Nicolau Santos

Tabu Search for Minimizing Total Tardiness in the Permutation Flowshop

Departamento de Matemática
Faculdade de Ciências da Universidade do Porto
Novembro de 2012

Nicolau Santos

Tabu Search for Minimizing Total Tardiness in the Permutation Flowshop

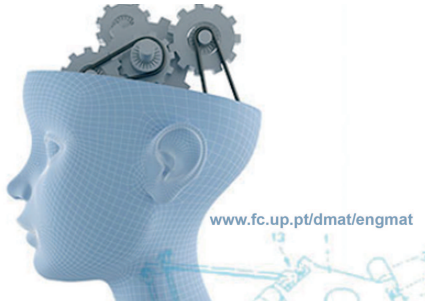
Tese submetida à Faculdade de Ciências da
Universidade do Porto para obtenção do grau de Mestre
em Engenharia Matemática

Orientador: Professor Doutor João Pedro Pedroso

Departamento de Matemática
Faculdade de Ciências da Universidade do Porto
Novembro de 2012

to Xana
and Leonardo

Acknowledgments



Tese realizada no âmbito do mestrado em Engenharia Matemática
Departamento de Matemática
Faculdade de Ciências da Universidade do Porto
http://www.fc.up.pt/dmat/engmat

Agradeço ao Engenheiro Rui Rebelo e a todos no INESC TEC pelo apoio no desenvolvimento deste trabalho. Agradeço à Professora Maria do Carmo Vaz Miranda Guedes e ao Professor João Nuno Tavares pela orientação no meu percurso académico.

Abstract

The practical application of flow shop problems puts them among the most widely studied topics in combinatorial optimization. This work presents a new tabu search approach to the permutation flow shop problem, with the objective of minimizing the total tardiness. Computational experiments with recent benchmark problems confirm the quality of the proposed algorithm.

Contents

List of Tables

List of Figures

Chapter 1

The Permutation Flowshop Problem

Flow shops represent production systems where different articles are industrialized in order to have the same sequence of technological operations. In this work, we assume there is only one machine available at each operation so we identify the sequence of operations as a sequence of machines. We also assume that infinite-capacity first-in-first-out buffers exist prior to each machine and job preemption is not allowed. The resulting problem is known as the permutation flow shop problem, as the production sequence can be defined by a single permutation. The problem is $\mathcal{NP}$-hard even when only one machine is considered [DJ90], and the number of elements in the solution space is $n!$. More precisely we have n jobs to process on m machines; given an n elements permutation $\pi = (\pi_1, \ldots, \pi_n)$, where π_k represents the job at position k; defining P_{ij} as the production time of job i in machine j and C_{ij} as the corresponding completion time, we can formulate the problem with the following equations:

$$C_{\pi_1,1} = P_{\pi_1,1} \tag{1.1}$$

$$C_{\pi_i,1} = C_{\pi_{i-1},1} + P_{\pi_i,1}, \qquad i = 2, \ldots, n \tag{1.2}$$

$$C_{\pi_1,j} = C_{\pi_1,j-1} + P_{\pi_1,j}, \qquad j = 2, \ldots, m \tag{1.3}$$

$$C_{\pi_i,j} = \max\{C_{\pi_i,j-1}, C_{\pi_{i-1},j}\} + P_{\pi_i,j}, \qquad i = 2, \ldots, n,\ j = 2, \ldots, m \tag{1.4}$$

Equation 1.1 defines the completion time of the first job on the first machine. Equation 1.2 states that in the first machine a job can only start after the previous job in the permutation sequence is concluded, while equation 1.3 states that the first job can only start at a given machine after completion in the previous machine. Equation 1.4 describes the general case: at a given machine, a job can only start after its processing in the previous machine has finished and after the processing of the previous job in the same machine has finished. This problem is widely studied, with an emphasis

usually being given to makespan minimization, i.e. minimization of the completion time of the last job in the last machine. In this work we will focus in minimizing the total tardiness. For every job i we define D_i as its due date. The tardiness of job i is $T_i = \max\{C_{im} - D_i, 0\}$. The total tardiness of a permutation π is defined as $T(\pi) = \sum_{i=1}^{n} T_i(\pi)$. We want to find the permutation π^* such that $T(\pi^*) \leq T(\pi)$, $\forall \pi \in \Pi$, where Π denotes the set off all the possible permutations of jobs. This objective is closely related to the timely fulfillment of client orders, and is thus oriented to service quality rather than speed in production. The current work has been applied to the shoe production industry, though there are many other applications.

Figure 1.1 displays an example of a flow shop with three jobs and three machines, and the production order (1,2,3). Job 3 is tardy, as it is completed after its due date. Note that although job 1 finishes before its due date it has no penalty, as earliness is not penalized; so its tardiness is the same of job 2 that finishes at the exact delivery time.

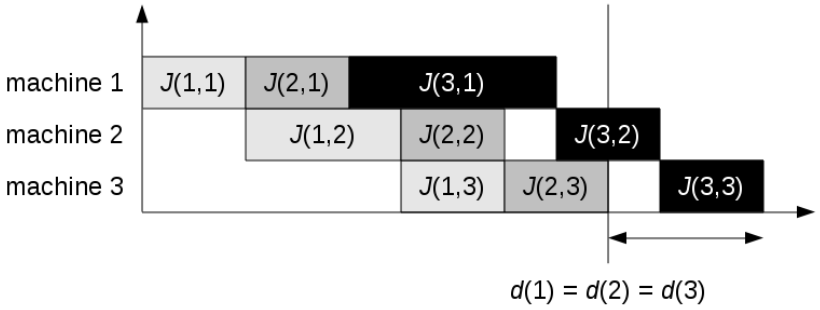


Figure 1.1: Gantt chart of example flow shop.

In the remaining of this chapter we provide a mixed-integer linear programming formulation for the permutation flowshop problem with total tardiness minimization objective and we describe the related literature. In the rest of the document we describe a tabu search metaheuristic developed to handle hundreds or even thousands of jobs in real production systems. In Chapter 2 we describe the characteristics of the referred algorithm, in Chapter 3 we present computational results conducted in benchmarks recently proposed in the literature. Finally, in Chapter 4, we present some conclusions.

1.1 Mathematical Formulation

In this section we describe a mixed-integer linear programming formulation of the described problem. For convenience we use slightly modified variables; we assume n jobs to process on m machines. The subscript symbols are: i for jobs ($1 \leq i \leq n$); k for job index ($1 \leq k \leq n$) and j for machines ($1 \leq j \leq m$). $P = \{P_{ij}\}$ is the production time of job i in machine j. The following variables are used:

R_i: release time of job i
D_i: due date of job i
$S_{kj} \geq 0$: start time of job in sequence position k on machine j
$F_{kj} \geq 0$: finish time of job in sequence position k on machine j
$T_k \geq 0$: tardiness of job in sequence position k
X_{ik}: $X_{ik} = 1$ if job i is executed in position k

X_{ik} is a binary integer variable. The remaining variables have the same type of the production time, so they can be integer or real. The following model defines the objective and the interactions between the variables:

$$\text{minimize } T = \sum_{j=1}^{n} T_j \tag{1.5}$$

$$\text{subject to } \sum_{i=1}^{n} X_{ik} = 1, \qquad \text{for } k = 1, \ldots, n \tag{1.6}$$

$$\sum_{k=1}^{n} X_{ik} = 1, \qquad \text{for } i = 1, \ldots, n \tag{1.7}$$

$$S_{kj} + \sum_{i=1}^{n} P_{ij} X_{ik} = F_{kj}, \qquad \text{for } k = 1, \ldots, n; j = 1, \ldots, m \tag{1.8}$$

$$F_{kj} \leq S_{k,j+1}, \qquad \text{for } k = 1, \ldots, n; j = 1, \ldots, m-1 \tag{1.9}$$

$$F_{kj} \leq S_{k+1,j}, \qquad \text{for } k = 1, \ldots, n; j = 1, \ldots, m-1 \tag{1.10}$$

$$F_{km} \leq \sum_{i=1}^{n} D_i X_{ik} + T_k, \qquad \text{for } k = 1, \ldots, n \tag{1.11}$$

$$\sum_{i=1}^{n} R_i X_{ik} \leq S_{k1}, \qquad \text{for } k = 1, \ldots, n \tag{1.12}$$

$$P_{ij}, R_i, D_i, S_{kj}, F_{kj}, T_k \geq 0, \qquad \forall i, j, k \tag{1.13}$$

$$X_{ik} \in \{0, 1\}, \qquad \forall i, k \tag{1.14}$$

Constraint 1.6 guarantees that each job is assigned to a sequence position while equation 1.7 insures each sequence position is assigned to only one job. Constraint 1.8 adds the processing time of each job to its starting time, resulting in the corresponding finishing time. Constraint 1.9 defines machine precedence, constraint 1.10 defines job precedence. Constraint 1.11 evaluates the total tardiness of each job, constraint 1.12 defines the starting time of jobs. Constraints 1.13 and 1.14 describe the types of the variables, the objective is to minimize the total tardiness as defined in 1.5.

1.2 Literature Review

Flow shops are widely studied in literature, since the pioneering work of [Joh54]. In a recent survey [GS06] more than 1200 papers on various aspects of the problem have been identified. Considering tardiness minimization related work the literature is divided in exact and heuristic solutions. Exact methods comprise branch and bound for the two-machine flow shop [SDG89, Kim93b, PCC02], and branch and bound methods for the general m-machine flow shop [Kim95, CFK06]; however, these methods can handle only small size instances (approximately 20 jobs and 8 machines). Heuristic methods range from simple dispatching rules ([PI77]) to more complex heuristics ([GS78], [Ow85], [Kim93a]). In [KLP96] the author extends is previous work with local search methods and more complex metaheuristics, namely simulated annealing (SA) and tabu search (TS). In [AR99] a TS method with intensification and diversification strategies is presented. In [PR98, HR04] the authors present SA methods with various perturbation schemes. In [FL08] a variable greedy algorithm was proposed. In [VRM08] a review and evaluation of various methods is presented. In this last work the authors propose a common benchmark set to conduct tests and evaluate the performance of the literature algorithms. In [VR10] three genetic algorithms (GA) with different generation schemes where proposed, including techniques as path-relinking. In [VR09] the authors conduct experiments in a parallel computational environment, evaluating the performance algorithms running as single or multiple processes. In [KTW10] a steady state GA with an elite based local search is proposed. These last three works represent the state of the art of the permutation flow shop problem with total tardiness criterion, and present results based on a common benchmark; so we will provide a direct comparison of our results with theirs in Chapter 3. In recent works related to the permutation flowshop, [RH09] conducted a study where blocking between machines is considered, in [RS08, SRP12] the authors analyze the problem for the permutation flowshop with sequence dependent setup times and total weighted

tardiness optimization criterion.

Chapter 2

Tabu Search

Tabu search (TS) was proposed in [Glo86] as a method for guiding heuristics based on solution improvement through the solution space. Given a solution π the move represents the way in which we modify π to obtain new solutions, $\mathcal{N}(\pi)$ represents the neighborhood of π and contains all the solutions we can obtain by applying a move. The solution π is locally optimal if $\pi \leq \pi^*$, $\forall \pi^* \in \mathcal{N}(\pi)$, π is a global optimum if $\pi \leq \pi^*$, $\forall \pi^* \in S$, where S denotes the space of all possible solutions. The main objective was to overcome the limitations of the traditional descent method described in figure 2.1.

```
procedure local_search(π)
    while π is not locally optimal do
        find s ∈ N(π) with f(s) < f(π)
        π ← s
    end do
    return π
end procedure
```

Figure 2.1: Local search *pseudocode*

This local search method explores the neighborhood of the current solution and moves to the best found neighbor until no improvement is possible; however, there is no guarantee of global optimality, and the quality of the obtained solution may be ratter poor. TS relies in two main mechanisms to escape local optima: the tabu move and the tabu list. In the tabu move, at each iteration we evaluate the neighborhood

and move to the best found neighbor, even if its objective is worse than the current solution's. By performing these moves to worse solutions TS is able to escape local optima; however this moves could be easily reversed in the following iterations. To prevent this second situation, TS has a tabu list, a short term memory list with the record of recently performed moves. These moves are forbidden for a given number of iterations, preventing reversion of recent moves and helping TS to travel through the solution space while avoiding local optima stagnation and cycling. Figure 2.2 describes the general process of TS.

```
procedure tabu_search(π)
    π_best ← π
    while termination criteria not met do
        find best s ∈ N(π)
        π ← s
        update tabu list
        if π < π_best then
            π_best ← π
        end if
    end do
    return π_best
end procedure
```

Figure 2.2: Tabu search *pseudocode*

Typically the algorithm runs until a given stopping criterion is met. The most common criteria are:

- maximum CPU allowed;
- maximum number of iterations reached;
- maximum number of iterations without improvement reached;
- a given lower bound is met.

In the literature there are many examples of successful TS applications, from the initial works of [HW87] in graph coloring to the fast implementation of TSAB [NS96], developed for the job shop problem. A generic introduction of TS can be found in [Gen03]; detailed information is available in [GL98]. In the remaining of this chapter we describe the characteristics of our TS implementation.

2.1 Moves and neighborhood

TS exploration is based in moving iteratively to a neighbor solution. Throughout the literature, three types of moves are used more prominently in scheduling problems, namely: swap-moves (swap job at position i with job at position $i+1$), exchange-moves (exchange the job at the i-th position with the job at the j-th position) and insertion-moves (remove the job at the i-th position and insert it at the j-th position).

