Heuristics for a multi-machine multi-objective job scheduling problem with smoothing costs

Proceeding for the GOL 2012 conference

Jean Respen
HEC - University of Geneva
Switzerland
jean.respen@unige.ch

Nicolas Zufferey*
HEC - University of Geneva
Switzerland
nicolas.zufferey-hec@unige.ch

Edoardo Amaldi
DEI - Politecnico di Milano
Italy
amaldi@elet.polimi.it

ABSTRACT

We propose a new multi-objective job scheduling problem on non-identical machines involving job and machine dependent setup costs and times, as well as smoothing costs. Smoothing issues are very important in several settings, such as for example car production, since they allow to balance resource utilization over an assembly line. In this paper, we describe the problem, give a mixed integer linear programming formulation, and propose several heuristics: three greedy procedures, two descent approaches, and a tabu search. Experiments, performed on realistic and challenging instances with up to 500 jobs and 8 machines, show that tabu search is a powerful method: it gives the best results for the large instances and is very competitive on the small instances.

General Terms

Metaheuristics, job scheduling, multi-resource

1. INTRODUCTION

Multi-objective scheduling problems often involve minimizing the makespan while considering setup costs and times. Various approaches have been proposed to tackle makespan minimization in the literature (see [8] for a good reference book). For a survey on scheduling techniques accounting for setup issues, the reader is referred to [1].

Nowadays, new constraints, known as *smoothing constraints*, are attracting a growing attention in the area of job scheduling (see the survey on smoothing constraints known as "balancing in assembly line" in [2]) and in particular for car sequencing problems, where cars must be scheduled before production in an order respecting many constraints (colors, options, due dates, etc.), while avoiding overloading some important resources. As an example, if the yellow cars with air-conditioning are scheduled first, the unlucky costumer who ordered a grey car without air-conditioning may wait for a long time. For the car plant, balancing between optional equipments and colors allows to respect customers deadlines and to prevent overloading resources (machines or employees), which has an impact on cost reduction. As mentioned in [12], there is a complex tradeoff at the core of many practical scheduling problems, which involves balancing the benefits of long production runs of a similar product against the costs of completing work before it is needed (and potentially causing other work to be tardy).

Part of the above problem was the subject of the ROADEF 2005 Challenge (`http://challenge.roadef.org/2005/en/`) proposed by the car manufacturer *Renault*, where instances involve hundreds of cars and thus no exact algorithm can be competitive. In the *Renault* problem, car families are defined so that two cars of the same family contain the same optional equipments. Each car option i is associated with a p_i/q_i ratio constraint, meaning that at most p_i vehicles with option i can be scheduled in any subsequence of q_i vehicles, otherwise a penalty occurs. Another goal consists in minimizing the number of color changes in the production sequence. Thus, the overall objective is to minimize a weighted function involving the numbers of ratio constraint violations and color changes. A simplified version of this problem was proved to be NP-hard in [6], and a survey of the above challenge can be found in [11]. The winner team proposed a local search algorithm called very fast local search (VFLS), and the second team proposed a variable neighborhood search (VNS) in combination with an iterated local search (ILS) procedure. The VFLS heuristic is described in [4] and is the conjunction of a standard local search with a tuned transformation step. [9] described a set of heuristics, based on the paradigms of VNS and ILS metaheuristics, along with intensification and diversification strategies. The tabu search proposed in [3] was ranked 7th.

In this work, we define and investigate a multi-objective production problem (P) with smoothing costs inspired by the *Renault* problem. Unlike the latter problem, we consider several non-identical machines (resources), eligibility constraints (a job cannot necessarily be performed on all the machines), and setup constraints. We aim at minimizing in a lexicographic way the overall makespan, setup costs and smoothing costs.

The remainder of the paper is organized as follows. In Sec-

*Corresponding author.

tion 2, we describe the problem (P), pointing out the differences with respect to the *Renault* problem, and we give a mixed integer linear programming (MILP) formulation. An interpretation as a variant of the vehicle routing problem (VRP) is mentioned in Section 3. Since the MILP formulation is very challenging even for small-size instances, in Section 4 we propose several heuristics, ranging from simple greedy procedures to a tabu search metaheuristic. Computational results are reported and discussed in Section 5. Finally, Section 6 contains some concluding remarks along with possible future work.

