Two Metaheuristics for the Single-machine Quadratic Tardiness Scheduling Problem

By

Tomás Cabrita Gonçalves

Thesis for Master Degree in Quantitative Methods in Economics and Management

Supervised by:

Professor Jorge Valente

2012

Biography

Tomás Cabrita Gonçalves was born on 5th of December 1988 in Porto, Portugal. He did all of his studies in Porto, enrolling into the Faculty of Economics of the University of Porto in 2006. In 2010, he finished his under-graduation in Economics and, in that same year, he enrolled in the same Faculty, for the Master degree in Quantitative Methods in Economics and Management.

Acknowledgments

I would like to thank my family for giving me the chance to pursue this Master degree; Sofia, Fátima and Pedro, for study companionship and friendship which kept me motivated (almost) at all times; all my friends, for relaxing times in between work and of course, my supervisor: Professor Jorge Valente.

Abstract

In this study, we consider the single machine scheduling problem with quadratic tardiness costs. A very efective problem-specific Local Search is presented, followed by two metaheuristics: an Iterated Local Search and a Variable Greedy. These two procedures include the above mentioned Local Search and also a Dispatching Rule, used in the generation of a high-quality initial solution.

Both metaheuristics are tested on a set of random computer generated problems, with a wide range of characteristics and difficulty and are shown to have good performances in very reasonable computational times, including optimally solving instances with up to 50 jobs.

Resumo

Neste estudo, consideramos o problema da minimização dos custos quadráticos de atraso, no escalonamento de trabalhos numa máquina. É apresentado um procedimento de Pesquisa Local altamente eficiente, específico para o problema, seguido de duas metaheurísticas: Iterated Local Search e Variable Greedy. Estes dois procedimentos incluem a supra-mencionada Pesquisa Local e também uma Dispatching Rule,usada na geração de uma solução inicial de elevada qualidade.

Ambas as metaheuristicas são testadas num conjunto de problemas gerados aleatoriamente por computador, com leque variado de características e dificuldades, e mostram ter boas performances com tempos de computação muito razoáveis, incluindo a capacidade de encontrar a solução óptima em instâncias com até 50 trabalhos.

Contents

List of Figures

1 Introduction

The present work deals with the weighted quadratic tardiness minimization problem, in a single machine environment. We can summarily present this problem as: a set of n independent jobs $\{J_1, J_2, \dots, J_n\}$ that need to be scheduled in a single machine which handles only one job at a time. Job $J_j, j = 1,2, \cdots, n$, at any given schedule, requires a processing time p_j, is completed at C_j, and is so, preferably before its due date d_j.

Single machine settings can be found, for instance, in the chemical (see (Wagner et al. 2002) for a real industry example) and paint industry, printing press and paper bag production (Dhingra 2010). Furthermore, the findings made in the single processor case, can often be transposed to more complex scheduling environments and even improve settings in which a single bottleneck machine is the source of inefficiency, hence the importance of studying this subject (Schaller and Valente 2012).

While, Early/Tardy Scheduling Problems are closely related to the concept of Just-in-Time (Valente 2006) and so, they are applied to cases in which, both stock and delay minimization is pursued; Tardiness Minimization Problems as the one addressed in this paper, fit better in settings where to some degree, the early production, that is, the accumulation of stock, is disregarded in favor of delivering to customers on time, i.e. when the cost of having inventory can be neglected when compared to that of late shipping. Note that, the cost of tardy supplying can weigh heavy on a company and may result from contractual penalties, or lead to loss of customers' good will and even lost sales (Valente and Schaller 2012). In (Sun et al. 1999), it is stated that comparatively to linear tardiness objective functions, its quadratic version is a more robust measure of the quality of service and that it better highlights, extreme tardiness situations. Furthermore, it is also said, in analogy to Taguchi's loss function (Taguchi 1986), that customers' dissatisfaction is quadractically related to tardiness. In his function there is an explicit relation between a

target level of output/production, the actual level and a loss for society. The loss equals the squared deviation from the target, multiplied by a weight.

Our objective function is defined as $\sum_{j=1}^{n}\left(w_j T_j^2\right)$, with $T_j = \max\{0, C_j - d_j\}$, standing for tardiness and w_j a weight associated with job j. In this case, instead of a target output level of production, the target is that all jobs are completed before their due dates, that is, $C_j \leq d_j \ \forall\, j \in [1, n]$. Note that, not including quadratic earliness penalties means there is no regard for the left side of the function. So, with this quadratic cost function, the later the Completion time is, compared to the Due Date, the heavier the penalty will be. This relation is congruent with the examples of the risk for tardy deliveries stated above.

Over the past years, since (Johnson 1954)'s first approach and identification of flow shop scheduling problems, there has been an intensive focus on the subject. In the surveys of (Gupta 2006) and (Potts and Strusevich 2009) we can find chronologically organized overviews on scheduling studies made in the five decades after that seminal work, with regard to technological and methodological advances and to the progressive widening of the scheduling problems considered by the research community. As these surveys show, there has been an extensive investigation in this area and the problems that emerged are very diverse. Nevertheless, for the sake of brevity, we will here emphasize, only the work done on the problems more closely related to our own, specifically, tardiness minimization problems and the ones involving squared tardiness.

Starting with standard and weighted tardiness minimization problems, we refer to (Abdul-Razaq et al. 1990), (Potts and Van Wassenhove 1991), (Sen et al. 2003) and (Koulamas 2010) for surveys on some exact approaches and heuristics, however noting, that these works barely explore the use of metaheuristics on the problem. Metaheuristics have been increasingly gaining more followers, for their quickness and efficiency, especially after (Du and Leung 1990) proved that the single-machine problem considering total unweighted tardiness minimization is NP-hard. This means that, there likely isn't any algorithm to optimally solve the problem in polynomial time. For reviews that include metaheuristics we suggest (Vallada et al. 2008)'s work for the m-machine case and (Allahverdi et al. 2008), which is a very complete and extensive survey on various

scheduling problems with setup considerations.

Comparatively, in terms of squared tardiness and weighted squared tardiness minimization problems, there has been, as of yet, very little research done. Due to its complexity, all of the approaches to squared tardiness use a Lagrangian relaxation, to decompose the problem into smaller sub-problems, which is a technique that was first applied to the travelling salesman problem, but was later adapted by (Fisher 1981) for Scheduling problems. This technique is used for quadratic tardiness minimization in the work of: (Hoitomt DJ 1990) on parallel machines with precedence constraints; of (Sun et al. 1999) on single-machines with sequence dependent setup times; in (Thomalla 2001)‘s approach on job shop scheduling with alternative process plans (e.g. machines that have different efficiency on processing the same job), in (Valente et al. 2011)’s genetic algorithms and finally both (Kianfar and Moslehi) and (Schaller and Valente 2012), who present different dominance conditions and branch-and-bound algorithms.

As for quadratic earliness and tardiness: (Valente and Alves 2008) presented several dispatching heuristics for the single-machine problem and tested them on large a range of instances, (Valente and Moreira 2009) propose a few greedy randomized dispatching rules for the same problem and show how they outperform simple dispatching rules for some instance sizes, later, in the work of (Valente 2010) a beam search is created and the effects of three different dispatching rules on the procedure, are tested.