In our algorithm we use the insertion-move: given a permutation π and a pair of positions (i, j), $i \neq j$, the permutation π' is obtained by removing job at position i and inserting it at position j, i.e.,

$\pi' = \pi_1, \ldots, \pi_{i-1}, \pi_{i+1}, \ldots, \pi_j, \pi_i, \pi_{j+1}, \ldots, \pi_n$ if $i < j$;

$\pi' = \pi_1, \ldots, \pi_{j-1}, \pi_i, \pi_j, \ldots, \pi_{i-1}, \pi_{i+1}, \ldots, \pi_n$ if $j < i$.

Having a set of jobs U we define $\mathcal{N}(U, \pi)$ as the neighborhood that contains all the possible insertion moves of the jobs in U. Besides solution quality, another advantage of using insertion moves is the speed up procedure we can apply. Suppose we have a permutation $\pi = (\pi_1, \pi_2, \pi_3)$ and we want to evaluate the possible insertions of a job π_4; then we would have to analyze the following permutations: $I_1 = (\pi_4, \pi_1, \pi_2, \pi_3)$, $I_2 = (\pi_1, \pi_4, \pi_2, \pi_3)$, $I_3 = (\pi_1, \pi_2, \pi_4, \pi_3)$ and $I_4 = (\pi_1, \pi_2, \pi_3, \pi_4)$. This could be done individually with formulas 1.1 to 1.4; however, by evaluating I_4 we are already evaluating the following underlined parts of $I_2 : (\underline{\pi_1}, \pi_4, \pi_2, \pi_3)$ and $I_3 : (\underline{\pi_1, \pi_2}, \pi_4, \pi_3)$. So with proper data structures important computational savings can be obtained. This was already observed in [FL08] and [VR10] but was not explained in detail. In section 2.3 we describe the steps of our speed up implementation . With this speed up we save approximately half of the computational time. Still, as the problems size increases, the complete neighborhood quickly becomes too large for full exploitation, so we apply a restriction on the size of the neighborhood to be evaluated. We select a subset of r randomly chosen jobs such that $r_{min} \leq r \leq r_{max}$ (both r_min and r_max are parameters of the algorithm). At each iteration we analyze only the restricted neighborhood and select the job that yields the best objective for the move.

2.2 Tabu list and search strategy

One of the key elements in TS is the tabu list, a mechanism of short term memory used to record information of recent moves. Our implementation of the tabu list consists

of an array τ: for each job i we assign a value τ_i. At iteration *iter* we say that job i is *tabu* if $\tau_i > iter$. Whenever we select a job i to perform the move it becomes tabu for t iterations, where $1 \leq t \leq t_{max}$, so $\tau_i = iter + t$. An exception occurs when the best known solution is improved; in such case we set $\tau_k = 0, \forall k \neq i$ and $\tau_i = iter + 1$ to prevent immediate reversion of job i. At each iteration the TS procedure can be summarized in the following steps:

- find the set L of legal (non tabu) moves;
- let r be an integer random number such that, $r_{min} \leq r \leq r_{max}$; define R as a set of r jobs randomly chosen from L;
- evaluate the moves in the set $\mathcal{N}(R, \pi)$ and perform the move that yields the best objective;
- taking into consideration the obtained objective value, update the tabu list.

This restricted neighborhood gives TS the ability to navigate very fast through the solution space, to maintain this characteristic we do not implement aspiration criteria.

2.3 Speedup implementation

In this section we describe the speed up implementation. Considering the generic case of n jobs to process on m machines, for each job i the production time on machine j is p_{ij} and the corresponding due date is d_i. Suppose we have a reduced permutation π of dimension $n-1$ and we want to evaluate all the possible insertions of a job π_t. All calculations are performed with two data structures: an $n \times m$ work matrix W and an n dimensional array a. The lines of W are used to calculate job insertions; more precisely, line i will be used to evaluate the completion times of all jobs in the permutation where job π_t is inserted at position i. Array a will be used for storing the accumulated tardiness of inserting job π_t at position i. The first step is to define the initial state of the machines and total initial tardiness of the system. We assume all these values to be zero initially and perform the following assignments:

$a_1 = 0$
for j=1 **to** m **do**
    $W_{1j} = 0$
end do

W_{1j} represents the initial state of machine j and a_1 the initial total tardiness. The next step is to evaluate the earliest completion time of each job of the reduced permutation:

for i=1 **to** n-1 **do**
    for j=1 **to** m **do**
        $W_{i+1,j} = \max\{W_{i+1,j-1}, W_{ij}\} + p_{\pi_i j}$
    end do
    $a_{i+1} = a_i + \max\{W_{i+1,m} - d_{\pi_i}, 0\}$
end do

After these calculations all the underlined partial permutations described in section 2.3 are evaluated, Now we evaluate the tardiness of inserting job π_t at each position:

for i=1 **to** n **do**
    for j=1 **to** m **do**
        $W_{ij} = \max\{W_{i,j-1}, W_{ij}\} + p_{\pi_t,j}$
    end do
    $a_i = a_i + \max\{W_{im} - d_{\pi_t}, 0\}$
end do

At this point the insertion of π_t at the n^{th} position is completely evaluated, now we finish the evaluation of the remaining insertions:

```
for i=1 to n-1 do
    for j=i to n-1 do
        for k=1 to m do
            W_ik = max{W_i,k-1, W_ik} + p_π_j,i
        end do
        a_i = a_i + max{W_im − d_π_j, 0}
    end do
end do
```

Note that in the above presented calculations $W_{0k} = 0$ and $W_{k0} = 0$, $\forall k$. The computational complexity of the presented speed up is $\mathcal{O}(n^2m)$ which is the same of evaluating all the insertions with formulas 1.1 to 1.4; however a substantial number of calculations is avoided and the used computational time is approximately halved.

2.4 Initialization

There are many ways of initializing TS and other metaheuristics, from random solutions to more sophisticated heuristics. The current trend in flowshop problems is the initialization using the NEH [NEJH83] described for instance in [RS07] for makespan objective. On our algorithm we will use the NEH_{edd} heuristic which is an adaptation of the original NEH for tardiness objective presented in [KLP96]. The initial solution is constructed with the following steps:

1. sort the jobs in non decreasing order of due dates (EDD rule);
2. select the first two jobs from EDD and sort them so as to minimize the associated total tardiness
3. for k from 2 to n: select the job k from EDD and insert it in all the k possible positions of sol; select the insertion that yields the lowest current total tardiness.

To calculate the sequence of the EDD we use a stable sorting algorithm, i.e., the relative order of elements with the same value is maintained. When performing the insertions, ties are broken in the following way: we test all the insertions from the first to the last position and assigning to the first found position.

It is also possible to initialize the algorithm with simpler or even random constructions, but the convergence is expected to become somewhat slower, as the first descent may become very long.

2.5 Diversification, intensification and calibration

Two recurrent subjects in TS are diversification and intensification. As the name suggests the first should allow the algorithm to travel through the solution space in a rich and diversified way; in contrast, intensification consists of focusing the exploration on a particular region. While studying the behavior of our algorithm in test problems it became clear that the r_{min}, r_{max} and t_{max} parameters could efficiently model the aforementioned situations. It was observed that low values of r_{min} and r_{max} combined with a high value of t_{max} would act as a diversification process while high values of r_{min} and r_{max} combined with a small t_{max} would localize the search and potentially provoke cycling. However the specific value that the parameters should assume or the frequency in which they should be changed seemed to be problem dependent. In order to overcome this situation we propose a three stage search procedure. The first stage consists of a highly diversified search with $r_{min} = 2$, $r_{max} = 3$ and $t_{max} = n$; the second stage makes a more thought exploration with $r_{min} = 3$, $r_{max} = 7$ and $t_{max} = n$; finally, a third stage with $r_{min} = 3$, $r_{max} = 7$ and $t_{max} = 1$ is applied as an intensification process. Computational time is divided evenly among each of the stages. Whenever a stage transition is applied we restart from the best found solution and clear the tabu list. These parameters were found experimentally and chosen in order to obtain a good compromise between solution quality and running time.

Chapter 3

Computational Results

The proposed tabu search was coded in mixed Python/Fortran wrapped with F2py [Pet09] and compiled with gfortran. The tests were performed on a computer with an AMD Athlon 64 Dual Core 3800+ processor and 2Gb of RAM running a Linux 64-bit operating system.

We used the benchmark test suit proposed in [VRM08] available in *http://soa.iti.es*. The benchmark was generated with different combinations of the following parameters: number of jobs n; number of machines m; tardiness factor (T) and due date range (R). Due dates are generated according to T and R with uniform distribution between $P\,(1 - T - R/2)$ and $P\,(1 - T + R/2)$ where P is a tight lower bound of the makespan proposed in [Tai93]. High values of T result in early due dates, and low values of R cause packed due dates. Job processing times are integers uniformly distributed between 1 and 99. The benchmark is composed of the following combinations for T, R, n and m: T={0.2, 0.4, 0.6}, R={0.2, 0.6, 1}, n={50, 150, 250, 350} and m={10, 30, 50}. The benchmark is composed of five instances of each combination of T, R, n and m, resulting in 108 groups and a total of 540 test instances.

To evaluate the quality of the proposed algorithm we conducted both regular and quick tests. In the regular tests we use CPU running times of $n.m.45$ milliseconds, allowing more computational time as the number of jobs/machines increases; a similar approach is also used in [VRM08, VR10, KTW10]. The regular computational times (in seconds) for the different combinations of n and m are presented in table 3.1 and we denote the algorithm by TS. In the fast tests we use one tenth of the aforementioned CPU time and denote the algorithm by FTS. In all runs the algorithms are executed using a single core. Due to the stochastic nature of the implemented algorithms both

methods are executed five times for each instance, results presented are their average. Usually flow shop algorithms are evaluated with the Relative Percent Deviation (RPD)

$n \backslash m$	10	30	50
50	22.5	67.5	112.5
150	67.5	202.5	337.5
250	112.5	337.5	562.5
350	157.5	472.5	787.5

Table 3.1: Computational times (in seconds) used for the various combinations of n and m in the normal runs (fast tests used one tenth of the time).

defined in equation 3.1; however, when the total tardiness objective is considered the optimal value of an instance may be zero and in that case a division by zero occurs. To overcome this situation one can replace the RPD by zero, if the algorithm finds the optimal solution and evaluate the error E, $E = Method_{sol} - Best$ otherwise. Another alternative is to use a different quality measure such as the the Relative Deviation Index (RDI) defined in equation 3.2.

$$RPD = \frac{Method_{sol} - Best}{Best} \times 100 \tag{3.1}$$

$$RDI = \frac{Method_{sol} - Best}{Worst - Best} \times 100 \tag{3.2}$$

In table 3.2 we present the mean RDI for the fast tests and in table 3.3 we present the mean RPD for the fast tests. We can observe that various instance groups with $T = 0.2$ and $R = 0.6$ or $R = 1$ seem to be easy to solve as the average RPD and RDI is zero. We can also already identify several groups with negative values, in these cases we are improving, on average, the objective with respect to previous best known solutions. The difficulties with the RPD also arise as we are getting zero division errors in five instances; these results are identified in the tables with the notation $^{a}/_{(b:c)}$ where a in the RPD for the instances of the group that behave normally, b is the number of instances of the group with zero division errors that did not achieve the optimal value and c is the corresponding mean error.

The tests with regular computational times show further improvement relatively to the fast tests results. It is now possible to identify a large number of groups with negative values. Paradoxically, the quality measures suggest different interpretations for some groups. As an example consider the instance group with $T = 0.2$, $R = 0.6$,

T	0.2			0.4			0.6			
R	0.2	0.6	1	0.2	0.6	1	0.2	0.6	1	*average*
50×10	1.07	0.00	0.00	1.10	1.17	0.87	1.36	1.27	1.66	0.94
50×30	1.94	1.68	0.82	1.91	1.84	1.61	2.40	2.09	2.20	1.83
50×50	2.18	1.74	1.14	1.96	2.65	2.03	2.31	2.02	1.97	2.00
150×10	0.87	0.00	0.00	0.41	0.93	0.22	2.13	1.20	0.67	0.71
150×30	0.27	-0.02	0.00	1.33	1.00	0.49	3.26	0.85	1.15	0.93
150×50	1.46	0.87	0.37	2.01	1.64	1.57	3.82	2.60	2.99	1.93
250×10	0.88	0.00	0.00	1.14	0.78	0.00	2.46	1.33	0.92	0.83
250×30	0.19	0.01	0.00	-0.94	-0.68	-0.09	2.64	0.30	-0.10	0.15
250×50	-0.81	-0.23	0.00	0.12	-0.33	0.64	3.46	0.72	0.12	0.41
350×10	0.77	0.00	0.00	0.91	0.92	0.07	2.92	2.67	2.95	1.25
350×30	1.11	0.00	0.00	-0.11	1.00	0.69	2.22	1.04	0.20	0.68
350×50	-0.24	-0.05	0.00	-0.66	-0.23	0.04	3.02	0.58	-0.68	0.20
average	0.81	0.33	0.19	0.77	0.89	0.68	2.67	1.39	1.17	0.99

Table 3.2: Average RDI for the fast tests.