2. PROBLEM AND MILP FORMULATION

In the considered problem (P), a set of n jobs have to be scheduled on a set of m non-identical parallel machines. Each job j belongs to one of the g available families and has a processing time p_j^i depending on the machine i. For each pair of job j and machine i, the eligibility of j on i is specified by the binary parameter u_j^i. The goal consists in minimizing a cost function taking into account makespan, smoothing costs and setup costs. A lexicographic approach is considered with the following priority order: makespan $>$ smoothing costs $>$ setup costs. As proposed below in Equation (1), the lexicographic optimization is achieved via coefficients $\alpha > \beta > \gamma > 0$, where the makespan C_{max} also involves a scaling parameter $\omega > 0$. A small value of C_{max} usually indicates a high occupancy rate of the machines, and as a consequence, the production system will be available sooner for future commands. Note that C_{max}^i denotes the completion time of machine i.

The setups are job and machine dependent: $c_{jj'}^i$ (resp. $s_{jj'}^i$) is the setup cost (resp. time) encountered to prepare machine i to perform job j' after job j. There are two types of setups: major (if the involved jobs belong to two different families) and minor (otherwise). Minor setups can be considered as a small encountered time/cost if a technician needs to slightly modify the configuration of the machine to perform the next job. On the opposite, a major setup occurs when external technicians (working at a higher hourly rate and bill for transportation) or a significant machine transformation (e.g., its capacity, its reprogramming) are required.

Consider the following sets of binary decision variables: 1) for every machine i and pair of jobs $\{j, j'\}$, $x_{jj'}^i = 1$ if job j is followed by job j' on machine i, and 0 otherwise; 2) for every machine i and triple of jobs $\{j, j', j''\}$, $y_{jj'j''}^i = 1$ if jobs j, j', j'' are consecutively scheduled on machine i, 0 otherwise; 3) for every machine i and job j, $z_j^i = 1$ if job j is scheduled on machine i, and 0 otherwise.

In addition, let r_j^i be the position index of job j if it is scheduled on machine i. For every triple of jobs $\{j, j', j''\}$, $f_{jj'j''} = 1$ if j, j', j'' are of the same family, and 0 otherwise. Finally, let k_f be the smoothing cost associated with the family f to which job j belongs, which is encountered only if $f_{jj'j''} = 1$ and $y_{jj'j''} = 1$.

Using these variables, the problem can be formulated as the following mixed integer linear program: minimize

$$\alpha \cdot \sum_i \sum_{j,j'} c_{jj'}^i \cdot x_{jj'}^i + \beta \cdot \sum_i \sum_{j,j',j''} k_j \cdot f_{jj'j''} \cdot y_{jj'j''}^i + \gamma \cdot \omega \cdot C_{max} \quad (1)$$

subject to the following constraints:

$$\sum_j p_j^i \cdot z_j^i + \sum_{j,j'} s_{jj'}^i \cdot x_{jj'}^i \leq C_{max} \quad \forall i \quad (2)$$

$$(2 - u_j^i - u_{j'}^i) \cdot x_{jj'}^i = 0 \quad \forall i, j, j' \quad (3)$$

$$z_j^i \leq u_j^i \quad \forall i, j \quad (4)$$

$$z_j^i + z_{j'}^i \geq 2 \cdot x_{jj'}^i \quad \forall i, j, j' \quad (5)$$

$$2 \cdot y_{jj'j''}^i \leq (x_{jj'}^i + x_{j'j''}^i) \cdot f_{jj'j''} \quad \forall i, j, j', j'' \quad (6)$$

$$(x_{jj'}^i + x_{j'j''}^i) \cdot f_{jj'j''} - 1 \leq y_{jj'j''}^i \quad \forall i, j, j', j'' \quad (7)$$