The last scheduling problem that is reviewed here is the linear earliness quadratic tardiness minimization. Attempts to solve this problem were made by the branch-and-bound and the heuristics in (Schaller 2004), the dispatching rules in (Valente 2007), later (Valente 2008a) proposed another beam search procedure, then, in (Valente and Gonçalves 2009) a genetic algorithm approach was presented. (Valente and Schaller 2010) also present a genetic algorithm and a backward Dispatching Rule and apply it to both the “no idle time” version of the problem and the one that includes it, (Behnamian and Zandieh 2011) use a colonial competitive algorithm to solve this problem in hybrid flow shops and (Rahmani and Mahdavi) propose another genetic algorithm for the single-machine problem with preemptions allowed. Lower bounding and branch-and-bound procedures are also presented in (Valente 2008b), for optimally solving the problem with instances up to 20 jobs.

The remainder of this work is organized as follows: a local search procedure tailored for the quadratic problem is presented in section 2 and the logic behind it pointed out. Afterwards, in section 3, the proposed metaheuristics are introduced and thoroughly explained, starting with the Iterated Local Search and ending with the Variable Greedy. The following section encloses the computational experiments and results and lastly, in section 5, final observations and conclusions are made.

2 Local Search Procedure

In this section we present our problem-specific Local Search Procedure (LS), which is used in both meta-heuristics presented in the next section.

In short, we can state that this procedure consists of adjacent interchanges guided by the idea that: early jobs should be "pushed" as forward as they can (that is, without them becoming tardy) in a schedule and tardy jobs "pulled" backwards.

Figure 1 shows the pseudo-code of the Local Search which is afterwards explained in detail.

Let $[i]$ be a position in the sequence and i be the job that position and $[j] = [i + 1]$ the next adjacent position to $[i]$ and j the job in that position.

1. Set $[i] = 1$.
2. While $[i] < n$:
   2.1. If jobs i and j are early:
      2.1.1. If $d_i > d_j$:
         2.1.1.1. Swap jobs i and j.
         2.1.1.2. If $[i] > 1$, set $[i] = [i - 1]$.
      2.1.2. Otherwise, set $[i] = [i + 1]$.
   2.2. Else if jobs i and j are tardy:
      2.2.1. If $w_i(2T_i + 1)p_j < w_j(2T_j + 1)p_i$:
         2.2.1.1. If the objective function value is improved by swapping jobs i and j:
            2.2.1.1.1. Swap jobs i and j.
            2.2.1.1.2. If $[i] > 1$, set $[i] = [i - 1]$.

2.2.1.2. Otherwise, set $[i] = [i + 1]$.

2.2.2. Otherwise, set $[i] = [i + 1]$.

2.3. Else if job i is early and job j is tardy:

2.3.1. If $d_i \geq C_j$:

2.3.1.1. Swap jobs i and j.

2.3.1.2. If $[i] > 1$, set $[i] = [i - 1]$.

2.3.2. Else if $w_{[i]}(C_j - d_i)^2 < w_j(2T_j + 1)p_i$

2.3.2.1. If the objective function value is improved by swapping jobs i and j:

2.3.2.1.1. Swap jobs i and j.

2.3.2.1.2. If $[i] > 1$, set $[i] = [i - 1]$.

2.3.2.2. Otherwise, set $[i] = [i + 1]$.

2.3.3. Else, set $[i] = [i + 1]$.

2.4. Else, set $[i] = [i + 1]$.

FIGURE 1 – PSEUDO-CODE FOR THE LOCAL SEARCH PROCEDURE

In sum, what the search described above does is: run one job at a time, testing the possible swap of that job with its next adjacent job. This search starts from the job standing in the first position and ends in the last one and throughout this run, three situations are of interest: first, if the two jobs being tested are early (corresponding in figure one to step 2.1.); secondly, if the jobs are both tardy (step 2.2.) and lastly, when the first job is early and the second is tardy (step 2.3.). Naturally, the situation where the first job is tardy and the second is early is not of interest, as the swap would certainly increase the objective function value, by making the first job even tardier, so this case is covered by step 2.4., where the search is instructed to bypass it.

In step 2.1., when the jobs are both early, the swap is made if the due date of the first job (d_i) is greater than that of the second (d_j), thus, following the logic stated at the beginning of this chapter.

In step 2.2., when the two jobs are tardy, a lower bound on the increase of the cost for the first job is compared with an upper bound on the decrease of the cost for the second

job. The idea behind the expression is that: we cannot say that, if the lower bound is inferior to the upper bound, there will be a sure advantage in doing the swap, but logically, if the opposite situation occurs, no improvement of the objective function value will be possible. So, the search space when both jobs are tardy is to be reduced to the first scenario.

The condition itself is based on the findings from (Schaller and Valente 2012). Both sides of the expression include the derivative of the objective function for the jobs we are testing: $w_i(2T_{[i]} + 1)$, for i and $w_j(2T_j + 1)$, for j. While on the left side, we have the derivative of the function at job i symbolizing the increase of cost in the function, from moving its completion time forward, by one unit; on the right side, we have the derivative relative to job j, that corresponds to the decrease in the cost function, caused by moving job i's completion time backwards by one unit.

Note that, tardiness,T, will vary in the same proportion and direction as completion times, C, and this, consequently, causes the objective function to increase/decrease quadractically, in accordance (see Section 1 for objective function and tardiness function in terms of completion times). Using that logic, we can state that: the increase in the cost function caused by moving the first job forward, is at least, equal to the derivative of the objective function at that job, times the processing time of job j, p_j, which is the measure of the time for which i is being postponed. Inversely, for job j we can say that the benefit will be at max, the derivative, times the processing time of i, p_i (the measure of time for which j is anticipated).

In step 2.3. the scenarios where i is early and j is tardy are covered. The first situation (step 2.3.1.) is straightforward. The swap is done if the due date of i is larger than the completion time of j, i.e. if after the swap i is still early. The possible advantage in the remaining situation (step 2.3.2.), in which the first job becomes tardy after the swap, naturally needs to be assessed. While the right side of the expression, the upper bound in cost reduction, remains the same as in the previous step (2.2.), the left side differs from before. Now, the measurement of the cost only needs to be considered from the point in which the job starts becoming tardy after the swap (otherwise it would fit in step 2.3.1.). So, the left side of the constraint ($w_i(C_j - d_i)^2$) is not a lower

bound for the increase in tardiness, but the real increase in the cost of the objective function caused by moving i forward.

Having scrutinized the functioning of the Local Search Procedure, we proceed to the next section, with the introduction of the metaheuristics that incorporate it.

3 Meta-heuristics

As mentioned previously, in this section we propose two metaheuristics that consider the specificities of the problem: an Iterated Local Search (referred to, in this paper, as ILS) and a Variable Greedy algorithm (referred to as VG). In their basic form, both these procedures are well known in this field of knowledge; still, we here explain more thoroughly the characteristics and functioning of our designs and their particularities.

3.1 ITERATED LOCAL SEARCH (ILS)

An iterated local search or iterated descent is a multi-start search procedure that is intended to overcome the problem of most local search procedures: getting "stuck" at local optima. This is done by a move called the *kick*, which is a pre-determined (that is, in the range of a certain defined neighborhood) type of modification applied to a local optima, hoping for that move to be sufficient to escape its descent, but simultaneously, avoid getting too far from an area of the search space that could be fruitful (Congram et al. 2002).

Our ILS, starts with the generation of a good solution through an efficient constructive heuristic, followed by the previously presented LS, unlike many multi-start heuristics which start with random-generated solutions. The rest of the algorithm is very similar to general iterated local searches, with a slight dissimilarity when it comes to the *kick*-acceptance criterion: which in our case is null i.e. the *kick* is always accepted, following the approach of (Congram et al. 2002).