T		0.2			0.4			0.6		
R	0.2	0.6	1	0.2	0.6	1	0.2	0.6	1	*average*
50×10	6.95	0.00	0.00	2.06	2.48	2.35	1.26	1.02	1.77	1.99
50×30	3.10	3.26	1.82	1.36	1.32	1.06	1.15	0.91	1.01	1.67
50×50	2.11	1.68	1.13	1.05	1.20	0.95	0.83	0.65	0.69	1.14
150×10	6.35	0.00	0.00	0.84	5.52	4.30	1.90	1.19	1.31	2.38
150×30	0.76	12.68	0.00	1.36	1.49	0.82	1.79	0.48	0.77	2.24
150×50	2.34	4.13	64.40	1.41	1.32	1.56	1.62	1.06	1.34	8.80
250×10	6.93	0.00	0.00	2.05	5.41	4.34	2.21	1.78	2.22	2.77
250×30	0.46	$^{0.00}/_{(3:74.20)}$	0.00	-0.95	-1.54	-0.99	1.48	0.15	-0.12	-0.17
250×50	-1.62	-3.63	0.00	0.09	-0.43	1.01	1.50	0.32	0.05	-0.30
350×10	6.01	0.00	0.00	1.54	6.95	$^{59.09}/_{(1:6.40)}$	2.25	3.36	7.10	9.59
350×30	4.21	0.00	0.00	-0.15	2.93	8.84	1.19	0.65	0.22	1.99
350×50	-0.55	$^{-2.19}/_{(1:3.00)}$	0.00	-0.52	-0.45	0.03	1.30	0.27	-0.48	-0.29
average	3.09	1.33	5.61	0.85	2.18	6.95	1.54	0.99	1.32	2.65

Table 3.3: Average RPD for the fast tests.

$n = 150$ and $m = 30$, the RDI value is -0.26 while the RPD value is -44.48. The best known solutions for the instances of this group range from hundreds to a few thousands and these small values seem to unbalance RPD's precision. Conversely, for the $T = 0.2$, $R = 0.6$, $n = 350$ and $m = 50$ instance group, the upper bounds are of tens of thousands and the RDI seems to ignore the obtained improvement. This indicates limitations in the performance measures, as to interpret the results we need explicit knowledge over the specific bounds of the instances. The overall performance of the algorithms is quite impressive, we were able to improve on average over the best known results of several groups of instances. In table 3.4 we present the mean RDI for the regular tests and in table 3.5 we present the mean RPD for the regular tests.

T	0.2			0.4			0.6			
R	0.2	0.6	1	0.2	0.6	1	0.2	0.6	1	*average*
50×10	0.48	0.00	0.00	0.34	0.29	0.32	0.51	0.02	0.52	0.28
50×30	0.70	0.80	0.36	0.89	1.05	0.60	0.89	0.71	0.94	0.77
50×50	1.09	0.85	0.48	1.04	1.35	1.04	1.33	0.82	0.87	0.99
150×10	0.00	0.00	0.00	-1.22	-0.38	-0.51	-0.76	-2.58	-1.75	-0.80
150×30	-2.09	-0.35	0.00	-1.65	-1.95	-1.44	0.19	-2.01	-1.79	-1.23
150×50	-0.43	-0.57	-0.43	-0.61	-0.80	-0.58	1.05	0.38	-0.01	-0.22
250×10	0.15	0.00	0.00	-1.03	-0.49	-0.01	-1.03	-3.00	-1.47	-0.76
250×30	-4.15	0.00	0.00	-5.96	-4.43	-2.17	-2.80	-5.57	-4.51	-3.29
250×50	-4.84	-2.15	0.00	-4.35	-3.66	-2.51	-1.07	-3.62	-3.90	-2.90
350×10	-0.04	0.00	0.00	-0.86	-0.30	-0.01	-1.44	-1.86	-0.66	-0.57
350×30	-3.75	0.00	0.00	-7.08	-3.07	-1.58	-4.66	-6.42	-5.35	-3.55
350×50	-5.92	-1.08	0.00	-7.02	-4.33	-3.19	-3.47	-6.31	-5.60	-4.10
average	-1.57	-0.21	0.03	-2.29	-1.39	-0.84	-0.94	-2.45	-1.89	-1.28

Table 3.4: Average RDI for the regular tests.

Next we compare the results of our tests with published results of the same benchmark.

In [VR10] the authors presented three genetic algorithms. The methods are coded in Delphi 2007 and tested on a Pentium IV 3.0 GHz with 1 GB of main memory. In their final analysis the methods with the best performance were two genetic algorithms, GAPR and GADV, which were tested 20 times for each instance with a CPU time of $n.m.60$ milliseconds. GAPR is a GA that replaces the traditional crossover by a path relinking strategy, GADV is a GA with diversification, and both are hybridized with local search. In the instances where the optimal solution is zero, both GAPR and GADV found the optimal value in all runs, so the corresponding the RPD was replaced by zero. In our implementation: FTS did not find the optimal zero solution

T		0.2			0.4			0.6		
R	0.2	0.6	1	0.2	0.6	1	0.2	0.6	1	*average*
50×10	3.13	0.00	0.00	0.60	0.62	0.77	0.46	0.02	0.56	0.68
50×30	1.11	1.63	0.73	0.63	0.75	0.39	0.43	0.31	0.44	0.71
50×50	1.10	0.82	0.45	0.56	0.60	0.49	0.48	0.26	0.31	0.56
150×10	0.12	0.00	0.00	-2.37	-2.53	-9.87	-0.68	-2.64	-3.39	-2.65
150×30	-5.98	-56.46	0.00	-1.79	-2.71	-2.67	0.11	-1.07	-1.23	-7.98
150×50	-0.72	-2.78	15.50	-0.43	-0.65	-0.57	0.43	0.16	0.00	1.22
250×10	1.45	0.00	0.00	-1.87	-2.95	-8.27	-0.91	-3.95	-3.67	-2.24
250×30	-13.85	0.00	0.00	-6.27	-10.19	-12.20	-1.57	-3.32	-4.07	-5.72
250×50	-9.33	-30.58	0.00	-3.17	-4.46	-3.86	-0.47	-1.53	-2.20	-6.18
350×10	-0.40	0.00	0.00	-1.54	-2.34	-6.97	-1.12	-2.28	-1.72	-1.82
350×30	-13.61	0.00	0.00	-7.19	-8.76	-16.83	-2.48	-4.04	-5.53	-6.49
350×50	-13.91	-43.14	0.00	-5.67	-6.99	-9.42	-1.49	-2.93	-3.98	-10.92
average	-4.24	-11.77	1.39	-2.38	-3.30	-5.96	-0.57	-1.75	-2.04	-3.40

Table 3.5: Average RPD for the regular tests.

for five instances, so five instances are not evaluated in the calculations. Summarizing the results, the mean RPD obtained by GADV was 15.60, GAPR 10.51, FTS 2.65 and our TS obtained a mean RPD of -1.78. We present the comparison of the results in table 3.6

	GADV	GAPR	FTS	TS
50×10	1.75	2.03	1.99	0.68
50×30	1.51	1.61	1.67	0.71
50×50	1.01	1.08	1.14	0.56
150×10	3.55	3.81	2.38	-2.65
150×30	12.40	14.32	2.24	-7.98
150×50	7.96	8.29	8.80	1.22
250×10	5.14	4.88	2.77	-2.24
250×30	38.12	35.25	-0.17	-5.72
250×50	3.38	2.99	-0.30	-6.18
350×10	95.00	39.29	9.59	-1.82
350×30	3.65	3.27	1.99	-6.49
350×50	13.69	9.25	-0.29	-10.92
average	15.6	10.51	2.65	-3.40

Table 3.6: Comparison with the results of [VR10]

In [VR09] the study develops in a parallel environment with an island model; the

algorithms were implemented in Delphi 2006 and tested in machines with Intel Core 2 Duo, 2.4 GHz processors and 1GB RAM. The implemented algorithms were run five times for each instance with a cpu time of $n.m.45$ milliseconds. Due to large time required to evaluate the hole benchmark set, the authors restricted the tests to the instance group where $T = 0.6$ and $R = 0.2$. The serial algorithm with the best performance was SGALS_T, a steady state genetic algorithm with local search running as a single process. CGA_PR(4) and CGA_PR(8) are cooperative genetic algorithm with path relinking running in an island model with respectively 4 and 8 processors. CGA_PR2(12) is a cooperative genetic algorithm with path relinking used as a diversification strategy running in an island model with 12 processors, that showed the best results among the cooperative methods. SGALS_T obtained an RPD of 2.19 and CGA_PR2 an RPD of 0.86. For the same subset of problems our TS obtained an RPD of 0.07. The results are presented in table 3.7

	SGALS_T	CGA_PR(4)	CGA_PR(8)	CGA_PR2(12)	FTS	TS
50×10	2.39	1.68	1.36	0.93	1.26	0.46
50×30	1.51	0.99	0.80	0.53	1.15	0.43
50×50	1.06	0.66	0.59	0.47	0.83	0.48
150×10	2.80	1.90	1.51	1.02	1.90	-0.68
150×30	2.30	1.67	1.13	1.24	1.79	0.11
150×50	1.52	1.07	0.76	0.70	1.62	0.43
250×10	2.79	1.71	1.13	0.92	2.21	-0.91
250×30	2.57	1.47	1.02	0.98	1.48	-1.57
250×50	2.22	1.32	1.18	1.01	1.50	-0.47
350×10	2.51	1.47	1.03	0.72	2.25	-1.12
350×30	2.50	1.55	0.91	0.96	1.19	-2.48
350×50	2.12	1.31	0.83	0.87	1.30	-1.49
average	2.19	1.40	1.02	0.86	1.54	-0.57

Table 3.7: Comparison with the results of [VR09]

In [KTW10] the authors present a elite guided steady-state GA (ssGA-Elite) that outperforms their implementation of GAPR in a simulation study. The algorithms were coded in C++ and tested on a Pentium IV 3.0 GHz with 1 GB RAM. Each instance was tested five times with cpu of $n.m.30$ milliseconds. The quality measure used was the RDI with the *Best* and *Worst* solutions available in the benchmark's site and no improvement in the best known solutions is reported. In table 3.8 we present the average RDI for the various combinations of T and R. The considered algorithms are ssGA-Elite, FTS and TS.

T	0.2			0.4			0.6		
R	0.2	0.6	1	0.2	0.6	1	0.2	0.6	1
ssGA-Elite	5.81	2.18	0.79	7.06	5.20	4.17	7.72	7.69	7.01
FTS	0.81	0.33	0.19	0.77	0.89	0.68	2.67	1.39	1.17
TS	-1.57	-0.21	0.03	-2.29	-1.39	-0.84	-0.94	-2.45	-1.89

Table 3.8: Comparison with the results of [KTW10]

It is clear that both TS and FTS outperform ssGA-Elite in all groups. We should remark that by it's construction ssGA-Elite cannot take benefit from a speed up like the one presented in section 2.3; this fact should have a severe impact in the algorithm's performance.

We also observed that both the RPD and RDI measures have some shortcomings: the RPD loses precision when best known solution of the considered instance are near zero; in a similar way the denominator of the RDI can take arbitrary large values as some of the $Worst$ values are considerably larger than simple heuristics as the EDD, as a result the difference in the numerator is practically neglected in the final result. An interesting alternative is to replace the $Worst$ value in the RDI with the value of the starting solution NEH_{sol} as described in equation 3.3. This adapted measure should provide a clear source for interpretation of the results. Nevertheless, the problem of the division by zero remains, as in some easier instances the NEH_{EDD} is able to find the optimal solution. In those cases we could replace the adapted RDI by zero as no error is committed (or simply remove the instances from the final results). Results for this adaptation are presented in tables 3.9 and 3.10 for the fast and regular runs.

$$RDI_{adapt} = \frac{Method_{sol} - Best}{NEH_{sol} - Best} \times 100 \tag{3.3}$$

An inconvenience of the aforementioned evaluation measures is that the results are sensible to the evolution of the best/worst known solutions, so the results can quickly become incomparable. To provide results that overcome this situation we use a fourth alternative, measuring the error in relation to the original solution. The metric is described in 3.4 and takes the value 100 when no improvement over the initial solution is obtained and zero when the zero objective value is obtained.

$$e_{NEH} = \frac{Method_{sol}}{NEH_{sol}} \times 100 \tag{3.4}$$

The values of e_{NEH} are present in Table 3.11. We can observe that the obtained results are higher for higher values of T and higher for smaller values of R, within the groups. The value also appears to increase as the number of machines increases.

T	0.2			0.4			0.6			
R	0.2	0.6	1	0.2	0.6	1	0.2	0.6	1	*average*
50×10	4.28	0.00	0.00	6.42	5.34	5.84	6.84	6.29	7.57	4.73
50×30	7.37	7.00	4.62	10.19	10.18	8.07	14.92	13.94	13.17	9.94
50×50	8.63	9.41	6.33	12.77	14.99	12.42	16.27	16.18	10.66	11.96
150×10	4.75	0.00	0.00	2.49	5.33	1.73	11.92	5.90	3.99	4.01
150×30	0.94	0.00	0.00	4.75	4.04	2.58	16.42	4.20	5.63	4.28
150×50	4.77	3.75	1.99	8.31	6.58	7.33	21.2	14.52	16.32	9.42
250×10	4.83	0.00	0.00	7.68	6.66	0.09	15.13	7.79	7.33	5.50
250×30	0.48	0.08	0.00	-4.32	-3.40	-0.39	13.94	1.29	-0.81	0.76
250×50	-2.71	-1.01	0.00	0.52	-1.56	3.10	17.83	4.09	0.12	2.26
350×10	6.00	0.00	0.00	6.49	8.25	1.17	20.37	18.05	21.77	9.12
350×30	4.52	0.00	0.00	-0.64	5.24	3.95	12.06	6.31	1.17	3.62
350×50	-0.93	-0.33	0.00	-3.10	-1.70	0.25	16.09	3.05	-4.16	1.02
average	3.58	1.57	1.08	4.30	5.00	3.85	15.25	8.47	6.90	5.55

Table 3.9: Average RDI_{mod} for the fast tests.

