(\lambda)$, can be computed by considering a lower bound for each problem $P_{M_u}(\lambda, C)$ instead of its optimal solution. We use the lower bound of Sourd... that returns a preemptive scheduling of operations in $O(M_u)$ respecting release constraints.

Lagrangean Heuristics

Our Lagrangean heuristic is based on the single machine schedules computed at the last iteration of the subgradient method. The completion times on these schedules are used as a priority measures in a heuristic dispatching rule: We build a left shifted jobshop schedule iteratively as follows. At any iteration, select an unscheduled operation, whose job predecessor is scheduled, and whose priority is minimum. Schedule this operation as soon as possible after the completion time of its predecssor and after the first time point at which the machine it requires is available.

This left shifted schedule is then improved as follows: We build a precedence graph of all operations in which we add an edge between consecutive operations of the same job and consecutive operations of the same machine. We then look for an optimal schedule of operations meeting the precedence graph. This problem reduces to a project scheduling problem with Early/Tardy cost functions (and no resource constraints). This can be solved in polynomial time by [5] or by linear programming.

3 Relaxing Resource Constraints

In this section we consider the Lagrangean relaxation of resource constraints. As many researchers did before, we rely on a time-indexed formulation of the problem. We introduce a time horizon T that can be computed as follows:

$$T = \max_{i,k} d_i^k + \sum_{i,k} p_i^k$$

We also introduce the set of binary $\{0, 1\}$ variables X, in which the generic variable x_{ik}^t refers operation o_i^k in the instant t. Each variable x_{ik}^t is defined as follows:

$$x_{ik}^t = \begin{cases} 1 & \text{if operation } o_i^k \text{ starts at time } t \\ 0 & \text{otherwise} \end{cases}$$

Considering this new set of variables X we can compute the earliness and tardiness costs as follows.

$$E_i^k = \sum_{t=0}^{T-1} x_{ik}^t \max(0, d_i^k - t - p_i^k) \qquad T_i^k = \sum_{t=0}^{T-1} x_{ik}^t \max(0, t + p_i^k - d_i^k)$$

Now consider the resource constraints. For each machine $M_u \in M$ and for each instant time $t \in [0, T]$, two operations in $O(M_u)$, can not overlap, i.e.,

$$\sum_{i=1}^{n} \sum_{k:\begin{cases} 1 \le k \le n_i \\ M(o_i^k) = u \end{cases}} \sum_{\tau=t-p_i^k+1}^{t} x_{ik}^\tau \le 1 \tag{7}$$

Hence, given a time instant $t \in [0, T]$ and a machine $u \in M$, the above constraint means that if an operation $o_i^k \in O(M_u)$ starts before t and has not yet completed in t, at least for $t - p_i^k + 1$ time instants before t, no other operations in $O(M_u)$ can start. Altogether, the initial problem can be formulated as follows.

$$\min \sum_{i=1}^{n} \sum_{k=1}^{n_i} \alpha_i^k E_i^k + \beta_i^k T_i^k$$

$$u.c. \begin{cases} C_i^k = \sum_{t=0}^{T-1} t x_{ik}^t + p_i^k & 1 \le i \le n, 1 \le k \le n_i \\ C_i^k \le C_i^{k+1} - p_i^{k+1} & 1 \le i \le n, 1 \le k \le n_i - 1 \\ E_i^k = \max(0, d_i^k - C_i^k) & 1 \le i \le n, 1 \le k \le n_i \\ T_i^k = \max(0, C_i^k - d_i^k) & 1 \le i \le n, 1 \le k \le n_i \\ \sum_{i=1}^{n} \sum_{k:\begin{cases} 1 \le k \le n_i \\ M(o_i^k) = u \end{cases}} \sum_{\tau=t-p_i^k+1}^{t} x_{ik}^\tau \le 1 & 1 \le u \le m, 0 \le t \le T-1 \\ x_{ik}^t \in \{0, 1\} & 1 \le i \le n, 1 \le k \le n_i, 0 \le t \le T-1 \end{cases}$$