The pseudo-code for the ILS we propose, is presented below:

Repeat until running time ≥ time_limit or the best solution found S_B has a cost of 0 and is therefore optimal:

Generate solution S using the Qback_v6 dispatching rule.

1. If S better than S_B :

   1.1 Apply LS.
2. Else, apply LS with probability ls_prob
3. Set solution as S'
4. Update S_B
5. If number of iterations without improvement of $S_B = \beta$:

   5.1 Do BackTracking, that is, set $S' = S_B$
6. Else, set S' as current solution S_C
7. Do *kick* from S_C
8. Go to step 2

FIGURE 2 – PSEUDO-CODE FOR THE ITERATED LOCAL SEARCH PROCEDURE

In step 1, the initial solution is generated using QBack_v6, which is the backward dispatching rule for the quadratic tardiness problem, that resulted in better objective function values, of all the dispatching rules tested in (Valente and Schaller 2012b). It consists of a constructive heuristic that, starting from the last position in the schedule, decides which job to add to the solution, in a backward order, based on the following priority index:

$$QBack\_v6 = \begin{cases} p_j & if\ S_j^B \leq 0 \\ -\left(\frac{w_j}{p_j^{mod}}\right)\left[\left(s_j^B\right)^2 - v\left(max\{t^B - p_{max} - d_j, 0\}\right)^2\right] & ,otherwise \end{cases}$$

FIGURE 3 – PRIORITY INDEX OF BACKWARD DISPATCHING RULE QBACK_V6

Where: w_j is the weight of job j (present in the objective function); $p_j^{mod} = min\{p_j, T_{min}\}$ is the modified processing time of job j, T_{min} represents the minimum

tardiness of all tardy jobs at the current time; $s_j^B = t^B - d_j$ is the slack of job j in the backward schedule *B*, v is set at 0.5; t^B is the current time in the backward schedule; p_{max}, is the maximum processing time of all unscheduled jobs and, d_j and p_j are as defined earlier in this work.

The logic behind this priority index is the following: if the job is early, then its priority index will be positive and equal to its processing time, which will allocate higher-processing-time early jobs closer to the end of the schedule; when all unscheduled jobs are tardy, the index becomes negative, but still increases with processing time (but now: p_j^{mod}). This latter part of the index takes, to a certain degree, into account the opportunity cost of scheduling job j at that point, by using both p_j^{mod} and p_{max}, that include information about other unscheduled jobs.

In step 2, the LS procedure presented in the previous section is used to improve the solution provided by the Dispatching Rule. For subsequent solutions, steps 2 and 3 result in always applying the local search procedure if the current solution is is better than the best one found so far, and applying that local search procedure with probability ls_prob otherwise.

The current best solution that is kept in memory, S_B, is updated in step 5 and if the count of iterations in which S_B has not been improved is equal to β, which is a user-defined parameter, the algorithm backtracks. This means that we go back to the previous best solution and restart the procedure from it.

Otherwise, we go to step 5 and then 6 where, a predetermined type of modification is made, called *kick*, which is the ILS's way to overcome local optima. In this case, it corresponds to α random swaps and these moves are always accepted.

Afterwards, LS is done again starting from this new solution (step 2), if the stopping criterion hasn't been reached i.e. a user defined time limit has not been exceeded and a solution with an objective function value of 0, which would naturally be optimal, has not been found.

3.2 VARIABLE GREEDY (VG)

Variable Greedy is a very recent hybrid metaheuristic created by (Framinan and Leisten 2008) that mixes components of both the Iterated Greedy and the Variable Neighborhood Search algorithms. Of the first algorithm, the VG has inherited the destruction and construction phases that will be explained below and as for the latter algorithm, the characteristic that was appropriated by the VG is the systematic change of neighborhood (variable neighborhood).

Below is presented the pseudo-code for the VG metaheuristic and further on, we proceed explaining its functioning:

Repeat until running time ≥ time_limit or the best solution found S_B has a cost of 0 and is therefore optimal:

1. Generate solution S, using the Qback_v6 dispatching rule
2. If S better than S_B :
    a. Apply LS.
3. Else, apply LS with probability ls_prob.
4. Update S_B.
5. Set $k = 1$,
6. Until $k = \%\text{ns} * (\text{n} - 1)$, do
    a. Remove k jobs with the highest Objective Function Value (ofv) from S, forming set S^R and the remaining non-removed jobs forming S^{NR}.
    b. Order each job in S^R in decreasing order of ofv
    c. Starting from the first job in S^R, until all k jobs are reinserted (obtaining S'), Insert job in best possible slot in the current partial sequence.
    d. If S better than S_B :
        i. Apply LS.
    e. Else, apply LS with probability ls_prob
    f. If S' better than S, then set $S' \rightarrow S$, update S_B and set $k = 1$.

g. Otherwise set $k \rightarrow k + 1$.

7. Go to step 5.

FIGURE 4 – PSEUDO-CODE FOR THE VARIABLE GREEDY PROCEDURE

This procedure starts-off in the same manner as the ILS: using Qback_v6 for the generation of the initial solution and applying LS.

Next, in step 6, we move on to the destruction phase, which is, as mentioned previously, a "heritage" of Iterated Greedy algorithms. The *k* jobs removed are the ones that most "damage" the Objective Function Value i.e. the ones with the biggest cost, and, after ordering, they are greedily reinserted (construction phase) one by one, in each job's best position. Afterwards, the solution found is compared with the initial one and, in short, the logic followed is: each time the solution is improved at an iteration, we iterate again, using that new solution to start from and reset the destruction parameter, *k*, to one; otherwise, we continue exploring an increasing part of the neighborhood of the same solution by raising *k* at each iteration, by one unit. The maximum limit for *k* is $\%ns * (n - 1)$. To the best of our knowledge, the parameter %ns consists of an innovation in this algorithm, as usually the maximum value for *k* is equal to (n-1).The stop criterion is identical to the one used in the ILS algorithm.

We conclude this sub-section by emphasizing the algorithm's regard for the problem specificities in three stages: the first is the Dispatching Rule which, as mentioned before, was tailored specifically for the quadratic tardiness scheduling problem; the same applies for the LS; and, finally, the fact that the selection and the sorting in the destruction phase are made in terms of tardiness.

4 Computational results

In this section, we present the problem test-bed generation method for the various instance sizes and comment on the preliminary parameter tuning, that was made prior to the actual tests. Lastly, the results of the algorithms are shown and analyzed for conclusions.

4.1 Experimental design

In this work, we use the same method that was used to create the linear weighted tardiness problem instances available in the OR-Library (http://people.brunel.ac.uk/_mastjjb/jeb/orlib/wtinfo.html) and it works as follows.

Both an integer processing time and an integer weight are generated for each job *j*; respectively, p_j is generated from a uniform distribution $[1, 100]$ and w_j from a $[1, 10]$. An integer due date, d_j, is also generated, for each job, from the uniform distribution $[P(1 - T - R/2), P(1 - T + R/2)]$, where P is the sum of the processing times of all jobs, T is the tardiness factor and R is the range of due dates. Both the tardiness factor and the range of due dates parameters were set at 0.2, 0.4, 0.6, 0.8 and 1.0 and for each combination of problem size n, T and R, 50 instances were randomly generated. So, a total of 1250 instances were generated for each problem size. These computational tests were tested on problem sizes of 10, 15, 20, 25, 30, 40, 50, 75, 100, 250 and 500.

The procedures were coded in C++ and executed on a personal computer with Intel Core2 Quad Q6600 2.40GHz processor and 3GB RAM on Windows 7 32bit Operative

System.