T	0.2			0.4			0.6			
R	0.2	0.6	1	0.2	0.6	1	0.2	0.6	1	*average*
50×10	1.76	0.00	0.00	2.00	1.29	1.85	2.49	-0.11	2.34	1.29
50×30	2.73	3.62	2.15	4.65	5.54	2.97	5.47	4.71	5.57	4.16
50×50	4.38	4.39	2.59	6.50	7.51	6.58	9.34	6.50	4.71	5.83
150×10	0.12	0.00	0.00	-7.27	-2.56	-3.81	-4.35	-13.06	-10.07	-4.56
150×30	-7.43	-1.97	0.00	-6.80	-8.01	-7.23	0.85	-9.87	-9.36	-5.54
150×50	-1.50	-2.45	-2.44	-2.49	-3.14	-3.15	5.48	2.07	0.09	-0.84
250×10	0.53	0.00	0.00	-7.01	-4.03	-0.17	-6.41	-18.59	-11.48	-5.24
250×30	-15.99	0.00	0.00	-25.74	-21.91	-11.89	-14.89	-29.91	-25.07	-16.16
250×50	-16.14	-9.36	0.00	-18.75	-17.19	-12.48	-5.55	-19.66	-20.31	-13.27
350×10	-0.32	0.00	0.00	-6.37	-2.77	-0.17	-10.59	-13.27	-5.34	-4.31
350×30	-14.9	0.00	0.00	-34.48	-15.96	-10.19	-25.69	-39.59	-31.41	-19.14
350×50	-22.29	-6.65	0.00	-34.41	-21.35	-18.12	-18.6	-34.49	-33.54	-21.05
average	-5.75	-1.03	0.19	-10.85	-6.88	-4.65	-5.20	-13.77	-11.16	-6.57

Table 3.10: Average RDI_{mod} for the regular tests.

T		0.2			0.4			0.6		
R	0.2	0.6	1	0.2	0.6	1	0.2	0.6	1	*average*
50×10	39.03	0.00	0.00	76.50	66.58	68.84	84.68	86.12	81.38	55.90
50×30	70.79	68.15	71.64	89.03	88.43	88.45	93.01	94.13	93.11	84.08
50×50	81.21	84.15	84.14	92.27	93.01	93.35	95.54	96.17	93.87	90.41
150×10	43.12	0.00	0.00	73.44	47.77	21.60	85.78	80.76	72.15	47.18
150×30	52.14	3.00	0.00	76.66	71.46	71.65	90.04	89.09	87.20	60.14
150×50	66.50	46.77	25.79	85.08	82.57	82.28	93.24	93.06	92.10	74.15
250×10	46.05	0.00	0.00	77.17	53.92	0.24	86.56	78.94	73.61	46.28
250×30	46.10	0.00	0.00	75.17	61.24	43.16	89.04	87.05	82.46	53.80
250×50	57.41	17.08	0.00	82.80	75.98	72.95	91.76	91.31	88.10	64.15
350×10	49.39	0.00	0.00	79.07	52.73	0.31	89.17	82.52	73.99	47.46
350×30	44.94	0.00	0.00	76.78	58.82	30.96	88.89	87.04	79.94	51.93
350×50	52.88	4.84	0.00	80.91	69.76	59.47	91.15	89.41	85.71	59.35
average	54.13	18.67	15.13	80.41	68.52	52.77	89.91	87.97	83.64	61.24

Table 3.11: Average e_{NEH} for the regular tests.

Overall, the quality of the proposed algorithm is evident: of the 540 benchmark instances analyzed, 85 are known to have an optimal (0) solution, of the remaining 455 we were able to improve the best known upper bound in 345. The advantages of using a local search procedure as Tabu Search for exploring the instances' landscape appears evident. Probably other methods would also benefit from hybridization with a more robust local search.

We also emphasize that the Permutation Flow shop Problem with Total Tardiness evaluation criterion is extremely hard to solve. As an example, we tested the integer programming formulation for the first instances of the groups with 50 and 150 jobs with the modern commercial solver Gurobi (*http://www.gurobi.com/*). The solver's CPU time was set to an hour, with the default parameter configuration. Results are presented in table 3.12. *Instance* is the selected problem instance, *ub* and *lb* respectively the upper and lower bound obtained by the solver, FTS and TS the average total tardiness obtained by the developed algorithms. When * is present the solver was not able to find a solution. It's interesting to observe that in most of the 150×30 and 150×50 instances the allowed time is not sufficient to solve the linear relaxation but in some cases the solver is able to provide a feasible solution (probably derived from an heuristic). The obtained lower bounds also appear to be weak, due to the large difference to the results obtained by TS.

Instance	ub	lb	FTS	TS
0,2_0,2_50_10_1	3417	1554	1990.8	1906.4
0,2_0,2_50_30_1	32677	5974	20142.2	19884.8
0,2_0,2_50_50_1	*	7149	35222.4	34926.4
0,2_0,2_150_10_1	24542	9372	12226.8	11688.2
0,2_0,2_150_30_1	*	*	48603.8	44818.0
0,2_0,2_150_50_1	*	*	108664.6	104119.2
0,2_0,6_50_10_1	601	0	0.0	0.0
0,2_0,6_50_30_1	47771	88	12798.0	12634.4
0,2_0,6_50_50_1	60287	846	38176.2	37781.6
0,2_0,6_150_10_1	22274	0	0.0	0.0
0,2_0,6_150_30_1	*	0	216.0	61.4
0,2_0,6_150_50_1	*	*	61712.0	58344.4
0,2_1_50_10_1	1160	0	0.0	0.0
0,2_1_50_30_1	51361	0	10119.0	9881.0
0,2_1_50_50_1	*	0	28761.2	28344.0
0,2_1_150_10_1	68911	0	0.0	0.0
0,2_1_150_30_1	*	0	0.0	0.0
0,2_1_150_50_1	*	*	419.6	208.8
0,4_0,2_50_10_1	15113	9685	12632.8	12401.0
0,4_0,2_50_30_1	67304	26341	50396.8	49965.4
0,4_0,2_50_50_1	102233	39884	79058.4	78934.0
0,4_0,2_150_10_1	124038	62064	75478.6	72935.0
0,4_0,2_150_30_1	*	*	163100.8	158071.0
0,4_0,2_150_50_1	*	*	289821.2	285155.0
0,4_0,6_50_10_1	16595	5532	13290.8	12934.2
0,4_0,6_50_30_1	59007	19707	45189.2	44934.8
0,4_0,6_50_50_1	98198	39233	78775.0	78107.0
0,4_0,6_150_10_1	72387	11549	16199.6	14922.2
0,4_0,6_150_30_1	*	*	166866.4	160808.2
0,4_0,6_150_50_1	*	*	287363.6	282891.8
0,4_1_50_10_1	11571	0	8667.0	8523.0
0,4_1_50_30_1	71774	27625	57608.6	57264.0
0,4_1_50_50_1	95939	32369	73867.2	73440.0
0,4_1_150_10_1	98458	0	4000.4	2996.4
0,4_1_150_30_1	390611	0	126796.2	121749.6
0,4_1_150_50_1	490783	*	218022.2	212433.0
0,6_0,2_50_10_1	35055	28167	32468.2	32313.4
0,6_0,2_50_30_1	96185	55295	80988.8	80252.6
0,6_0,2_50_50_1	146602	85132	121950.0	121496.0
0,6_0,2_150_10_1	253127	172733	201106.4	195525.0
0,6_0,2_150_30_1	*	*	369856.4	364146.8
0,6_0,2_150_50_1	*	*	516077.4	512253.6
0,6_0,6_50_10_1	31477	21344	28953.8	28759.2
0,6_0,6_50_30_1	97258	58119	83213.6	82804.4
0,6_0,6_50_50_1	154159	96556	135733.2	135311.4
0,6_0,6_150_10_1	225404	137711	188192.4	179625.4
0,6_0,6_150_30_1	594939	*	358268.4	353620.0
0,6_0,6_150_50_1	708360	*	511775.4	507309.4
0,6_1_50_10_1	27441	12317	23447.2	23192.8
0,6_1_50_30_1	94816	52081	80000.8	79545.2
0,6_1_50_50_1	124615	72739	108662.8	108206.2
0,6_1_150_10_1	153420	40921	103891.0	97095.4
0,6_1_150_30_1	505894	*	295615.0	290017.8
0,6_1_150_50_1	665697	*	436079.6	430447.6

Table 3.12: Comparison with the results obtained by the Gurobi solver.

Chapter 4

Conclusions

In this work we presented a tabu search metaheuristic for the permutation flowshop problem with total tardiness objective. The method is characterized by the evaluation of a small neighborhood at each iteration, a dynamic tabu list and dynamic parameters. This last feature allows the algorithm's parameters to vary during execution and prevents overfitting to the test instances used in the initial calibration. Tests conducted with recent literature benchmarks confirm the quality of the method. We were able to outperform literature results with one tenth of the running time regularly used in the reported experiments, and improve the best known solution for a large number of instances in the normal runs, when using equivalent CPU. In future work we will apply the developed method to other problems and experiment different diversification and intensification strategies.

Appendix A

Detailed Results

In the following tables we describe the results of the developed Tabu Search for the possible combination of the T and R parameters. The following information is available:

- *Instance*: the instance tested;
- *Best*: the best known solution from *http://soa.iti.es*;
- *Worst*: the worst solution provided at *http://soa.iti.es*;
- *NEH*: the objective value of the initial heuristic;
- *runs* 1 *to* 5: the obtained results for each run of the algorithm.

Instance	Best	Worst	NEH	run1	run2	run3	run4	run5
I_0,2_0,2_50_10_1	1876	23712	5883	1939	1881	1876	1964	1872
I_0,2_0,2_50_10_2	2202	17290	4811	2255	2255	2255	2255	2255
I_0,2_0,2_50_10_3	2564	18479	7878	2878	2818	2686	2748	2831
I_0,2_0,2_50_10_4	2425	17602	7324	2475	2465	2454	2432	2503
I_0,2_0,2_50_10_5	3056	18746	6604	3135	3115	3030	3057	3101
I_0,2_0,2_50_30_1	19475	47599	26390	19938	19788	19875	19897	19926
I_0,2_0,2_50_30_2	20223	49129	28069	20091	20258	20199	20222	20498
I_0,2_0,2_50_30_3	18701	50315	26731	19158	19148	19392	18827	18995
I_0,2_0,2_50_30_4	19379	51674	28057	19521	19484	19403	19579	19513
I_0,2_0,2_50_30_5	19006	52737	29163	19137	19067	19189	19003	19112
I_0,2_0,2_50_50_1	34368	70188	44328	34971	34722	34898	35029	35012
I_0,2_0,2_50_50_2	36651	78240	45249	37092	37220	37303	36731	37066
I_0,2_0,2_50_50_3	36456	72199	45311	36663	36929	37055	37115	37004
I_0,2_0,2_50_50_4	40150	70384	48697	40525	40289	40249	40325	40108
I_0,2_0,2_50_50_5	30477	60931	38002	30713	30807	30894	30643	30806
I_0,2_0,2_150_10_1	11797	78491	28494	11684	11611	11727	11669	11750
I_0,2_0,2_150_10_2	11727	94910	24644	11666	11876	11826	11660	11768
I_0,2_0,2_150_10_3	11480	96730	26994	11635	11457	11184	11261	11164
I_0,2_0,2_150_10_4	12706	100295	31631	12639	12913	12680	12736	12513
I_0,2_0,2_150_10_5	11107	109474	25452	11548	10895	11577	11472	11444
I_0,2_0,2_150_30_1	47978	198124	86055	44180	44917	44624	45600	44769
I_0,2_0,2_150_30_2	49892	180614	99166	46901	46677	46254	46918	46563
I_0,2_0,2_150_30_3	47882	181826	84241	43366	45836	45075	45482	46087
I_0,2_0,2_150_30_4	48913	180934	80520	47605	46654	46395	45956	45825
I_0,2_0,2_150_30_5	47574	195205	88870	45016	44641	43331	45498	44576
I_0,2_0,2_150_50_1	105825	266518	153000	104515	106183	103907	102839	103152
I_0,2_0,2_150_50_2	86843	260227	130595	86415	86180	86209	86842	86965
I_0,2_0,2_150_50_3	114580	270128	166041	114964	115398	114480	114915	113275
I_0,2_0,2_150_50_4	96298	262860	150556	95208	95961	95966	97369	97489
I_0,2_0,2_150_50_5	95546	259961	144147	94036	92957	93872	94109	94529
I_0,2_0,2_250_10_1	25342	231314	62721	25927	25848	25297	25678	25767
I_0,2_0,2_250_10_2	27091	210819	58155	25872	26606	25927	26448	27015
I_0,2_0,2_250_10_3	21093	204808	63839	23116	23435	22319	22700	23074
I_0,2_0,2_250_10_4	21740	178954	38069	21684	21602	21709	21692	21684
I_0,2_0,2_250_10_5	22983	177430	45026	22908	22870	22830	23451	22965
I_0,2_0,2_250_30_1	86952	369904	169362	73061	72374	74339	75612	73586
I_0,2_0,2_250_30_2	74892	327654	136183	63265	64063	64315	64852	64882
I_0,2_0,2_250_30_3	85247	352604	160729	75054	71175	71298	74689	71923
I_0,2_0,2_250_30_4	90429	366319	170352	79981	82389	81919	82791	79649
I_0,2_0,2_250_30_5	88027	427536	159739	73945	76953	74590	73750	73414
I_0,2_0,2_250_50_1	185323	530460	289849	170954	171151	168569	169085	170755
I_0,2_0,2_250_50_2	190699	513084	302225	173306	176621	177364	172780	176518
I_0,2_0,2_250_50_3	181181	521587	282436	163051	162742	162782	163629	161952
I_0,2_0,2_250_50_4	149165	488095	249403	133645	134757	133179	132941	133135
I_0,2_0,2_250_50_5	182313	538042	277970	165959	163236	165002	163813	164461
I_0,2_0,2_350_10_1	46659	431957	87368	47538	46884	46439	46256	45805
I_0,2_0,2_350_10_2	30425	360053	68512	29929	29346	30252	30036	29753
I_0,2_0,2_350_10_3	47906	370816	99183	47664	46638	47964	47338	48073
I_0,2_0,2_350_10_4	38998	341837	77351	39075	39219	38643	38993	40082
I_0,2_0,2_350_10_5	53116	409730	103491	52504	54375	52791	53528	53048
I_0,2_0,2_350_30_1	144749	631932	274658	122881	126538	124783	128163	125729
I_0,2_0,2_350_30_2	155430	627626	289646	131056	130782	133262	129591	128266
I_0,2_0,2_350_30_3	138584	601559	252453	119467	119629	118990	121590	118600
I_0,2_0,2_350_30_4	110525	644747	227890	93669	91590	98341	98085	97321
I_0,2_0,2_350_30_5	125923	677262	249645	111603	113506	109499	111918	108300
I_0,2_0,2_350_50_1	262910	982825	441742	217541	224135	222510	229978	222734
I_0,2_0,2_350_50_2	271466	827292	421153	238580	227222	240110	234571	236155
I_0,2_0,2_350_50_3	215112	801709	372442	186598	192979	187409	189736	187348
I_0,2_0,2_350_50_4	292824	878444	452490	251539	254987	254626	254447	256310
I_0,2_0,2_350_50_5	239300	819237	394705	200204	199067	202365	202364	202827

Table A.1: Individual results for the instances with $T = 0.2$ and $R = 0.2$.