$$\sum_j z_j^i - 1 = \sum_{j,j'} x_{jj'}^i \quad \forall i \quad (8)$$

$$\sum_{j'} x_{jj'}^i \leq 1 \quad \forall i, j \quad (9)$$

$$\sum_j x_{jj'}^i \leq 1 \quad \forall i, j' \quad (10)$$

$$z_j^i \leq r_j^i \leq n \cdot z_j^i \quad \forall i, j \quad (11)$$

$$r_{j'}^i \geq (r_j^i + 1) - n \cdot (1 - x_{jj'}^i) \quad \forall i, j, j' \quad (12)$$

$$x_{jj'}^i + x_{j'j}^i \leq 1 \quad \forall i, j, j' \quad (13)$$

$$\sum_i z_j^i = 1 \quad \forall j \quad (14)$$

$$0 \leq y_{jj'j''}^i \leq 1 \quad \forall i, j, j', j'' \quad (15)$$

$$x_{jj'}^i, z_j^i \in \{0, 1\} \quad \forall i, j, j' \quad (16)$$

Constraints (2) guarantee a correct makespan computation. Constraints (3) and (4) ensure that eligibility is satisfied on each machine. Constraints (5) allow two jobs to be scheduled consecutively only if they are scheduled on the same machine, while Constraints (6) and (7) ensure that three jobs can be scheduled consecutively only if all the three jobs are scheduled consecutively two at a time. Constraints (8) guarantee that the correct number of jobs will be consecutively scheduled, while Constraints (9), (10) and (13) ensure the correct number of consecutive jobs. Constraints (11) and (12) are the subtour elimination constraints given in [10]. Constraints (14) guarantee that each job is scheduled exactly once. Finally, fractional values are possible in Constraints (15) because of the objective function (1) and of the Constraints (6) and (7). Clearly, relaxing the integrality of the $y_{jj'j''}^i$ variables makes the MILP formulation easier.

3. GRAPH INTERPRETATION

Assuming that setup costs are proportional to setup times (which is realistic from a practical standpoint), our production problem can be easily interpreted as a variant of the well-known Vehicle Routing Problem (VRP) on a graph.

Let the n jobs be represented by n vertices labeled $1, \ldots, n$ of a complete directed graph $G = (V, E)$. A color is associated with each job family. Since each vertex belongs to a unique family, it has a single color and the vertices can be grouped by color. Each vertex is connected to all the others vertices by an arc. Arcs are used instead of edges as setup times are not necessarily symmetric. If two vertices are of the same family, then the arc linking them is colored with the family color and the corresponding setup time is minor. Otherwise, the arc is uncolored and the setup time is major.

Two vectors are associated with each arc (j, j'): a processing time vector $P_{j'} = (p^1_{j'}, p^2_{j'}, \ldots, p^m_{j'})$ and a setup time vector $S_{jj'} = (s^1_{jj'}, s^2_{jj'}, \ldots, s^m_{jj'})$. In addition, a virtual uncolored vertex labeled 0 is defined. With each arc $(0, j)$ are associated a processing time vector $P_j = (p^1_j, p^2_j, \ldots, p^m_j)$, and a setup time vector S_{0j} where $s^i_{0j} = 0$ for each machine i. With each arc $(j, 0)$ are associated a processing time vector P_0 where $p^i_0 = 0$ for each machine i, and a setup time vector S_{j0} where $s^i_{j0} = 0$ for each machine i.

In the VRP, n clients have to be visited exactly once by a set of m vehicles so as to optimize an objective function (see [7] for a recent survey). Here the clients correspond to the jobs, the vehicles to the machines, and the sequence of vertices visited by vehicle i to the sequence of jobs executed by machine i. Given the graph G, the objective is to find m circuits (starting from and ending at vertex 0, representing a depot), such that every vertex is visited exactly once. A smoothing cost is paid each time a circuit visits consecutively three or more vertices within the same color class (and is thus a special case of the p_i/q_i ratio of the *Renault* problem, where $p_i = 2$ and $q_i = 3$).