4.2 Preliminary parameter adjustments

Both the metaheuristics' parameters were carefully tested, for maximization of their overall performance, and their results were studied in terms of the number of jobs (n), tardiness factor (T) and due date range (R).

The ILS was tested first for ls_prob values of 0%, 10%,...,100%, and provided good results for values of 90% and slightly better ones for 100%, the value that was chosen as final. We then proceeded testing the α and β values, simultaneously. Alfa was tested for values of 5, 6 and 7, and beta for values of 5, 10, 20 and 25. We observed that the combination of both being equal to 5 yielded better results.

As for the VG, the ls_prob and %ns were also tested together. The first, for values of 0%, 10%,…, 100%, and the latter for values of 10%, 20%,…,100%. The overall best combination of values was: ls_prob=60% and %ns=90%.

The time_limit parameter used in the stop criterion was also determined by preliminary tests. As a result of these tests, the maximum computation time (time_limit) was set equal to $0.5 + 0.0001n^2$. This maximum time was the same for both algorithms, so that both metaheuristics used the same stop criterion and were tested under similar conditions.

The parameter adjustment tests were performed on a separate and smaller set of instances. The instances in this smaller set were generated just as previously described for the larger set. However, the smaller set contained only instances with 10, 25, 50, 100, 250 and 500 jobs, and only 5 instances were generated for each combination of T and R.

4.3 Results

We now proceed with the analysis, of the performance of the two presented algorithms, involving: a comparison with the optimal results for problems with up to 50 jobs; a comparison with the performance of the previously mentioned Dispatching Rule

(Qback_v6), with and without Local Search and a comparison between the ILS and the VG.

But, before the presentation of the results we would like to make a quick note, justifying some of the variables utilized in the analysis: it is very common, in this line of work, to use a measure called relative improvement, which directly measures the relative improvement of the OFV of a heuristic over another, calculated as a simple percentage. However, we draw attention to the fact that this may, for some instances, originate a division by zero, thus making it possible to have undefined values for those instances. To overcome this problem, other measures were used in this work, which will be explained further on. Also, we remark that 10 different seeds were used for the metaheuristics.

4.3.1 Comparison with Optimal Results

For this subsection, the optimal results were obtained with resource to the Branch-and-Bound procedure of (Schaller and Valente 2012), for instances with up to 50 jobs, and then compared to the two metaheuristics' results.

We present three relative improvement ratios that measure the relative improvement of the optimum over the several heuristics (table 1), and also n_opt, which is a statistic for the number of times the optimum is attained. The ivh% (Improvement versus heuristic) for a particular instance is as follows: if the objective function value of the heuristic (h_i) we are comparing ($OFVh_i$), is equal to the optimum (OFVopt), then, the ivh% is set at 0, otherwise, it is equal to $(OFVh_i\text{-}OFVopt)/(OFVh_i)x100$. Note that this formula is directly applicable to DR and DR+LS, because these are deterministic heuristics, and hence only applied once for each instance; however, this is not the case for the ILS and the VG metaheuristics which, as previously mentioned, have been ran for ten seeds each. So, the ivg_avg% and the ivh_best are calculated for these two heuristics, analogously to the ivh%, but, the first using the average of the ten seeds for each instance, instead of the $OFVh_i$, and the latter using only the best $OFVh_i$ of the ten seeds. For the same reason, the n_opt statistic is calculated differently: while for the dispatching rule a value of 1 is attributed for each time the heuristic equals the optimum, for the other two algorithms, a value of 0,1 is attributed to each seed if it equals the optimum, thus being equal to one, for

a certain instance, only if all the seeds led to an optimal solution.

In Table1 it is visible that the Local Search procedure greatly improves the Dispatching rule, both in n_opt (see column 8 versus 9) and in the reduction of the "distance" to the optimum, measured by the ivh% (column 2 versus 3).

It is also clear that the ILS is, on average, the closest to the optimum of all four procedures, failing to reach optimality in only 10 out of 12500 seeds, and being closer than the other three heuristics, even for that instance size (n=40). The VG shows a poorer performance than the ILS and it appears to be more strongly influenced by instance size (decreasing its performance as *n* increases). However, the ivh_best% suggests that, it is also capable of achieving very good performances, in some cases. We would also like to restate that both the ILS and the VG have the DR+LS procedure embedded in their code, and they both significantly outperform it, for these instances, thus proving the effectiveness of these two metaheuristics.

Table 2 provides a more thorough analysis of the n=40 sized results, for the two presented metaheuristics and the Dispatching Rules. Looking at the three types of ivh, we see that the lowest values, in general, for the four heuristics, occur for T= 1 and 0,8, which suggests that the heuristics deal well with larger tardiness factors. Although, in particular, the T=0,8 and R=0,8 instances represent the worst results of ivh_avg% and ivh_best%, for the ILS and the VG, they are still below the results of the Dispatching Rules' and they are still low. The worst results of the improvement versus heuristic ratios, are those of the DR and DR+LS, for T=0,2 and 0,4, which reach more than 3% "distance" to the optimum, in some cases. Comparatively, this is a very high percentage for this problem size and suggests that the Dispatching Rule, may not deal well with low tardiness factors. These conclusions are in accordance with the results for the Qback_v6 Dispatching Rule (Valente and Schaller 2012).

It is also of note that very little influence of the Range factor (R), over any of the variables is perceivable in the table, except for the fact that heuristics seem to take advantage of low T values (e.g. 0,2) when combined with high R values. These instances tend to be the easier ones, as the low number of tardy jobs, allied with a possible wide variety of due dates, gives the algorithms more flexibility in finding better results. When looking at the last four columns, we again see that the best results occur for T=0,2, where

the optimum is reached in more than 90% of the times, for all heuristics. It is also visible that good results are obtained for T=0,4, although the DR and DR+LS start decreasing their performance. Looking closer at what happens as T grows, we can see that the LS procedure takes advantage of larger tardiness factors, as it clearly differentiates the DR+LS's n_opt statistic from that of the DR. However, this effect does not affect the ivh%, so clearly, showing that the relative corresponding improvement may not be as relevant.

4.3.2 Comparison with the Dispatching Rule

As was stated before, both the metaheuristics we present in this paper use the previously mentioned Dispatching Rule and the Local Search procedures. Being so, we saw fit to dedicate a subsection to the comparison and analysis of the two sets of procedures, in order to see what improvements were made by the first set over the latter.

In Table 4, the comparison with the results of the Qback_v6 Dispatching Rule, with and without Local Search, is presented. In analogy with the ivh and the n_opt statistics, ivd and ivd+a and n_btr_d and n_btr_d+a were created for making such a comparison. The ivd or Improvement versus Dispatching Rule is calculated the same way as ivh, only substituting $OFVh_i$ for OFVd, the objective function value of the Dispatching Rule. As for n_btr_d, it measures the number of times that the metaheuristic, to which we are comparing, yielded a smaller, thus better (since we are minimizing) OFV, than OFVd. Like before, as we have ran ten seeds, for each program and instance, we confer a value of 0,1 each time the metaheuristic result for a seed is better than the result, for that same instance, of the Dispatching Rule.

Ivd+a and n_btrd+a are calculated in the same manner, but this time with reference to the Dispatching Rule with Local Search, instead of just the Dispatching Rule.

The number of times the ILS and the VG improve over the DR and DR+LS are very close to one another and have a light growing (non linear) tendency with the instance size, although the ILS proves to do slightly better for n>75.