Instance	Best	Worst	NEH	run1	run2	run3	run4	run5
I_0,2_0,6_50_10_1	0	12839	1452	0	0	0	0	0
I_0,2_0,6_50_10_2	0	16324	1149	0	0	0	0	0
I_0,2_0,6_50_10_3	0	15901	3819	0	0	0	0	0
I_0,2_0,6_50_10_4	0	11776	1242	0	0	0	0	0
I_0,2_0,6_50_10_5	0	15426	2112	0	0	0	0	0
I_0,2_0,6_50_30_1	12325	37372	18796	12686	12577	12633	12643	12633
I_0,2_0,6_50_30_2	10763	37373	18285	11185	10796	11050	10956	11162
I_0,2_0,6_50_30_3	16258	44618	24592	16227	16392	16412	16420	16217
I_0,2_0,6_50_30_4	18977	52444	24867	19537	19538	19592	19477	19474
I_0,2_0,6_50_30_5	21734	51558	31774	21643	21711	21707	21645	21754
I_0,2_0,6_50_50_1	37528	73045	43455	37798	37905	37823	37700	37682
I_0,2_0,6_50_50_2	28642	64313	38109	28965	28827	28753	28937	28969
I_0,2_0,6_50_50_3	34361	66054	41036	34609	34645	34528	34714	34625
I_0,2_0,6_50_50_4	41712	83484	47409	41945	42195	41849	42123	42013
I_0,2_0,6_50_50_5	42073	76945	50038	42610	42469	42520	42352	42649
I_0,2_0,6_150_10_1	0	87589	2246	0	0	0	0	0
I_0,2_0,6_150_10_2	0	81506	2150	0	0	0	0	0
I_0,2_0,6_150_10_3	0	80066	4387	0	0	0	0	0
I_0,2_0,6_150_10_4	0	89737	701	0	0	0	0	0
I_0,2_0,6_150_10_5	0	82183	8698	0	0	0	0	0
I_0,2_0,6_150_30_1	147	150157	22006	35	88	47	75	62
I_0,2_0,6_150_30_2	356	181014	18334	106	112	128	97	120
I_0,2_0,6_150_30_3	3026	183696	35270	2062	2119	1764	1899	2032
I_0,2_0,6_150_30_4	311	167000	28189	47	29	0	4	35
I_0,2_0,6_150_30_5	4874	172061	41354	3593	3613	3521	3369	3355
I_0,2_0,6_150_50_1	59354	321792	113776	60279	56797	58751	56501	59394
I_0,2_0,6_150_50_2	36667	245721	87579	35126	35033	34566	35608	35088
I_0,2_0,6_150_50_3	46185	276083	101252	43728	45704	42631	43582	42952
I_0,2_0,6_150_50_4	76961	332651	132951	75164	76083	75539	76244	76274
I_0,2_0,6_150_50_5	43705	277098	102244	42218	43566	43650	42807	43827
I_0,2_0,6_250_10_1	0	205740	774	0	0	0	0	0
I_0,2_0,6_250_10_2	0	215423	1427	0	0	0	0	0
I_0,2_0,6_250_10_3	0	220854	1271	0	0	0	0	0
I_0,2_0,6_250_10_4	0	176256	0	0	0	0	0	0
I_0,2_0,6_250_10_5	0	228906	155	0	0	0	0	0
I_0,2_0,6_250_30_1	0	421148	44113	0	0	0	0	0
I_0,2_0,6_250_30_2	0	475708	65665	0	0	0	0	0
I_0,2_0,6_250_30_3	0	365053	48704	0	0	0	0	0
I_0,2_0,6_250_30_4	0	335205	42337	0	0	0	0	0
I_0,2_0,6_250_30_5	0	361940	51283	0	0	0	0	0
I_0,2_0,6_250_50_1	37578	528689	169569	26690	24777	24356	25344	25319
I_0,2_0,6_250_50_2	39064	593749	154135	27833	25910	26540	30347	24759
I_0,2_0,6_250_50_3	40105	535353	146257	28117	26954	26946	25363	28562
I_0,2_0,6_250_50_4	16991	571229	131624	10652	10482	9597	10337	11480
I_0,2_0,6_250_50_5	66586	623084	204369	54021	53608	54319	53975	53162
I_0,2_0,6_350_10_1	0	310138	0	0	0	0	0	0
I_0,2_0,6_350_10_2	0	367439	0	0	0	0	0	0
I_0,2_0,6_350_10_3	0	339965	0	0	0	0	0	0
I_0,2_0,6_350_10_4	0	343046	0	0	0	0	0	0
I_0,2_0,6_350_10_5	0	432939	236	0	0	0	0	0
I_0,2_0,6_350_30_1	0	617640	49379	0	0	0	0	0
I_0,2_0,6_350_30_2	0	526825	29676	0	0	0	0	0
I_0,2_0,6_350_30_3	0	700013	55089	0	0	0	0	0
I_0,2_0,6_350_30_4	0	632606	43196	0	0	0	0	0
I_0,2_0,6_350_30_5	0	624905	45069	0	0	0	0	0
I_0,2_0,6_350_50_1	18651	798075	148561	10729	9731	9250	10047	8657
I_0,2_0,6_350_50_2	19076	888818	144754	7227	6594	7250	6842	7971
I_0,2_0,6_350_50_3	18689	836431	170315	9962	8460	8702	9323	10640
I_0,2_0,6_350_50_4	0	723240	92184	0	0	0	0	0
I_0,2_0,6_350_50_5	28824	935621	177512	11114	12836	12047	13252	14707

Table A.2: Individual results for the instances with $T = 0.2$ and $R = 0.6$.

Instance	Best	Worst	NEH	run1	run2	run3	run4	run5
I_0,2_1_50_10_1	0	17990	0	0	0	0	0	0
I_0,2_1_50_10_2	0	15143	0	0	0	0	0	0
I_0,2_1_50_10_3	0	14548	0	0	0	0	0	0
I_0,2_1_50_10_4	0	17714	0	0	0	0	0	0
I_0,2_1_50_10_5	0	21210	2748	0	0	0	0	0
I_0,2_1_50_30_1	9849	39142	16870	9889	9891	9849	9891	9885
I_0,2_1_50_30_2	15478	46369	21577	15672	15650	15730	15733	15766
I_0,2_1_50_30_3	18933	47912	23098	19167	19088	19131	19067	19067
I_0,2_1_50_30_4	11830	44878	16176	11860	11921	11916	11847	11854
I_0,2_1_50_30_5	13839	48686	19677	14012	13839	13941	13839	13919
I_0,2_1_50_50_1	28314	63437	35352	28314	28443	28314	28335	28314
I_0,2_1_50_50_2	34288	69385	40347	34433	34471	34386	34469	34462
I_0,2_1_50_50_3	40409	75858	47261	40673	40848	40572	40682	40902
I_0,2_1_50_50_4	36203	72112	45268	36418	36417	36215	36213	36501
I_0,2_1_50_50_5	38744	74799	43913	38925	38785	38930	39075	38920
I_0,2_1_150_10_1	0	110461	0	0	0	0	0	0
I_0,2_1_150_10_2	0	103435	0	0	0	0	0	0
I_0,2_1_150_10_3	0	107620	0	0	0	0	0	0
I_0,2_1_150_10_4	0	108165	0	0	0	0	0	0
I_0,2_1_150_10_5	0	114979	0	0	0	0	0	0
I_0,2_1_150_30_1	0	224099	5697	0	0	0	0	0
I_0,2_1_150_30_2	0	216782	22611	0	0	0	0	0
I_0,2_1_150_30_3	0	205017	16085	0	0	0	0	0
I_0,2_1_150_30_4	0	173766	0	0	0	0	0	0
I_0,2_1_150_30_5	0	211354	26607	0	0	0	0	0
I_0,2_1_150_50_1	104	268284	35055	215	326	253	126	124
I_0,2_1_150_50_2	25898	291400	64491	23115	23440	24274	23769	23599
I_0,2_1_150_50_3	25805	298456	87863	24312	23313	24094	24344	23552
I_0,2_1_150_50_4	30305	318883	81957	30055	28056	29472	30109	29013
I_0,2_1_150_50_5	23555	321501	78810	21926	23108	23184	22277	22502
I_0,2_1_250_10_1	0	274788	0	0	0	0	0	0
I_0,2_1_250_10_2	0	331706	0	0	0	0	0	0
I_0,2_1_250_10_3	0	270323	0	0	0	0	0	0
I_0,2_1_250_10_4	0	293023	0	0	0	0	0	0
I_0,2_1_250_10_5	0	268106	0	0	0	0	0	0
I_0,2_1_250_30_1	0	430488	0	0	0	0	0	0
I_0,2_1_250_30_2	0	503633	4363	0	0	0	0	0
I_0,2_1_250_30_3	0	450297	0	0	0	0	0	0
I_0,2_1_250_30_4	0	439529	0	0	0	0	0	0
I_0,2_1_250_30_5	0	408469	0	0	0	0	0	0
I_0,2_1_250_50_1	0	673718	44201	0	0	0	0	0
I_0,2_1_250_50_2	0	544049	2689	0	0	0	0	0
I_0,2_1_250_50_3	0	628177	52787	0	0	0	0	0
I_0,2_1_250_50_4	0	679747	11235	0	0	0	0	0
I_0,2_1_250_50_5	0	639995	52516	0	0	0	0	0
I_0,2_1_350_10_1	0	484386	0	0	0	0	0	0
I_0,2_1_350_10_2	0	462932	0	0	0	0	0	0
I_0,2_1_350_10_3	0	542073	0	0	0	0	0	0
I_0,2_1_350_10_4	0	555976	0	0	0	0	0	0
I_0,2_1_350_10_5	0	471445	0	0	0	0	0	0
I_0,2_1_350_30_1	0	748152	0	0	0	0	0	0
I_0,2_1_350_30_2	0	852236	0	0	0	0	0	0
I_0,2_1_350_30_3	0	693495	0	0	0	0	0	0
I_0,2_1_350_30_4	0	718808	0	0	0	0	0	0
I_0,2_1_350_30_5	0	787918	0	0	0	0	0	0
I_0,2_1_350_50_1	0	1198477	39126	0	0	0	0	0
I_0,2_1_350_50_2	0	871316	0	0	0	0	0	0
I_0,2_1_350_50_3	0	987292	28616	0	0	0	0	0
I_0,2_1_350_50_4	0	976823	0	0	0	0	0	0
I_0,2_1_350_50_5	0	981061	0	0	0	0	0	0

Table A.3: Individual results for the instances with $T = 0.2$ and $R = 1$.