We associate a Lagrangean multiplier $\lambda_{ut} \ge 0$ to each resource constraint. As earliness and tardiness can be expressed as

$$E_i^k = \sum_{t=0}^{T-1} x_{ik}^t \max(0, d_i^k - t - p_i^k) \qquad T_i^k = \sum_{t=0}^{T-1} x_{ik}^t \max(0, t + p_i^k - d_i^k)$$

the lagrangean function $L(\lambda)$ is exactly

$$\min \sum_{i=1}^{n} \sum_{k=1}^{n_i} \sum_{t=0}^{T-1} w_i i k^t(\lambda) x_{ik}^t - K(\lambda)$$

$$u.c. \begin{cases} C_i^k = \sum_{t=0}^{T-1} t x_{ik}^t + p_i^k & 1 \le i \le n, 1 \le k \le n_i \\ C_i^k \le C_i^{k+1} - p_i^{k+1} & 1 \le i \le n, 1 \le k \le n_i - 1 \\ x_{ik}^t \in \{0, 1\} & 1 \le i \le n, 1 \le k \le n_i, 0 \le t \le T-1 \end{cases}$$

where $wiik^t$ and K are simple functions of λ

$$w^t_{i,k}(\lambda) = \alpha^k_i \max(0, d^k_i - t - p^k_i) + \beta^k_i \max(0, t + p^k_i - d^k_i) + \sum_{u=1}^{m} \sum_{s=t}^{t+p^k_i-1} \lambda_{us} \qquad K(\lambda) = \sum_{u=1}^{m} \sum_{t=0}^{T-1} \lambda_{ut}$$

It is easy to see that $L(\lambda)$ can be decomposed as $\sum_{i=1}^{n} L_i(\lambda) - K(\lambda)$ where $L_i(\lambda)$ equals

$$\min \sum_{k=1}^{n_i} \sum_{t=0}^{T-1} wiik^t(\lambda) x^t_{ik}$$

$$u.c. \begin{cases} C^k_i = \sum_{t=0}^{T-1} t x^t_{ik} + p^k_i & 1 \le k \le n_i \\ C^k_i \le C^{k+1}_i - p^{k+1}_i & 1 \le k \le n_i - 1 \\ x^t_{ik} \in \{0, 1\} & 1 \le k \le n_i, 0 \le t \le T-1 \end{cases}$$

Solving a single subproblem consists in scheduling the operations $o^1_i \dots o^{n_i}_i$ of job J_i, considering only the precedence constrains among them, in order to minimize an arbitrary objective function. As noticed by several authors [3, 7], this problem can be solved by dynamic programing in linear time.

We search for the optimal value of Lagrangean multiplier vector, λ^*, that maximizes the function $L(X^*, \lambda)$ iterating the above computation and applying the classical subgradient method for updating the vector λ.

4 Local Search

To improve the quality of the schedule we apply a simple local search method starting from the best known solution. A solution is defined as a the sequence of operations on all machines. As mentionned earlier, given a arbitrary set of sequences, we can compute in polynomial time the optimal schedule meeting this sequence. Of course, we can also detect that there is cycle in the sequence.

Two moves are considered in our local search. Given a sequence, we either swap two randomly chosen operations or we insert an operation at some other place in its sequence. We change the current sequence as soon as the cost of the resulting sequence is improved (provdied of course that no cycle is induced by the move).

5 Experimental Results

For each $(n, m) \in (\{10, 15, 20\} \times \{2, 5, 10\})$, we have generated 8 instances. Each instance is named according to the following pattern `I-n-m-DD-W-ID`. The meaning of `DD`, `W` and `ID` is explained below. In all cases, the processing times are chosen randomly in $[10, 30]$. Jobs are processed exactly once on each machine and the machine $M(o^k_i)$ in which operation o^k_i has to be processed is chosen randomly. The due date of the first operation of a job is chosen randomly. Due dates of other operations are generated according to two different schemes:

- Either the distance between due dates of consecutive operations is exactly equal to the processing time of the last operation (`DD = tight`)
- Or, the distance between due dates of consecutive opertaions is exactly equal to the processing time of the last operation plus a random number taken in [0, 10] (`DD = loose`).

The earliness and tardiness costs (α, β) are either both chosen randomly in $[0.1, 1]$ (`W = equal`) or α is chosen randomly in $[0.1, 0.3]$ while β is chosen randomly in $[0.1, 1]$. The latter distribution corresponds to the situation where we have a small earliness cost (typically storage cost) while the tardiness cost is high (`W = largeTardiness`).