For small instance sizes, this happens because the optimum is attained for a large number of instances by the Dispatching Rules, so no improvement can be made by the other procedures. To further prove this point, we added a Table 5 to the Table's section summarizing the n_eql_d and n_eql_d+a statistics, which give us the number of times the metaheuristics' OFV equaled respectively, the OFV of the DR and the DR+LS. By doing this, we found out that, at least for instances with up to 50 jobs, the sum of n_eql_d and n_eql_d+a fairly coincide with the sum of n_opt from the first table and, furthermore, very few instances apart from these, were not in fact instances where improvement occurred.

On average, the improvement over the procedure with Local Search is lower than for the DR, especially for smaller instances. This is to be expected, and explained by the fact that the DR+LS procedure has a better performance as shown in Table 3 (indeed, DR+LS can only provide better or equal results to those of DR), which makes it more difficult to get better results, both in number (n_btr_d+a) and in quality (ivd+a_avg% and ivd+a_best%) . This is also shown by the poor improvement on those smaller instances, which again suggests that optimality was reached by the DR+LS.

Table 3 provides a measure for the effect of the LS procedure over the Dispatching Rule. Although the number of times that the DR is improved by the Local Search increases with instance size, its average ivd% varies in the opposite direction. While the first result likely happens because there are fewer instances where the optimum has not been reached, for lower sized problems (as explained before), the latter has to do with the fact that the bigger the instances, the harder improvements (in terms of ivd%) get; although, in another perspective, more opportunities for improvements lie within these problems, which can also explain the high n_btr_d for large instances.

We analyze the case of n=100 in Table 6, discriminating by T and R, both for the comparison of the metaheuristics with the DR and with the DR+LS. For both these comparisons, the largest improvements lie in instances with low values of tardiness, drastically decreasing for bigger tardiness levels. We also point out to the fact that for medium and large T values (i.e. 0,6; 0,8; 1), both heuristics improve DR and DR+LS in a large number of instances; in fact, for the DR all the instances are improved by both procedures. It is also again visible, for instances with 100 jobs, that the improvement over the Dispatching Rule with Local Search is lower, on average, for every instance type.

4.3.3 Comparison between metaheuristics

We now proceed with the comparison of the two metaheuristics presented in this work. To do so, and in line with what has been described before, a very simple comparison measure is utilized: the ivw or improvement versus worst. The ivw%, for a particular instance and seed, is set at zero if $OFVh_i$, that is, the objective function value of the algorithm whose performance we are measuring, is equal to that of the worse OFV (OFVw) of all 20 seeds (10 seeds of each algorithm), for that instance. Otherwise, the ivw% is equal to $(OFVw-OFVh_i)/(OFVw)x100$. Logically, the ivw_avg%, for any one of the algorithms, is equal to the average of the 10 seeds' ivw%, for each instance, and the ivw_best% corresponds to the largest of those 10 ivw%.

The ILS shows, in table 7 better results for all sizes of problems tested, whether on average, or when comparing the best results. We denote that this advantage over the VG tends to increase with instance size. This becomes especially clear for the n=250 and 500 sizes, when the ILS's mean of the ivw_avg% becomes more than twelve times bigger than the ones of the VG.

The effects of the T and R factors for n=250 instances were tested in Table 8. Again, low tardiness factors (0,2 and 0,4) implied larger values of ivw, and from the values the improvements started to decrease significantly, finally reaching extremely low values. The values shown for the ILS are superior to those of the VG, for all combinations of T and R, once more proving the supremacy of the first metaheuristic over the latter, in what regards solution quality.

After the comparison of the ILS and the VG in terms of OFV, we now proceed to study their efficiency, by analyzing their computational runtimes.Table 9 summarizes the average runtimes of the heuristics, in seconds, until the best solution of that instance was found. As it was expected, runtimes grow with instance size and it is also clear that the ILS is faster for instances with up to 100 jobs, point from where the VG takes clear advantage, reaching a value of more than eleven times less computational time than the ILS, for n=500.

Finally, we made the parameter influence analysis of T and R, for the runtimes of the two algorithms, this time for instances of 250 jobs (Table 10). The conclusions of that

analysis are that the ILS is slower than the VG for all levels of T, except for T=1. While the first increases until T=0,6 and then decreases, having particularly bad performances for that value of T and also for T=0,8, the VG increases more steadily along with the increase of T.

5 Conclusion

The present work dealt with the single machine scheduling problem with quadratic tardiness costs. Two metaheuristics, an Iterated Local Search and a Variable Greedy, and a Local Search procedure, specifically tailored for this problem, were proposed and analyzed. The Dispatching Rule that is used to generate initial solutions was also fairly explored and used for comparison of results, along with the Local Search.

The results show that, for small sized problems, both metaheuristics reach the optimum on most of the instances, with a very low computational effort. When compared to the Dispatching Rule, a procedure proven to be very effective and efficient for this problem, the two metaheuristics, again, demonstrated their value by providing better results for a very large number of instances, , for most instance sizes, within very low computational times.

Our study, however, suggests that the ILS is the overall, better metaheuristic, for almost every analysis. Although, on average, it has comparatively high runtimes for instances with 250 and 500 jobs, the improvement it makes is over 12 times larger, than that of the VG. Since these improvements can consist in very large cost savings for a company, and since the runtimes of the ILS, even for the highest instance sizes, still seem reasonable in a real environment, this procedure is to be preferred.

As was mentioned previously, this problem is NP-hard and so, it is our understanding that future research should be focused on producing better and faster, metaheuristics, cleverly tailored for the specificities of this and other similar problems. By "similar problems" we wish to suggest that new real world constraints could be included and problem complexity increased, as to better approximate the reality of industrial environments and to widen the applicability of these techniques. As such, the consideration

of release dates, setup times or additional machines are certainly interesting opportunities for further research.

6 Tables

	Average ivh%		Average ivh_avg%		Average ivh_best%		Sum of n_opt			
N	**DR**	**DR+LS**	**ILS**	**VG**	**ILS**	**VG**	**DR**	**DR+LS**	**ILS**	**VG**
10	0,5142	0,2626	0	0	0	0	836	1117	1250	1250
15	0,5904	0,3906	0	0	0	0	670	1036	1250	1250
20	0,6522	0,4660	0	0	0	0	531	935	1250	1249,9
25	0,7801	0,6217	0	0	0	0	485	882	1250	1248,2
30	0,7026	0,5552	0	0,0034	0	0	441	849	1250	1237,6
40	0,6591	0,5559	0,0080	0,0195	0,0080	0,0093	372	744	1249	1210,2
50	0,7388	0,6453	0	0,0209	0	0,0074	330	680	1250	1144,3