Instance	Best	Worst	NEH	run1	run2	run3	run4	run5
I_0,4_0,2_50_10_1	12440	37610	16811	12325	12479	12319	12345	12537
I_0,4_0,2_50_10_2	12246	38317	18154	12364	12418	12316	12542	12264
I_0,4_0,2_50_10_3	13431	35455	16019	13484	13413	13520	13545	13548
I_0,4_0,2_50_10_4	13371	36676	17043	13579	13454	13433	13545	13466
I_0,4_0,2_50_10_5	15666	41246	20482	15710	15762	15866	15809	15777
I_0,4_0,2_50_30_1	49501	86040	55607	50184	49578	49889	50312	49864
I_0,4_0,2_50_30_2	44964	78373	50871	45180	45373	45589	45505	45451
I_0,4_0,2_50_30_3	47698	77497	53331	48150	48120	47894	47994	47693
I_0,4_0,2_50_30_4	44095	75249	51021	44291	44425	44247	45083	44702
I_0,4_0,2_50_30_5	44330	77520	49737	44023	44018	44140	44326	44205
I_0,4_0,2_50_50_1	78504	116088	89285	79156	78971	79034	78983	78526
I_0,4_0,2_50_50_2	76005	112695	81770	76120	76566	75980	76558	76070
I_0,4_0,2_50_50_3	77432	121245	83128	77521	77975	77551	77944	77863
I_0,4_0,2_50_50_4	69271	112466	75851	69888	69527	69737	70024	70011
I_0,4_0,2_50_50_5	81265	123042	86964	81964	81884	81459	81826	81914
I_0,4_0,2_150_10_1	74610	214663	99802	73924	72671	72939	72880	72261
I_0,4_0,2_150_10_2	68060	224570	92253	67083	67033	66569	66953	67593
I_0,4_0,2_150_10_3	68570	195474	90275	67099	67652	66809	66907	66716
I_0,4_0,2_150_10_4	83840	229742	108769	82399	81611	81570	80405	81450
I_0,4_0,2_150_10_5	78297	243266	105001	75790	75113	76551	75472	76558
I_0,4_0,2_150_30_1	160726	344393	208804	157600	158432	159656	156668	157999
I_0,4_0,2_150_30_2	167096	345166	223300	168595	166428	164172	164376	165611
I_0,4_0,2_150_30_3	178896	365357	219686	175745	174282	174546	173993	174777
I_0,4_0,2_150_30_4	181210	384627	227785	174872	175312	177234	173541	174106
I_0,4_0,2_150_30_5	174370	335800	224808	174465	172666	173386	171936	172943
I_0,4_0,2_150_50_1	286315	476649	332744	284121	284489	286619	285315	285231
I_0,4_0,2_150_50_2	277514	498646	330115	276151	278540	276080	275744	277180
I_0,4_0,2_150_50_3	283469	492305	333203	279283	281097	282920	281410	282064
I_0,4_0,2_150_50_4	306611	514098	355256	306111	308062	305498	305296	306970
I_0,4_0,2_150_50_5	293482	485725	342426	294075	292138	290932	292051	289027
I_0,4_0,2_250_10_1	174248	473388	224613	172532	172657	171610	174766	173137
I_0,4_0,2_250_10_2	185652	507837	225149	182781	181780	180717	182675	181796
I_0,4_0,2_250_10_3	176022	523125	225076	172151	172619	174199	172922	174108
I_0,4_0,2_250_10_4	181552	556221	236106	178930	178910	175548	179529	177530
I_0,4_0,2_250_10_5	202280	540373	259155	197588	195437	194507	197740	194946
I_0,4_0,2_250_30_1	328501	702794	423826	309122	310141	311091	308913	314278
I_0,4_0,2_250_30_2	398048	784893	479046	374828	380434	374146	371025	368846
I_0,4_0,2_250_30_3	361399	752581	451264	333830	332690	340463	339982	338848
I_0,4_0,2_250_30_4	343587	707748	431145	322134	323434	321616	318698	317161
I_0,4_0,2_250_30_5	352110	713226	436694	329404	330215	328973	329609	328383
I_0,4_0,2_250_50_1	574459	989756	670647	554288	553915	551234	555632	552660
I_0,4_0,2_250_50_2	529885	925117	628280	510743	511261	507689	506694	510379
I_0,4_0,2_250_50_3	557528	957928	653347	542703	545474	545923	539254	548717
I_0,4_0,2_250_50_4	571225	1002874	671010	554784	553269	559025	550932	554157
I_0,4_0,2_250_50_5	576269	981947	661618	560404	560071	557345	560097	555459
I_0,4_0,2_350_10_1	349812	933840	446812	338964	344163	345080	344324	345167
I_0,4_0,2_350_10_2	352470	1067922	434488	343167	343559	346240	343118	346190
I_0,4_0,2_350_10_3	338175	940785	441832	331646	333391	334844	335687	333613
I_0,4_0,2_350_10_4	379651	954031	461138	378921	378969	374178	376148	379388
I_0,4_0,2_350_10_5	345862	945805	415573	340943	342761	335593	339903	339395
I_0,4_0,2_350_30_1	615029	1343398	758472	569839	564556	570050	573259	567592
I_0,4_0,2_350_30_2	609633	1230165	734810	562879	570739	561401	557258	559306
I_0,4_0,2_350_30_3	650046	1277603	773292	602370	609535	614380	612350	607081
I_0,4_0,2_350_30_4	662634	1295388	803541	624015	615793	614838	616307	618695
I_0,4_0,2_350_30_5	651888	1288047	784830	602223	595797	596591	615329	599366
I_0,4_0,2_350_50_1	917309	1664071	1091924	863279	874983	868497	871527	866825
I_0,4_0,2_350_50_2	857089	1635388	994125	801286	805741	806673	802458	804575
I_0,4_0,2_350_50_3	934951	1674854	1092103	880662	885405	874732	878196	879366
I_0,4_0,2_350_50_4	967417	1677322	1118974	904018	909185	908252	919334	910879
I_0,4_0,2_350_50_5	939405	1697469	1084947	899975	885936	891911	896687	882988

Table A.4: Individual results for the instances with $T = 0.4$ and $R = 0.2$.

Instance	Best	Worst	NEH	run1	run2	run3	run4	run5
I_0,4_0,6_50_10_1	12803	37020	17418	12904	13033	12852	13008	12874
I_0,4_0,6_50_10_2	6575	27566	12154	6598	6653	6645	6611	6622
I_0,4_0,6_50_10_3	9739	30146	14105	9739	9739	9739	9739	9739
I_0,4_0,6_50_10_4	13194	35027	19270	13304	13238	13243	13346	13307
I_0,4_0,6_50_10_5	10625	36276	16157	10678	10748	10664	10664	10685
I_0,4_0,6_50_30_1	44630	72562	52053	44762	44957	44971	44904	45080
I_0,4_0,6_50_30_2	43927	73458	51238	44330	44321	44240	44336	44358
I_0,4_0,6_50_30_3	41842	74353	46450	42184	42101	42097	42224	42149
I_0,4_0,6_50_30_4	45133	78371	49585	45315	45284	45369	45496	45592
I_0,4_0,6_50_30_5	42981	77542	49810	43290	43312	43504	43281	43332
I_0,4_0,6_50_50_1	77609	115099	83228	77825	78085	78278	78182	78165
I_0,4_0,6_50_50_2	79076	110998	84708	79929	79824	79513	79605	79381
I_0,4_0,6_50_50_3	84546	121272	91327	85077	85268	84638	85131	84982
I_0,4_0,6_50_50_4	84872	123673	91409	85270	85048	85096	85279	85244
I_0,4_0,6_50_50_5	69945	103258	77508	70301	70174	70545	70397	70837
I_0,4_0,6_150_10_1	15178	157878	41133	14880	14595	14972	14971	15193
I_0,4_0,6_150_10_2	23359	183459	45973	22674	21902	22202	22202	22149
I_0,4_0,6_150_10_3	29421	198920	56256	27742	27920	27272	27306	28712
I_0,4_0,6_150_10_4	31152	184219	60113	31452	31737	31435	31527	31609
I_0,4_0,6_150_10_5	25000	173228	46892	24908	24517	24625	24240	24415
I_0,4_0,6_150_30_1	165106	378263	216190	163441	160818	157796	161424	160562
I_0,4_0,6_150_30_2	147688	326072	199573	146404	145475	142995	145311	144139
I_0,4_0,6_150_30_3	164307	360565	207354	157660	159920	158788	156527	159918
I_0,4_0,6_150_30_4	130975	323780	176416	126569	126466	126969	128135	128231
I_0,4_0,6_150_30_5	91676	283405	144505	90444	87297	90348	89401	88505
I_0,4_0,6_150_50_1	285036	500241	346602	282755	284020	281676	282935	283073
I_0,4_0,6_150_50_2	244189	463914	299753	241345	244519	239592	243689	242158
I_0,4_0,6_150_50_3	238616	466850	291649	237397	235359	237011	237643	236552
I_0,4_0,6_150_50_4	276152	452533	319758	277086	275517	273648	273642	274538
I_0,4_0,6_150_50_5	263796	475957	315386	261290	264494	261896	262558	262488
I_0,4_0,6_250_10_1	84881	477950	131285	81256	81238	81293	82287	81460
I_0,4_0,6_250_10_2	55435	417834	102121	52739	52061	53759	52133	53055
I_0,4_0,6_250_10_3	56512	465246	118621	55656	55315	56624	53310	54635
I_0,4_0,6_250_10_4	65837	514909	111676	63698	62973	65063	63780	63231
I_0,4_0,6_250_10_5	45659	440122	87017	46118	44992	45925	45825	44802
I_0,4_0,6_250_30_1	245976	748314	350730	220005	219404	222842	221845	220413
I_0,4_0,6_250_30_2	218551	757816	338795	201568	199628	198011	197975	195776
I_0,4_0,6_250_30_3	276265	824945	379680	253397	250075	251637	254487	255772
I_0,4_0,6_250_30_4	208685	795630	317722	183813	184533	182239	183309	183357
I_0,4_0,6_250_30_5	217212	707819	320655	193626	191746	191680	192220	195605
I_0,4_0,6_250_50_1	495156	1041976	621607	478723	474655	478804	471500	468798
I_0,4_0,6_250_50_2	464542	1016157	590112	443309	448731	443668	444570	438976
I_0,4_0,6_250_50_3	484289	1071597	583817	465185	464479	466424	473872	467945
I_0,4_0,6_250_50_4	394318	929416	510423	374321	370038	372241	371438	370388
I_0,4_0,6_250_50_5	453074	1003790	573681	438475	430602	431572	431000	432084
I_0,4_0,6_350_10_1	70159	816426	134122	68542	68428	68731	68007	69196
I_0,4_0,6_350_10_2	135250	990376	252849	130024	132674	132963	137674	132336
I_0,4_0,6_350_10_3	119195	924886	215307	114398	112248	118302	113211	116947
I_0,4_0,6_350_10_4	94824	836207	173376	92523	93982	92395	92037	91011
I_0,4_0,6_350_10_5	75281	788473	139041	73997	73820	73860	73916	73941
I_0,4_0,6_350_30_1	275188	1253296	438180	251602	249077	253476	250550	252530
I_0,4_0,6_350_30_2	305379	1284636	495371	278153	275917	270912	270449	268131
I_0,4_0,6_350_30_3	306374	1249957	489130	280042	280281	295244	281807	286662
I_0,4_0,6_350_30_4	421979	1310128	615684	391693	392117	385052	384816	395977
I_0,4_0,6_350_30_5	352512	1292256	528844	315913	321149	321189	312121	318761
I_0,4_0,6_350_50_1	636032	1574946	868628	590559	608250	597738	600354	599050
I_0,4_0,6_350_50_2	641927	1670527	828873	589017	586207	591737	584359	587851
I_0,4_0,6_350_50_3	664344	1660496	893487	631375	631444	630492	620279	630834
I_0,4_0,6_350_50_4	654106	1832601	860521	596296	607015	605654	606794	590699
I_0,4_0,6_350_50_5	673320	1788567	909174	626555	632301	626320	617103	617779

Table A.5: Individual results for the instances with $T = 0.4$ and $R = 0.6$.

Instance	Best	Worst	NEH	run1	run2	run3	run4	run5
I_0,4_1_50_10_1	8499	33694	10808	8535	8499	8533	8514	8534
I_0,4_1_50_10_2	8208	37102	11864	8372	8208	8238	8229	8254
I_0,4_1_50_10_3	2397	25405	4124	2399	2397	2397	2399	2402
I_0,4_1_50_10_4	7116	32276	12604	7230	7143	7353	7240	7196
I_0,4_1_50_10_5	19526	47690	24646	19732	19729	19823	19810	19704
I_0,4_1_50_30_1	57159	90623	62741	57132	57314	57174	57550	57150
I_0,4_1_50_30_2	53443	90392	60148	53798	53471	53921	53720	53854
I_0,4_1_50_30_3	44839	73827	51589	45196	45202	45085	45034	45184
I_0,4_1_50_30_4	47767	78783	53290	47894	47900	47920	47940	47955
I_0,4_1_50_30_5	40849	68896	48621	40862	41008	40981	40867	40956
I_0,4_1_50_50_1	73137	110040	80116	73321	73557	73498	73492	73332
I_0,4_1_50_50_2	80918	121467	85998	81299	81649	81143	81414	81590
I_0,4_1_50_50_3	81996	115056	87950	82372	82139	82117	82025	81847
I_0,4_1_50_50_4	67534	98399	73992	68189	67824	67997	67814	68418
I_0,4_1_50_50_5	89618	128392	94735	90190	90187	90002	90140	89864
I_0,4_1_150_10_1	3777	215460	29577	3078	2650	2936	2950	3368
I_0,4_1_150_10_2	11540	231415	36284	10721	11135	10840	10624	10067
I_0,4_1_150_10_3	23887	241657	53446	21452	22049	23053	21642	21627
I_0,4_1_150_10_4	0	205585	6831	0	0	0	0	0
I_0,4_1_150_10_5	14904	235131	47298	13069	12813	13200	12956	12684
I_0,4_1_150_30_1	126483	377459	178523	121780	123508	121393	121516	120551
I_0,4_1_150_30_2	114910	357652	164767	113328	111036	108678	109428	109969
I_0,4_1_150_30_3	168363	408862	211491	167281	165416	165789	166041	165399
I_0,4_1_150_30_4	110250	337163	158373	106198	109393	106911	107629	108658
I_0,4_1_150_30_5	155548	383986	199149	152069	151364	153633	153487	150878
I_0,4_1_150_50_1	213902	445887	266053	211185	212414	212773	214558	211235
I_0,4_1_150_50_2	270506	511724	323552	267856	270265	268888	271996	268201
I_0,4_1_150_50_3	254563	497004	288594	252245	251032	251837	251377	253432
I_0,4_1_150_50_4	207909	427569	262388	207396	204880	207478	210016	206558
I_0,4_1_150_50_5	257771	507282	312998	256754	255408	255862	257839	257025
I_0,4_1_250_10_1	0	574682	3258	0	0	0	0	0
I_0,4_1_250_10_2	0	569371	676	0	0	0	0	0
I_0,4_1_250_10_3	0	550596	3199	0	0	0	0	0
I_0,4_1_250_10_4	487	532580	23853	166	464	346	209	243
I_0,4_1_250_10_5	0	524065	2828	0	0	0	0	0
I_0,4_1_250_30_1	89346	753161	192442	81099	81267	80310	80330	78473
I_0,4_1_250_30_2	69611	755430	199380	60949	57764	55595	57876	56257
I_0,4_1_250_30_3	162379	825977	307695	144011	144097	139726	138430	142801
I_0,4_1_250_30_4	103737	809065	213027	91442	93028	92636	93752	93688
I_0,4_1_250_30_5	188176	829173	303290	166115	170102	171479	168021	164887
I_0,4_1_250_50_1	497371	1072043	608800	477064	478594	479421	479001	476969
I_0,4_1_250_50_2	428307	1063575	566625	413363	406797	415592	413889	417024
I_0,4_1_250_50_3	419641	1051571	552506	406743	411007	407906	398613	402378
I_0,4_1_250_50_4	367747	1048571	510512	351289	357465	359685	355347	354904
I_0,4_1_250_50_5	342810	987849	462534	326479	320976	325979	326516	323273
I_0,4_1_350_10_1	0	1107180	10410	0	0	0	0	0
I_0,4_1_350_10_2	0	1160268	1914	0	0	0	0	0
I_0,4_1_350_10_3	0	1006234	880	0	0	0	0	0
I_0,4_1_350_10_4	0	984961	2930	0	0	0	0	0
I_0,4_1_350_10_5	1696	1210101	71250	1642	1153	1055	777	897
I_0,4_1_350_30_1	232344	1536886	413310	199354	195929	200929	193092	199751
I_0,4_1_350_30_2	127685	1455530	332438	101194	98491	102191	99384	106064
I_0,4_1_350_30_3	181957	1420092	413733	170640	168120	160651	163777	160361
I_0,4_1_350_30_4	74992	1395009	348915	56761	60667	59615	61718	62662
I_0,4_1_350_30_5	55787	1477067	231578	46115	40777	46388	44424	46352
I_0,4_1_350_50_1	504134	1791943	715862	461591	459840	469432	462406	461416
I_0,4_1_350_50_2	487355	1802832	706430	445714	429474	441050	427632	436042
I_0,4_1_350_50_3	321590	1611422	569966	286419	281141	285485	285202	279627
I_0,4_1_350_50_4	507325	1804010	742225	468260	462446	459255	467974	477774
I_0,4_1_350_50_5	423172	1729840	663002	390639	386936	389109	382818	383024