Two instances (`ID = 1`) or (`ID = 2`) have been generated for each combination of parameters. Hence leading to $3 \times 3 \times 2 \times 2 \times 2 = 72$ instances of the problem. All of them are available online there *www.martahomepage*.

For both Lagrangian relaxation, the subgardient algorithm works as follows. At each iteration, we modify the the vector of lagrangean multipliers by adding $\rho\nabla$ where $\rho \in \Re^+$ and where ∇ is a subgradient. At first, $\rho = 1$ and all multipliers equal 0 and then, ρ is multiplied by 0.7 when the value of the objective function is not improved for ten consecutive iterations. The algorithm is stoped when the parameter ρ is less than 10^{-3}.

First we ran both lower bounds on all instances with 10 jobs. We report in Table 1 the results obtained on the 24 corresponding instances. The time-indexed formulation is better than the other formulation for the 75% of cases. It is also faster in the 70% of cases.

For the remaining experiments, we ran the time-indexed formulation on all instances as well as the local search. The following table reports the LB and UB values as well as the CPU time used. We also report the relative gap between the lower and the upper bound.

Instance	LR Machine Constraints		LR Precedence Constraints	
	LB	CPU	LB P	CPU
I-10-2-tight-equal-1	434	13	433	6
I-10-2-tight-equal-2	357	7	418	3
I-10-5-tight-equal-1	660	32	536	112
I-10-5-tight-equal-2	592	23	612	40
I-10-10-tight-equal-1	1126	121	812	225
I-10-10-tight-equal-2	1535	363	819	733
I-10-2-loose-equal-1	218	6	219	4
I-10-2-loose-equal-2	313	10	298	9
I-10-5-loose-equal-1	1263	114	1205	85
I-10-5-loose-equal-2	878	126	780	147
I-10-10-loose-equal-1	331	117	294	570
I-10-10-loose-equal-2	246	62	211	385
I-10-2-tight-largeTardiness-1	168	7	174	17
I-10-2-tight-largeTardiness-2	143	7	138	85
I-10-5-tight-largeTardiness-1	361	41	322	960
I-10-5-tight-largeTardinessual-2	420	26	461	67
I-10-10-tight-largeTardiness-1	574	158		
I-10-10-tight-largeTardiness-2	666	207	469	2184
I-10-2-loose-largeTardiness-1	413	9	408	13
I-10-2-loose-largeTardiness-2	135	9	137	18
I-10-5-loose-largeTardiness-1	168	42	159	428
I-10-5-loose-largeTardiness-2	355	17	313	158
I-10-10-loose-largeTardiness-1	356	228	314	1104
I-10-10-loose-largeTardiness-2	138	60	119	583

Table 1: Results Obtained on Instances with 10 Jobs

Instance	LB	CPU	UB	GAP(%)
I-15-2-tight-equal-1	2902	136	3559	18.4
I-15-2-tight-equal-2	1253	17	1579	20.6
I-15-5-tight-equal-1	964	26	1663	42
I-15-5-tight-equal-2	1630	79	2989	45.4
I-15-10-tight-equal-1	4389	555	8381	47.6
I-15-10-tight-equal-2	3539	825	7039	49.7
I-15-2-loose-equal-1	1014	13	1169	13.2
I-15-2-loose-equal-2	490	10	520	5
I-15-5-loose-equal-1	2449	218	4408	44.4
I-15-5-loose-equal-2	2818	323	4023	29.9
I-15-10-loose-equal-1	758	267	1109	31.6
I-15-10-loose-equal-2	1242	395	2256	44.9
I-15-2-tight-largeTardiness-1	720	16	916	21.3
I-15-2-tight-largeTardiness-2	843	11	956	11.8
I-15-5-tight-largeTardiness-1	1008	39	1538	34.4
I-15-5-tight-largeTardiness-2	626	34	848	26.1
I-15-10-tight-largeTardiness-1	649	77	972	33.2
I-15-10-tight-largeTardiness-2	955	268	1656	42.3
I-15-2-loose-largeTardiness-1	616	13	730	15.6
I-15-2-loose-largeTardiness-2	278	13	310	10.3
I-15-5-loose-largeTardiness-1	1098	110	1723	36.2
I-15-5-loose-largeTardiness-2	314	29	374	16
I-15-10-loose-largeTardiness-1	258	106	312	17.3
I-15-10-loose-largeTardiness-2	476	243	936	49.1