TABLE 1 – COMPARISON WITH OPTIMUM RESULTS

		Average ivh%		Average ivh_avg%		Average ivh_best%		Sum of n_opt			
T	**R**	**DR**	**DR+LS**	**ILS**	**VG**	**ILS**	**VG**	**DR**	**DR+LS**	**ILS**	**VG**
0,2											
	0,2	2,6022	2,3322	0,0000	0,1571	0,0000	0,0169	29	36	50,0	46,0
	0,4	0,6478	0,0000	0,0000	0,0000	0,0000	0,0000	48	50	50,0	50,0
	0,6	0,0000	0,0000	0,0000	0,0000	0,0000	0,0000	50	50	50,0	50,0
	0,8	0,0000	0,0000	0,0000	0,0000	0,0000	0,0000	50	50	50,0	50,0
	1	0,0000	0,0000	0,0000	0,0000	0,0000	0,0000	50	50	50,0	50,0
		0,6500	**0,4664**	**0,0000**	**0,0314**	**0,0000**	**0,0034**	**227**	**236**	**250,0**	**246,0**
0,4											
	0,2	0,9374	0,7939	0,0000	0,0824	0,0000	0,0076	9	21	50,0	40,1
	0,4	3,1943	3,0098	0,0000	0,0067	0,0000	0,0000	9	19	50,0	48,0
	0,6	2,5639	2,2808	0,0000	0,0091	0,0000	0,0000	15	29	50,0	49,0
	0,8	1,0386	0,8410	0,0000	0,0006	0,0000	0,0000	36	42	50,0	49,9
	1	1,2643	1,1478	0,0000	0,0000	0,0000	0,0000	42	45	50,0	50,0
		1,7997	**1,6146**	**0,0000**	**0,0197**	**0,0000**	**0,0015**	**111**	**156**	**250,0**	**237,0**
0,6											
	0,2	0,2235	0,1544	0,0000	0,0104	0,0000	0,0017	2	22	50,0	43,3
	0,4	0,6478	0,5414	0,0000	0,0090	0,0000	0,0000	2	17	50,0	46,9
	0,6	0,7812	0,7169	0,0000	0,0029	0,0000	0,0005	3	17	50,0	49,0
	0,8	1,0191	0,8627	0,0000	0,0068	0,0000	0,0053	2	19	50,0	46,4
	1	0,8060	0,6353	0,0000	0,0008	0,0000	0,0000	0	16	50,0	49,7
		0,6955	**0,5821**	**0,0000**	**0,0060**	**0,0000**	**0,0015**	**9**	**91**	**250,0**	**235,3**
0,8											
	0,2	0,0679	0,0444	0,0000	0,0001	0,0000	0,0000	2	16	50,0	48,4
	0,4	0,1712	0,1384	0,0000	0,0006	0,0000	0,0002	1	19	50,0	46,9
	0,6	0,0713	0,0471	0,0000	0,0000	0,0000	0,0000	0	17	50,0	49,7
	0,8	0,2966	0,2686	0,2008	0,2008	0,2008	0,2008	0	15	49,0	48,8
	1	0,0829	0,0606	0,0000	0,0001	0,0000	0,0000	4	26	50,0	49,0
		0,1380	**0,1118**	**0,0402**	**0,0403**	**0,0402**	**0,0402**	**7**	**93**	**249,0**	**242,8**
1											
	0,2	0,0130	0,0028	0,0000	0,0000	0,0000	0,0000	1	35	50,0	49,4
	0,4	0,0123	0,0053	0,0000	0,0000	0,0000	0,0000	4	33	50,0	50,0
	0,6	0,0116	0,0054	0,0000	0,0000	0,0000	0,0000	4	27	50,0	49,7
	0,8	0,0103	0,0022	0,0000	0,0000	0,0000	0,0000	5	38	50,0	50,0
	1	0,0131	0,0057	0,0000	0,0000	0,0000	0,0000	4	35	50,0	50,0
		0,0121	0,0043	0,0000	0,0000	0,0000	0,0000	18	168	250,0	249,1
Total		**0,6591**	**0,5559**	**0,0080**	**0,0195**	**0,0080**	**0,0093**	**372**	**744**	**1249,0**	**1210,2**

TABLE 2 – COMPARISON WITH OPTIMUM RESULTS FOR INSTANCES WITH 25 JOBS

N	Average ivd%	Sum of n_btr_d
10	0,2518	315
15	0,2003	454
20	0,1877	582
25	0,1594	639
30	0,1494	692
40	0,1049	790
50	0,0953	845
75	0,0694	913
100	0,0582	944
250	0,0534	962
500	0,0511	966

TABLE 3 – IMPROVEMENT OF LOCAL SEARCH ON DISPATCHING RULE

	Average ivd_avg%		Average ivd_best%		Sum of n_btr_d		Average ivd+a_avg%		Average ivd+a_best%		Sum of n_btr_d+a	
N	**ILS**	**VG**	**ILS**	**VG**	**ILS**	**VG**	**ILS**	**VG**	**ILS**	**VG**	**ILS**	**VG**
10	0,5142	0,5142	0,5142	0,5142	414,0	414,0	0,2626	0,2626	0,2626	0,2626	133,0	133,0
15	0,5904	0,5904	0,5904	0,5904	580,0	580,0	0,3906	0,3906	0,3906	0,3906	214,0	214,0
20	0,6522	0,6522	0,6522	0,6522	719,0	718,9	0,4660	0,4660	0,4660	0,4660	315,0	314,9
25	0,7801	0,7799	0,7801	0,7801	765,0	764,9	0,6217	0,6215	0,6217	0,6217	368,0	367,8
30	0,7026	0,6994	0,7026	0,7025	809,0	808,0	0,5552	0,5520	0,5552	0,5552	401,0	397,6
40	0,6510	0,6403	0,6510	0,6498	878,0	873,0	0,5478	0,5371	0,5478	0,5465	506,0	492,4
50	0,7388	0,7186	0,7388	0,7315	920,0	916,0	0,6453	0,6251	0,6453	0,6380	570,0	532,5
75	0,6881	0,6552	0,6881	0,6715	955,0	952,9	0,6196	0,5866	0,6196	0,6030	719,0	635,0
100	0,7091	0,6555	0,7091	0,6713	967,0	963,1	0,6521	0,5985	0,6521	0,6143	788,0	668,8
250	0,6548	0,6130	0,6548	0,6182	969,0	969,0	0,6023	0,5605	0,6024	0,5657	937,9	800,8
500	0,5629	0,5278	0,5630	0,5294	968,0	967,0	0,5126	0,4774	0,5126	0,4790	959,0	856,1

TABLE 4 – COMPARISON OF ILS AND VG WITH DISPATCHING RULE WITH AND WITHOUT LOCAL SEARCH

TABLE 5 – COMPARISON OF N_EQL_D AND N_EQL_D+A WITH N_OPT

	Sum of n_eql_d		Sum of n_opt	Sum of n_eql_d+a		Sum of n_opt
N	**ILS**	**VG**	**DR**	**ILS**	**VG**	**DR+LS**
10	836	836	836	1117	1117	1117
15	670	670	670	1036	1036	1036
20	531	531,1	531	935	935,1	935
25	485	485,1	485	882	882,2	882
30	441	442	441	849	852,4	849
40	372	377	372	744	757,6	744
50	330	334	330	680	717,5	680
75	295	297,1		531	615	
100	283	286,9		462	581,2	
250	281	281		312,1	449,2	
500	282	283		291	393,9	
Total	4806	4823,2		7839,1	8337,1	