Table A.6: Individual results for the instances with $T = 0.4$ and $R = 1$.

Instance	Best	Worst	NEH	run1	run2	run3	run4	run5
I_0,6_0,2_50_10_1	32239	58313	38632	32307	32206	32423	32370	32261
I_0,6_0,2_50_10_2	31288	62168	37593	31602	31422	31653	31378	31344
I_0,6_0,2_50_10_3	31208	62355	36666	31449	31280	31106	31208	31236
I_0,6_0,2_50_10_4	33937	65897	40377	34169	34304	34207	33994	34249
I_0,6_0,2_50_10_5	33744	62439	39413	34008	34046	33861	33994	33811
I_0,6_0,2_50_30_1	80024	115855	85480	80081	80263	80130	80508	80281
I_0,6_0,2_50_30_2	78802	117310	86937	79152	79111	79048	79182	79314
I_0,6_0,2_50_30_3	78004	113342	84357	78188	78088	78231	78377	78337
I_0,6_0,2_50_30_4	75424	113987	81750	76077	75955	75789	75726	76160
I_0,6_0,2_50_30_5	82519	123226	87733	82929	82786	82983	82749	82910
I_0,6_0,2_50_50_1	121013	166909	127429	121275	121560	121672	121227	121746
I_0,6_0,2_50_50_2	131296	181670	137145	131842	132003	132054	132231	131909
I_0,6_0,2_50_50_3	125744	170467	131427	125925	126738	126196	126333	126111
I_0,6_0,2_50_50_4	125255	166468	131919	125906	125427	126002	125891	126049
I_0,6_0,2_50_50_5	120302	163139	127834	120722	120956	121097	121017	121067
I_0,6_0,2_150_10_1	196988	398364	227227	196498	194929	195200	195572	195426
I_0,6_0,2_150_10_2	200711	374676	234101	199347	199852	200135	199631	199089
I_0,6_0,2_150_10_3	204607	364627	236063	202629	204317	204077	203054	203568
I_0,6_0,2_150_10_4	192752	362646	225690	192182	191299	190213	192285	190945
I_0,6_0,2_150_10_5	203703	385257	233231	202992	201906	201590	201176	201902
I_0,6_0,2_150_30_1	364656	564356	404273	365223	364661	363496	363033	364321
I_0,6_0,2_150_30_2	336615	518316	378260	338703	338469	335245	336161	337285
I_0,6_0,2_150_30_3	372948	562370	406214	373731	374040	372333	374343	371469
I_0,6_0,2_150_30_4	357272	565515	394591	357980	355091	356399	357856	360636
I_0,6_0,2_150_30_5	349045	549806	395741	349843	351687	350649	349946	349260
I_0,6_0,2_150_50_1	509024	729909	550424	513452	514777	510417	510842	511780
I_0,6_0,2_150_50_2	524130	737882	559504	526738	530188	525481	526077	525885
I_0,6_0,2_150_50_3	527933	751228	576581	533633	530435	533082	526963	531733
I_0,6_0,2_150_50_4	529795	777607	566735	532548	527785	527652	529246	528982
I_0,6_0,2_150_50_5	542100	758877	582887	543952	543937	543399	547399	545493
I_0,6_0,2_250_10_1	498796	956559	573672	492918	496088	497634	493327	495135
I_0,6_0,2_250_10_2	495493	934932	565804	490353	485803	492160	490175	488819
I_0,6_0,2_250_10_3	516555	966953	596178	512704	511737	516576	508309	514117
I_0,6_0,2_250_10_4	493583	958899	565885	488879	492648	490712	491035	487601
I_0,6_0,2_250_10_5	515899	950996	583406	509559	506619	512823	508815	511723
I_0,6_0,2_250_30_1	815312	1344618	898944	803935	796394	805603	805306	797921
I_0,6_0,2_250_30_2	830108	1290326	920872	822301	822899	819779	817681	816087
I_0,6_0,2_250_30_3	820304	1271781	905211	807487	810501	803844	808229	806207
I_0,6_0,2_250_30_4	825271	1269213	922363	813883	808587	812914	804582	813762
I_0,6_0,2_250_30_5	837421	1269636	916954	817934	827402	824612	825149	826140
I_0,6_0,2_250_50_1	1113981	1615363	1207778	1111148	1103233	1109652	1107014	1106240
I_0,6_0,2_250_50_2	1137566	1642028	1227092	1132718	1137819	1129751	1130297	1132122
I_0,6_0,2_250_50_3	1088911	1561489	1195230	1084457	1081620	1090601	1084034	1086468
I_0,6_0,2_250_50_4	1071931	1521119	1158208	1069893	1067674	1062322	1068723	1073866
I_0,6_0,2_250_50_5	1135474	1616992	1229806	1130928	1127426	1125122	1126467	1129916
I_0,6_0,2_350_10_1	951162	1736522	1057383	935278	930413	939523	933108	933133
I_0,6_0,2_350_10_2	979519	1719161	1076430	970175	970675	970934	972159	967658
I_0,6_0,2_350_10_3	955705	1721446	1072377	947927	950078	953035	955373	944000
I_0,6_0,2_350_10_4	959473	1688177	1048821	943081	948772	944484	947635	944410
I_0,6_0,2_350_10_5	992129	1725112	1109937	972253	993948	983120	977742	990378
I_0,6_0,2_350_30_1	1428199	2207607	1575233	1397459	1402641	1394623	1409570	1397101
I_0,6_0,2_350_30_2	1432854	2255392	1582107	1387131	1378059	1390529	1393829	1389058
I_0,6_0,2_350_30_3	1438431	2182920	1578081	1404323	1414039	1395571	1400893	1410187
I_0,6_0,2_350_30_4	1458543	2189638	1598031	1428482	1422853	1424063	1412410	1423759
I_0,6_0,2_350_30_5	1426237	2173832	1548196	1375595	1397094	1391888	1399460	1388386
I_0,6_0,2_350_50_1	1945815	2698203	2083472	1922071	1920922	1912516	1933812	1920119
I_0,6_0,2_350_50_2	1935135	2771965	2107241	1899403	1909963	1900482	1911626	1907015
I_0,6_0,2_350_50_3	1897063	2756088	2063476	1859241	1868595	1873825	1863483	1874901
I_0,6_0,2_350_50_4	1936200	2734515	2074282	1906228	1895285	1896706	1912551	1899699
I_0,6_0,2_350_50_5	1835320	2691923	1992407	1817908	1796665	1814047	1808264	1811304

Table A.7: Individual results for the instances with $T = 0.6$ and $R = 0.2$.

Instance	Best	Worst	NEH	run1	run2	run3	run4	run5
I_0,6_0,6_50_10_1	28619	51206	34556	28653	28710	28931	28735	28767
I_0,6_0,6_50_10_2	33942	59632	38635	33784	33749	33775	33749	33795
I_0,6_0,6_50_10_3	31571	57358	36655	31694	31465	31556	31497	31630
I_0,6_0,6_50_10_4	41665	75153	47923	41657	41661	41709	41570	42015
I_0,6_0,6_50_10_5	34689	63894	39965	34693	34542	34864	34772	34538
I_0,6_0,6_50_30_1	82636	122406	87053	82583	83006	82703	82860	82870
I_0,6_0,6_50_30_2	79253	107379	83925	79625	79445	79654	79518	79302
I_0,6_0,6_50_30_3	80985	119019	86632	81054	81283	81266	81449	81296
I_0,6_0,6_50_30_4	84554	121543	91082	84897	84722	84613	84462	84881
I_0,6_0,6_50_30_5	76477	111832	81781	76618	76881	76772	76765	77140
I_0,6_0,6_50_50_1	135134	174921	139847	135178	135786	135242	135178	135173
I_0,6_0,6_50_50_2	128467	169723	132570	128923	129031	128693	128520	128922
I_0,6_0,6_50_50_3	129470	173384	136861	129630	129868	129499	129813	130010
I_0,6_0,6_50_50_4	121553	162625	127640	121644	121887	122015	121953	121858
I_0,6_0,6_50_50_5	131517	172507	136665	132053	132114	132034	132145	132059
I_0,6_0,6_150_10_1	185012	361994	224603	179024	179092	180953	179117	179941
I_0,6_0,6_150_10_2	143558	325788	178048	140041	140584	140169	139232	139053
I_0,6_0,6_150_10_3	152484	313771	186404	147602	147842	146668	148179	147195
I_0,6_0,6_150_10_4	197974	385236	227321	191950	193600	193749	192582	193002
I_0,6_0,6_150_10_5	187137	360624	225870	183801	183621	182194	184116	184257
I_0,6_0,6_150_30_1	357587	544947	393131	355236	352869	350688	354534	354773
I_0,6_0,6_150_30_2	370485	578009	410392	363671	365124	365255	367874	365631
I_0,6_0,6_150_30_3	351715	551423	395174	349784	350447	353163	348997	347953
I_0,6_0,6_150_30_4	362360	538802	401990	358331	357209	356183	357053	357302
I_0,6_0,6_150_30_5	350084	548417	389446	346711	347837	345971	345423	346894
I_0,6_0,6_150_50_1	505307	721412	545866	508192	507102	507275	507111	506867
I_0,6_0,6_150_50_2	515928	727214	562430	517475	514457	519058	517992	517214
I_0,6_0,6_150_50_3	535192	743187	563225	535440	537530	534891	535549	534673
I_0,6_0,6_150_50_4	525278	732299	573403	524877	524282	522875	526143	527430
I_0,6_0,6_150_50_5	518183	741633	553541	519466	518473	518645	518531	518294
I_0,6_0,6_250_10_1	411155	917494	495102	388414	393929	390437	389570	391934
I_0,6_0,6_250_10_2	329544	819958	398883	315211	314449	320369	315786	319971
I_0,6_0,6_250_10_3	407861	913502	481105	393242	391580	392616	390579	391658
I_0,6_0,6_250_10_4	355978	817926	437199	343811	343743	344838	342695	342886
I_0,6_0,6_250_10_5	326431	783347	412759	313475	314579	316229	314178	313660
I_0,6_0,6_250_30_1	751804	1277341	837909	724728	728898	729161	724985	728961
I_0,6_0,6_250_30_2	841210	1286364	936777	808835	816300	811484	812645	814079
I_0,6_0,6_250_30_3	856067	1295235	951465	831894	825289	830097	827470	829364
I_0,6_0,6_250_30_4	850246	1344571	938427	829697	824735	823590	830245	826754
I_0,6_0,6_250_30_5	780628	1314501	866873	748416	752316	750432	743417	751768
I_0,6_0,6_250_50_1	1095812	1569067	1198404	1077072	1084800	1074853	1073245	1076150
I_0,6_0,6_250_50_2	1140204	1613926	1219469	1118740	1122831	1117721	1120399	1121146
I_0,6_0,6_250_50_3	1136446	1674900	1226537	1122762	1128018	1124074	1120346	1120641
I_0,6_0,6_250_50_4	1091625	1526942	1166994	1083033	1078392	1077053	1076740	1071235
I_0,6_0,6_250_50_5	1106850	1559500	1196544	1081848	1093409	1086480	1088555	1089114
I_0,6_0,6_350_10_1	782751	1723388	924850	772882	767455	770157	767190	762719
I_0,6_0,6_350_10_2	711434	1630810	852402	695608	697679	687635	698026	690839
I_0,6_0,6_350_10_3	766663	1658597	880581	750363	731836	740107	744899	731664
I_0,6_0,6_350_10_4	780456	1758434	913419	755064	762839	764117	756530	749009
I_0,6_0,6_350_10_5	680830	1598120	833360	679218	676904	681278	676754	668971
I_0,6_0,6_350_30_1	1419244	2260147	1569751	1354362	1360684	1364399	1356514	1364375
I_0,6_0,6_350_30_2	1391842	2289908	1518680	1339508	1334120	1338465	1338104	1352046
I_0,6_0,6_350_30_3	1341378	2168015	1477135	1282371	1288634	1266363	1281290	1276947
I_0,6_0,6_350_30_4	1389860	2289689	1542662	1332128	1347051	1328714	1355317	1340102
I_0,6_0,6_350_30_5	1363329	2262252	1505427	1314404	1297722	1302680	1309567	1307756
I_0,6_0,6_350_50_1	1884667	2795036	2030852	1827152	1840348	1824881	1830398	1835065
I_0,6_0,6_350_50_2	1857069	2688651	2016270	1804171	1796477	1790110	1786018	1787471
I_0,6_0,6_350_50_3	1841663	2757395	2005921	1799260	1778988	1793808	1801218	1795066
I_0,6_0,6_350_50_4	1838435	2634656	2024653	1778139	1800259	1775119	1777882	1788275
I_0,6_0,6_350_50_5	1828421	2697561	1966006	1777669	1781860	1775492	1774754	1775686

Table A.8: Individual results for the instances with $T = 0.6$ and $R = 0.6$.