Table 2: Results Obtained on Instances with 15 Jobs

Instance	LB	CPU	UB	GAP(%)
I-20-2-tight-equal-1	1747	16	2008	12.9
I-20-2-tight-equal-2	858	14	1014	15.3
I-20-5-tight-equal-1	2506	152	3090	18.8
I-20-5-tight-equal-2	4923	329	7537	34.6
I-20-10-tight-equal-1	6656	1226	12951	48.6
I-20-10-tight-equal-2	5705	1190	9435	39.5
I-20-2-loose-equal-1	2388	25	2708	11.8
I-20-2-loose-equal-2	2970	63	3318	10.4
I-20-5-loose-equal-1	5571	639	9697	42.5
I-20-5-loose-equal-2	5496	518	8152	32.5
I-20-10-loose-equal-1	3538	1243	6732	47.4
I-20-10-loose-equal-2	1344	279	2568	47.6
I-20-2-tight-largeTardiness-1	1515	26	1913	20.8
I-20-2-tight-largeTardiness-2	1375	31	1594	13.7
I-20-5-tight-largeTardiness-1	2507	142	4147	39.5
I-20-5-tight-largeTardiness-2	1633	154	1916	14.7
I-20-10-tight-largeTardiness-1	3003	623	5968	49.6
I-20-10-tight-largeTardiness-2	2740	865	3788	27.6
I-20-2-loose-largeTardiness-1	1194	22	1271	6
I-20-2-loose-largeTardiness-2	734	13	857	14.3
I-20-5-loose-largeTardiness-1	2177	249	3377	35.5
I-20-5-loose-largeTardiness-2	2643	147	5014	47.2
I-20-10-loose-largeTardiness-1	2462	678	6237	60.5
I-20-10-loose-largeTardiness-2	1226	392	1830	33

Table 3: Results Obtained on Instances with 20 Jobs

References

[1] Adams, J., Balas, E. and Zawack D. (1988): The Shifting Bottleneck Procedure for Job Shop Scheduling. *Management Science*, *34*, 391–401.

[2] Beck, J.C. and Refalo, P. (2003): A Hybrid Approach to Scheduling with Earliness and Tardiness Costs. *Annals of Operations Research*, *118*, 49–71.

[3] Chen, H., Chu, C. and Proth, J.M. (1998): An Improvement of the Lagrangean Relaxation Approach for Job Shop Scheduling: A Dynamic Programming Method. *IEEE Transactions on Robotics and Automation*, *14*, 786–795.

[4] Chen, A. and Luh, P. An alternaltive framework to Lagrangian relaxation approach for job shop scheduling. *European Journal of Operational Research*, 149:499–512, 2003.

[5] Chretienne, Ph. and Sourd F. (2003): PERT Scheduling with Convex Cost Functions. *Theoretical Computer Science*, *292*, 145–164.

[6] Danna, E. (2003): Improving and Extending the Disjunctive MIP Model for the Earliness-Tardiness Job-Shop Scheduling Problem. *Proceedings of PMS'04*.

[7] Möhring, R., Schulz, A., Stork, F., and Uetz, M. Solving Project Scheduling Problems by Minimum Cut Computations. *Management Science* 49(3), March 2003, pp. 330-350.

[8] Nuijten, W., Bousonville, T., Focacci, F., Godard, D. and Le Pape, C. (2003): Towards a RealLife Manufacturing Scheduling Problem and Test Bed. *Proceedings of PMS'04*.

[9] Skutella, M. (2002): List Scheduling in Order of Alpha-Points on a Single Machine. *Technische Universitat Berlin Technical Report*, *734-2002*.

[10] Sourd, F. and Kedad-Sidhoum, S. (2003): The one Machine Scheduling with Earliness and Tardiness Penalties. *Journal of Scheduling*, *6*, 533–549.

[11] Vanhoucke, M., Demeulemeester, E. and Herroelen, W. (2001): An Exact Procedure for the Ressource-Constrained Weighted Earliness-Tardiness Project Scheduling Problem. *Annals of Operations Research*, *102*, 179–196.