		Average ivd_avg%		Average ivd_best%		Sum of n_btr_d		Average ivd+a_avg%		Average ivd+a_best%		Sum of n_btr_d+a	
T	**R**	**ILS**	**VG**	**ILS**	**VG**	**ILS**	**VG**	**ILS**	**VG**	**ILS**	**VG**	**ILS**	**VG**
0.2													
	0.2	4,7905	4,1597	4,7905	4,3107	45	41,4	4,6042	3,9725	4,6042	4,1237	38	31,2
	0.4	0,4724	0,4724	0,4724	0,4724	3	3	0,4249	0,4249	0,4249	0,4249	1	1
	0.6	0,0000	0,0000	0,0000	0,0000	0	0	0,0000	0,0000	0,0000	0,0000	0	0
	0.8	0,0000	0,0000	0,0000	0,0000	0	0	0,0000	0,0000	0,0000	0,0000	0	0
	1	0,0000	0,0000	0,0000	0,0000	0	0	0,0000	0,0000	0,0000	0,0000	0	0
		1,0526	**0,9264**	**1,0526**	**0,9566**	**48**	**44,4**	**1,0058**	**0,8795**	**1,0058**	**0,9097**	**39**	**32,2**
0.4													
	0.2	1,0761	0,9816	1,0761	0,9846	50	50	0,9614	0,8668	0,9614	0,8699	43	34,5
	0.4	3,0298	2,6540	3,0298	2,8058	50	50	2,8362	2,4598	2,8362	2,6119	46	37,3
	0.6	2,7005	2,6831	2,7005	2,6967	47	46,7	2,4563	2,4389	2,4563	2,4525	33	31,6
	0.8	2,1805	2,1694	2,1805	2,1792	20	20	1,8383	1,8272	1,8383	1,8371	12	11,3
	1	0,0475	0,0475	0,0475	0,0475	2	2	0,0302	0,0302	0,0302	0,0302	1	1
		1,8069	**1,7071**	**1,8069**	**1,7428**	**169**	**168,7**	**1,6245**	**1,5246**	**1,6245**	**1,5603**	**135**	**115,7**
0.6													
	0.2	0,2323	0,1990	0,2324	0,2005	50	50	0,1733	0,1400	0,1733	0,1415	44	26,7
	0.4	0,4151	0,3859	0,4152	0,3949	50	50	0,3432	0,3140	0,3434	0,3230	47	30,5
	0.6	0,6040	0,5802	0,6041	0,5890	50	50	0,5643	0,5405	0,5644	0,5493	48	36,7
	0.8	1,1921	1,1195	1,1923	1,1497	50	50	1,1611	1,0885	1,1613	1,1186	49	44,6
	1	0,8010	0,7627	0,8011	0,7748	50	50	0,7609	0,7225	0,7610	0,7347	49	39,4
		0,6489	**0,6095**	**0,6490**	**0,6218**	**250**	**250**	**0,6005**	**0,5611**	**0,6007**	**0,5734**	**237**	**177,9**
0.8													
	0.2	0,0560	0,0500	0,0560	0,0510	50	50	0,0493	0,0432	0,0493	0,0443	47	35,2
	0.4	0,0453	0,0415	0,0454	0,0436	50	50	0,0389	0,0351	0,0390	0,0372	48	40,2
	0.6	0,0344	0,0334	0,0344	0,0340	50	50	0,0281	0,0272	0,0282	0,0277	45	39,9
	0.8	0,0206	0,0199	0,0206	0,0204	50	50	0,0158	0,0150	0,0158	0,0155	34	30,8
	1	0,0145	0,0140	0,0145	0,0143	50	50	0,0093	0,0089	0,0093	0,0092	42	37,6

	0,0342	**0,0318**	**0,0342**	**0,0327**	**250**	**250**	**0,0283**	**0,0259**	**0,0283**	**0,0268**	**216**	**183,7**
1												
0.2	0,0032	0,0032	0,0032	0,0032	50	50	0,0015	0,0015	0,0015	0,0015	39	39
0.4	0,0025	0,0025	0,0025	0,0025	50	50	0,0011	0,0011	0,0011	0,0011	35	34,7
0.6	0,0034	0,0034	0,0034	0,0034	50	50	0,0019	0,0018	0,0019	0,0018	29	28,7
0.8	0,0027	0,0027	0,0027	0,0027	50	50	0,0012	0,0012	0,0012	0,0012	32	31,9
1	0,0025	0,0025	0,0025	0,0025	50	50	0,0013	0,0013	0,0013	0,0013	26	25
	0,0029	**0,0029**	**0,0029**	**0,0029**	**250**	**250**	**0,0014**	**0,0014**	**0,0014**	**0,0014**	**161**	**159,3**
Total	**0,7091**	**0,6555**	**0,7091**	**0,6713**	**967**	**963,1**	**0,6521**	**0,5985**	**0,6521**	**0,6143**	**788**	**668,8**

TABLE 6 – COMPARISON OF IMPROVEMENTS VERSUS DISPATCHING RULE WITH AND WITHOUT LOCAL SEARCH, FOR INSTANCES WITH 100 JOBS

	Average ivw_avg%		Average ivw_best%	
N	**ILS**	**VG**	**ILS**	**VG**
10	0	0	0	0
15	0	0	0	0
20	0,0002	0,0002	0,0002	0,0002
25	0,0013	0,0011	0,0013	0,0013
30	0,0109	0,0076	0,0109	0,0109
40	0,0204	0,0090	0,0204	0,0191
50	0,0308	0,0101	0,0308	0,0234
75	0,0557	0,0218	0,0557	0,0387
100	0,0675	0,0126	0,0675	0,0289
250	0,0468	0,0037	0,0469	0,0091
500	0,0386	0,0023	0,0387	0,0039

TABLE 7 – IVW COMPARISON BETWEEN ILS AND VG

		Average ivw_avg%		Average ivw_best%	
T	**R**	**ILS**	**VG**	**ILS**	**VG**
0.2					
	0.2	0,4491	0,0133	0,4491	0,0304
	0.4	0,0000	0,0000	0,0000	0,0000
	0.6	0,0000	0,0000	0,0000	0,0000
	0.8	0,0000	0,0000	0,0000	0,0000
	1	0,0000	0,0000	0,0000	0,0000
		0,0898	**0,0027**	**0,0898**	**0,0061**
0.4					
	0.2	0,1037	0,0020	0,1037	0,0034
	0.4	0,1868	0,0084	0,1868	0,0164
	0.6	0,1940	0,0146	0,1940	0,0901
	0.8	0,0646	0,0411	0,0646	0,0552
	1	0,0000	0,0000	0,0000	0,0000
		0,1098	**0,0132**	**0,1098**	**0,0330**
0.6					
	0.2	0,0192	0,0003	0,0192	0,0003
	0.4	0,0263	0,0005	0,0263	0,0009
	0.6	0,0341	0,0024	0,0342	0,0079
	0.8	0,0531	0,0052	0,0540	0,0137
	1	0,0310	0,0048	0,0314	0,0083
		0,0327	**0,0026**	**0,0330**	**0,0062**
0.8					
	0.2	0,0055	0,0001	0,0056	0,0002

0.4	0,0020	0,0001	0,0021	0,0005
0.6	0,0006	0,0001	0,0006	0,0002
0.8	0,0004	0,0000	0,0004	0,0001
1	0,0006	0,0001	0,0006	0,0003
	0,0019	**0,0001**	**0,0019**	**0,0003**
1				
0.2	0,0000	0,0000	0,0000	0,0000
0.4	0,0000	0,0000	0,0000	0,0000
0.6	0,0000	0,0000	0,0000	0,0000
0.8	0,0000	0,0000	0,0000	0,0000
1	0,0000	0,0000	0,0000	0,0000
	0,0000	**0,0000**	**0,0000**	**0,0000**
Total	**0,0468**	**0,0037**	**0,0469**	**0,0091**

TABLE 8 – IVW COMPARISON BETWEEN ILS AND VG FOR INSTANCES WITH 250 JOBS

N	ILS	VG
10	0,0000	0,0000
15	0,0000	0,0001
20	0,0000	0,0006
25	0,0000	0,0029
30	0,0000	0,0069
40	0,0002	0,0146
50	0,0013	0,0273
75	0,0106	0,0612
100	0,0409	0,0947
250	0,8090	0,3524
500	6,3268	1,3457