Instance	Best	Worst	NEH	run1	run2	run3	run4	run5
I_0,6_1_50_10_1	23036	48375	30350	23170	23296	23198	23144	23156
I_0,6_1_50_10_2	23514	45955	28880	23640	23811	23635	23653	23753
I_0,6_1_50_10_3	27859	52257	33356	27998	27969	27967	28032	27926
I_0,6_1_50_10_4	17471	41097	21527	17657	17472	17504	17613	17589
I_0,6_1_50_10_5	22792	50995	27568	22988	22891	22857	22789	22818
I_0,6_1_50_30_1	79209	114405	85937	79649	79496	79412	79282	79887
I_0,6_1_50_30_2	69169	102055	73633	69281	69042	69586	69563	69108
I_0,6_1_50_30_3	69971	102097	75089	70554	70530	70652	70562	70621
I_0,6_1_50_30_4	71469	103335	76973	71842	71486	71673	71345	71638
I_0,6_1_50_30_5	67699	101159	74078	67913	68019	68038	68096	68112
I_0,6_1_50_50_1	107931	149917	118243	108203	108178	108122	108299	108229
I_0,6_1_50_50_2	101528	137881	108364	101907	102027	102029	101948	101806
I_0,6_1_50_50_3	114653	151807	123102	114988	114714	114899	115085	114845
I_0,6_1_50_50_4	119406	152793	124544	119415	119769	119873	119674	119767
I_0,6_1_50_50_5	92716	129356	98707	93021	93054	93047	93081	93208
I_0,6_1_150_10_1	103978	298374	138468	97140	97576	97099	96443	97219
I_0,6_1_150_10_2	91137	302150	125743	89090	88864	89596	88272	88090
I_0,6_1_150_10_3	116150	316146	154585	112836	113324	113502	112813	113998
I_0,6_1_150_10_4	92508	294063	117458	90899	90435	90156	89745	90823
I_0,6_1_150_10_5	103018	295546	143164	99654	99712	100821	99665	99650
I_0,6_1_150_30_1	295179	511389	331891	290391	289308	289681	291130	289579
I_0,6_1_150_30_2	323391	528594	360368	320020	320288	321815	320888	321002
I_0,6_1_150_30_3	303022	493976	348644	299731	300795	298700	298954	299860
I_0,6_1_150_30_4	288861	502971	335759	285772	284850	284031	286050	284607
I_0,6_1_150_30_5	317087	525922	353170	315543	312195	313789	314350	311262
I_0,6_1_150_50_1	429452	630366	464082	432261	430993	429866	429330	429788
I_0,6_1_150_50_2	464109	688927	513794	462079	464599	465419	464778	464840
I_0,6_1_150_50_3	499452	715682	536451	500677	500359	499949	500946	501067
I_0,6_1_150_50_4	489453	683366	519520	489803	488977	488725	488045	487464
I_0,6_1_150_50_5	476889	696883	528036	476519	475495	479186	471859	473226
I_0,6_1_250_10_1	257828	836022	318406	253069	254828	253176	251046	253227
I_0,6_1_250_10_2	191707	789424	265147	182096	178739	185951	182566	179872
I_0,6_1_250_10_3	222470	752406	296713	211891	213259	212468	211577	211480
I_0,6_1_250_10_4	246053	806107	320862	236568	233569	231172	235297	233957
I_0,6_1_250_10_5	247520	830619	321576	245994	241702	242467	241362	242918
I_0,6_1_250_30_1	554132	1069612	639809	523584	533227	528298	522319	527231
I_0,6_1_250_30_2	579517	1104814	665697	555849	554928	553078	553081	551669
I_0,6_1_250_30_3	583916	1104504	686667	553382	556183	556700	558504	550849
I_0,6_1_250_30_4	612955	1116479	715213	602504	598685	596178	597386	594065
I_0,6_1_250_30_5	665300	1288715	779307	640581	644674	643197	639371	640056
I_0,6_1_250_50_1	845268	1321979	953012	825120	830213	834076	831680	827171
I_0,6_1_250_50_2	916480	1395015	1004219	894789	899043	890834	892656	894043
I_0,6_1_250_50_3	920584	1397485	1024720	906121	905140	905767	897534	902539
I_0,6_1_250_50_4	929462	1415087	1030709	910601	908958	910653	915653	912005
I_0,6_1_250_50_5	829509	1393472	917412	809064	810155	802697	805509	798722
I_0,6_1_350_10_1	423277	1582364	567468	402203	403470	407296	410833	413570
I_0,6_1_350_10_2	310948	1465904	454737	297657	308333	301059	303633	303877
I_0,6_1_350_10_3	607498	1744699	749176	602973	590535	603154	596039	587717
I_0,6_1_350_10_4	543470	1645620	701133	536273	533854	537515	528975	533580
I_0,6_1_350_10_5	486422	1548653	649967	490796	484695	489688	500086	498665
I_0,6_1_350_30_1	941091	2024447	1154055	894526	902336	896749	898412	887205
I_0,6_1_350_30_2	859582	1877268	1057156	813308	801903	806529	801464	808049
I_0,6_1_350_30_3	1070878	2097773	1221440	1016334	1022186	1017349	1003050	1011661
I_0,6_1_350_30_4	1031538	2078007	1193021	973911	976247	978249	958779	972259
I_0,6_1_350_30_5	1058936	2021068	1230373	1005294	996934	1001983	1005997	991629
I_0,6_1_350_50_1	1415090	2450563	1598078	1351442	1353271	1362672	1356162	1352025
I_0,6_1_350_50_2	1539061	2549963	1687649	1477060	1469399	1483212	1478294	1480126
I_0,6_1_350_50_3	1345111	2365170	1508070	1275631	1298740	1289503	1284733	1291655
I_0,6_1_350_50_4	1391371	2372950	1582347	1332498	1339844	1337920	1342995	1332608
I_0,6_1_350_50_5	1430927	2441269	1599260	1377910	1382048	1385968	1360513	1394375

Table A.9: Individual results for the instances with $T = 0.6$ and $R = 1$.

References

[AR99] V.A. Armentano and D.P. Ronconi. Tabu search for total tardiness minimization in flowshop scheduling problems. *Computers & operations research*, 26(3):219–235, 1999.

[CFK06] C.S. Chung, J. Flynn, and
"O. Kirca. A branch and bound algorithm to minimize the total tardiness for m-machine permutation flowshop problems. *European Journal of Operational Research*, 174(1):1–10, 2006.

[DJ90] J. Du and Y.T.L. Joseph. Minimizing total tardiness on one machine is NP-hard. *Mathematics of operations research*, 15(3):483–495, 1990.

[FL08] J.M. Framinan and R. Leisten. Total tardiness minimization in permutation flow shops: a simple approach based on a variable greedy algorithm. *International Journal of Production Research*, 46(22):6479–6498, 2008.

[Gen03] M. Gendreau. An introduction to tabu search. *Handbook of metaheuristics*, pages 37–54, 2003.

[GL98] F. Glover and M. Laguna. *Tabu search*, volume 1. Springer, 1998.

[Glo86] F. Glover. Future paths for integer programming and links to artificial intelligence. *Computers & Operations Research*, 13(5):533–549, 1986.

[GS78] L.F. Gelders and N. Sambandam. Four simple heuristics for scheduling a flow-shop. *International Journal of Production Research*, 16(3):221–231, 1978.

[GS06] J.N.D. Gupta and E.F. Stafford. Flowshop scheduling research after five decades. *European Journal of Operational Research*, 169(3):699–711, 2006.

[HR04] S. Hasija and C. Rajendran. Scheduling in flowshops to minimize total tardiness of jobs. *International Journal of Production Research*, 42(11):2289–2301, 2004.

[HW87] A. Hertz and D. Werra. Using tabu search techniques for graph coloring. *Computing*, 39(4):345–351, 1987.

[Joh54] SM Johnson. Optimal two-and three-stage production schedules with setup times included. *Naval Research Logistics Quarterly*, 1(1):61–68, 1954.

[Kim93a] Y.D. Kim. Heuristics for flowshop scheduling problems minimizing mean tardiness. *Journal of the Operational Research Society*, 44(1):19–28, 1993.

[Kim93b] Y.D. Kim. A new branch and bound algorithm for minimizing mean tardiness in two-machine flowshops. *Computers & operations research*, 20(4):391–401, 1993.

[Kim95] Y.D. Kim. Minimizing total tardiness in permutation flowshops. *European Journal of Operational Research*, 85(3):541–555, 1995.

[KLP96] Y.D. Kim, H.G. Lim, and M.W. Park. Search heuristics for a flowshop scheduling problem in a printed circuit board assembly process. *European Journal of Operational Research*, 91(1):124–143, 1996.

[KTW10] T. Kellegöz, B. Toklu, and J. Wilson. Elite guided steady-state genetic algorithm for minimizing total tardiness in flowshops. *Computers & Industrial Engineering*, 58(2):300–306, 2010.

[NEJH83] M. Nawaz, E.E. Enscore Jr, and I. Ham. A heuristic algorithm for the m-machine, n-job flow-shop sequencing problem. *Omega*, 11(1):91–95, 1983.

[NS96] E. Nowicki and C. Smutnicki. A fast taboo search algorithm for the job shop problem. *Management Science*, 42(6):797–813, 1996.

[Ow85] P.S. Ow. Focused scheduling in proportionate flowshops. *Management Science*, 31(7):852–869, 1985.

[PCC02] J.C.H. Pan, J.S. Chen, and C.M. Chao. Minimizing tardiness in a two-machine flow-shop. *Computers & Operations Research*, 29(7):869–885, 2002.

[Pet09] P. Peterson. F2PY: a tool for connecting Fortran and Python programs. *International Journal of Computational Science and Engineering*, 4(4):296–305, 2009.

[PI77] SS Panwalkar and W. Iskander. A survey of scheduling rules. *Operations research*, 25(1):45–61, 1977.

[PR98] S. Parthasarathy and C. Rajendran. Scheduling to minimize mean tardiness and weighted mean tardiness in flowshop and flowline-based manufacturing cell. *Computers & Industrial Engineering*, 34(2):531–546, 1998.

[RH09] D.P. Ronconi and L.R.S. Henriques. Some heuristic algorithms for total tardiness minimization in a flowshop with blocking. *Omega*, 37(2):272–281, 2009.

[RS07] R. Ruiz and T. Stützle. A simple and effective iterated greedy algorithm for the permutation flowshop scheduling problem. *European Journal of Operational Research*, 177(3):2033–2049, 2007.

[RS08] R. Ruiz and T. Stützle. An iterated greedy heuristic for the sequence dependent setup times flowshop problem with makespan and weighted tardiness objectives. *European Journal of Operational Research*, 187(3):1143–1159, 2008.

[SDG89] T. Sem, P. Dileepan, and J.N.D. Gupta. The two-machine flowshop scheduling problem with total tardiness. *Computers and Operations Research*, 16(4):333–340, 1989.

[SRP12] N. Santos, R. Rebelo, and J.P. Pedroso. A tabu search for the permutation flow shop problem with sequence dependent setup times. *Int. J. Data Analysis Techniques and Strategies*, 2012. Accepted for publication.

[Tai93] E. Taillard. Benchmarks for basic scheduling problems. *European Journal of Operational Research*, 64(2):278–285, 1993.

[VR09] E. Vallada and R. Ruiz. Cooperative metaheuristics for the permutation flowshop scheduling problem. *European Journal of Operational Research*, 193(2):365–376, 2009.

[VR10] E. Vallada and R. Ruiz. Genetic algorithms with path relinking for the minimum tardiness permutation flowshop problem. *Omega*, 38(1-2):57–67, 2010.

[VRM08] E. Vallada, R. Ruiz, and G. Minella. Minimising total tardiness in the m-machine flowshop problem: A review and evaluation of heuristics and metaheuristics. *Computers and Operations Research*, 35(4):1350–1373, 2008.