TABLE 9 – RUNTIMES UNTIL BEST SOLUTION, IN SECONDS

		ILS	VG
T	**R**		
0.2			
	0.2	0,0937	0,0019
	0.4	0,0023	0,0013
	0.6	0,0014	0,0014
	0.8	0,0015	0,0015
	1	0,0015	0,0015
		0,0201	**0,0015**
0.4			
	0.2	0,4011	0,0038
	0.4	0,2083	0,0345
	0.6	0,1827	0,6467
	0.8	0,0414	0,0607
	1	0,0017	0,0018
		0,1670	**0,1495**
0.6			
	0.2	0,7473	0,0097
	0.4	1,1861	0,0071
	0.6	1,6607	0,1804
	0.8	3,0267	0,5081
	1	3,0880	0,5881
		1,9418	**0,2587**
0.8			
	0.2	1,5940	0,0594
	0.4	2,7506	0,3978
	0.6	1,5882	1,1422
	0.8	1,0591	0,6272
	1	1,2485	0,9618
		1,6481	**0,6377**
1			
	0.2	0,1417	0,7433
	0.4	0,2693	0,9800
	0.6	0,2168	0,7806
	0.8	0,2577	0,5967
	1	0,4540	0,4716
		0,2679	**0,7144**
	Total	**0,8090**	**0,3524**

TABLE 10 – RUNTIMES UNTIL BEST SOLUTION, IN SECONDS, FOR INSTANCES WITH 250 JOBS

7 References

Abdul-Razaq, T. S., C. N. Potts, and L. N. Van Wassenhove. 1990. A survey of algorithms for the single machine total weighted tardiness scheduling problem. *Discrete Applied Mathematics* 26 (2–3):235-253.

Allahverdi, A., C. T. Ng, T. C. E. Cheng, and M. Y. Kovalyov. 2008. A survey of scheduling problems with setup times or costs. *European journal of operational research* 187 (3):985-1032.

Behnamian, J., and M. Zandieh. 2011. A discrete colonial competitive algorithm for hybrid flowshop scheduling to minimize earliness and quadratic tardiness penalties. *Expert Systems with Applications* 38 (12):14490-14498.

Congram, R. K., C. N. Potts, and S. L. van de Velde. 2002. An iterated dynasearch algorithm for the single-machine total weighted tardiness scheduling problem. *INFORMS JOURNAL ON COMPUTING* 14 (1):52-67.

Dhingra, A. K. 2010. Multi-objective flow shop scheduling using metaheuristics, Mechanical Engineering, National Institute of Technology, Kurukshetra.

Du, J., and J. Y. T. Leung. 1990. Minimizing Total Tardiness on One Machine Is NP-Hard. *Mathematics of Operations Research* 15 (3):483-495.

Fisher, M. L. 1981. The Lagrangian Relaxation Method for Solving Integer Programming Problems. *Management Science* 27 (1):1-18.

Gupta, J. N. D. S., Edward F. 2006. Flowshop scheduling research after five decades. *European journal of operational research* 169 (3):699 -711.

Hoitomt DJ, L. P., Max E, Pattipati KR. 1990. Scheduling jobs with simple precedence constraints on parallel machines. *Control Systems Magazine, IEEE* 10 (2):34-40.

Johnson, S. M. 1954. *Optimal Two- and Three-stage Production Schedules with Setup Times Included*: Rand Corporation.

Kianfar, K., and G. Moslehi. A branch-and-bound algorithm for single machine scheduling with quadratic earliness and tardiness penalties. *Computers & Operations Research*.

Koulamas, C. 2010. The single-machine total tardiness scheduling problem: Review and extensions. *European journal of operational research* 202 (1):1-7.

Potts, C. N., and V. A. Strusevich. 2009. Fifty years of scheduling: a survey of milestones. *J Oper Res Soc* 60 (S1):S41-S68.

Potts, C. N., and L. N. Van Wassenhove. 1991. Single Machine Tardiness Sequencing Heuristics. *IIE Transactions* 23 (4):346-354.

Rahmani, K., and I. Mahdavi, 2012. A genetic algorithm for the single machine preemptive scheduling problem with linear earliness and quadratic tardiness penalties. *The International Journal of Advanced Manufacturing Technology*:1-8.

Schaller, J. 2004. Single machine scheduling with early and quadratic tardy penalties. *Computers & Industrial Engineering* 46 (3):511-532.

Schaller, J., and J. M. S. Valente. 2012. Minimizing the weighted sum of squared tardiness on a single machine. *Comput. Oper. Res.* 39 (5):919-928.

Sen, T., J. M. Sulek, and P. Dileepan. 2003. Static scheduling research to minimize weighted and unweighted tardiness: A state-of-the-art survey. *International Journal of Production Economics* 83 (1):1-12.

Sun, X., J. S. Noble, and C. M. Klein. 1999. Single-machine scheduling with sequence dependent setup to minimize total weighted squared tardiness. *IIE Transactions* 31 (2):113-124.

Taguchi, G. 1986. *Introduction to Quality Engineering : Designing Quality into Products and Processes*: Tokio, Japan : Asian Productivity Organization.

Thomalla, C. S. 2001. Job shop scheduling with alternative process plans. *International Journal of Production Economics* 74 (1–3):125-134.

Valente, J., and M. Moreira. 2009. Greedy randomised dispatching heuristics for the single machine scheduling problem with quadratic earliness and tardiness penalties. *The International Journal of Advanced Manufacturing Technology* 44 (9):995-1009.

Valente, J., M. Moreira, A. Singh, and R. Alves. 2011. Genetic algorithms for single machine scheduling with quadratic earliness and tardiness costs. *The International Journal of Advanced Manufacturing Technology* 54 (1):251-265.

Valente, J. M. S. 2007. Heuristics for the single machine scheduling problem with early and quadratic tardy penalties. *European Journal of Industrial Engineering* 1 (4):431-448.

Valente, J. M. S. 2008a. Beam Search Heuristics for the Single Machine Scheduling Problem with Linear Earliness and Quadratic Tardiness Costs. *Asia-Pacific Journal of Operational Research (APJOR)* 26 (3):319-339.

Valente, J. M. S. 2008b. An exact approach for the single machine scheduling problem with linear early and quadratic tardy penalties. *Asia-Pacific Journal of Operational Research* 25 (2):169-186.

Valente, J. M. S.. 2010. Beam search heuristics for quadratic earliness and tardiness scheduling. *J Oper Res Soc* 61 (4):620-631.

Valente, J. M. S., and R. A. F. S. Alves. 2008. Heuristics for the single machine scheduling problem with quadratic earliness and tardiness penalties. *Computers & Operations Research* 35 (11):3696-3713.

Valente, J. M. S., and J. F. Gonçalves. 2009. A genetic algorithm approach for the single machine scheduling problem with linear earliness and quadratic tardiness penalties. *Computers & Operations Research* 36 (10):2707-2715.

Valente, J. M. S., and J. E. Schaller. 2010. Improved heuristics for the single machine scheduling problem with linear early and quadratic tardy penalties. *European Journal of Industrial Engineering* 4 (1):99-129.

Valente, J. M. S., and J. E. Schaller. 2012. Dispatching heuristics for the single machine weighted quadratic tardiness scheduling problem. *Computers & Operations Research* 39 (9):2223-2231.

Valente, J. M. S. G., J.F.; Alves, A.F.S. 2006. A Hybrid Genetic Algorithm for the Early/Tardy Scheduling Problem. *Asia-Pacific Journal of Operational Research* 23 (3):393-405.

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

Wagner, B. J., D. J. Davis, and H. V. Kher. 2002. The Production of Several Items in a Single Facility with Linearly Changing Demand Rates. *Decision Sciences* 33 (3):317-346.