An illustration with $n = 8$ and $m = 2$ is given in Figure 1, where edges are used instead of arcs for the sake of simplicity (the graph is not drawn complete for the same reason). Vertices are grouped by color classes, and dashed edges connecting vertices of different color classes are uncolored. On the contrary, colored plain edges connect vertices of the same color class. A double weighted edge $[1, 2]$ is indicated as an example: job 1 has a processing time of 110 on machine 1 and 135 on machine 2; a minor setup of 6 occurs when finishing job 1 and preparing machine 1 for job 2, and the corresponding setup for the second machine is 8. Figure 2 shows a feasible solution for the instance of Figure 1. Note that the circuit visiting the first color class encounters a smoothing cost because it consecutively visits the three vertices with that color.

4. HEURISTICS

In this section, we describe different greedy algorithms for (P), as well as various local search techniques, namely a descent method, a descent with a learning process, and a tabu search. All theses methods will be compared according to a time limit of t seconds. If a method stops before t, it is restarted as long as t is not reached, and the provided solution is the best one generated within t seconds. For the greedy algorithm, each restart occurs when a complete solution is built, whereas for a descent method, each restart occurs when a local optimum is found.

In the **greedy procedures**, to build a feasible solution step by step from scratch, we *select* at each step an unscheduled job and *insert* it at the best position (i.e., leading to the least augmentation of the objective function value), while respecting eligibility and setup constraints. Different job selection criteria are considered: 1) *randomly* among all the unscheduled jobs; 2) the least *flexible* job first (the flexibility of a job is defined as the number of machines it can be performed on); 3) *exhaustive* search (each insertion is tested for each non performed job). We propose three greedy procedures for (P): GrR which is the greedy algorithm based on the random selection process, GrF based on the flexibility strategy, and GrE based on the exhaustive strategy. Ties often occur and are always randomly broken.

A **basic local search** starts from an initial solution x_0 and then successively generates a sequence of solutions $x_1, x_2, \ldots$ in the search space such that $x_{r+1} \in N(x_r)$, where $N(x_r)$ denotes the set of *neighbor* solutions of x_r. To generate x_{r+1} from solution x_r, the algorithm performs a specific modification, called a *move*. The process is stopped when a specific criterion is met (e.g. a time limit). In a *descent* algorithm, the best move is performed at each iteration and the process stops when the first local optimum is found. In a *tabu search*, when a move is performed from x_r to x_{r+1}, the reverse move is forbidden (tabu) for *tab* (parameter) iterations. The reader interested in more information on local search techniques, and more generally on metaheuristics, is referred to [5].

The proposed **local search techniques for (P)** are: Des (a descent), $LDes$ (a descent with a learning process), and TS(a tabu search). In Des as well as in $LDes$, the best move out of a portion of $r\%$ of all possible moves is performed at each iteration, whereas in TS, $r\%$ of the non tabu neighbor solutions are generated and the best one among this random sample is selected. Preliminary experiments showed that $r = 10\%$ is a an appropriate choice. The specificities of these three methods are now presented.

In all the proposed local search algorithms, a **move** consists in reinserting a job somewhere else in the solution. More precisely, a job can be moved to a different position on the same machine, or it can be moved from a machine to another (as long as the eligibility constraint is satisfied). In TS, when a job has been moved, it is tabu to move it again for *tab* iterations, where preliminary experiments showed that it is reasonable to choose the integer *tab* uniformly at random in $[\frac{n}{25}, \frac{n}{13}]$. Moreover, a second tabu structure is used: when a job j, initially positioned between jobs j' and j'', is reinserted somewhere else in the schedule, it is then tabu to move j back between j' and j'' for $2 \cdot tab$ iterations (as this second tabu structure is less restrictive than the first one, the tabu duration is set to a larger value). Note that when a move improves the best ever visited solution, its two associated tabu durations are increased by a random integer uniformly generated within interval $[\frac{n}{25}, \frac{n}{15}]$.

An important issue is the way **initial solutions** are generated for the local search heuristics. In Des and in TS, the initial solution is always generated with GrE. $LDes$ is a two-phase algorithm. In the first phase, the three greedy algorithms are alternately used if the running time is not above $\frac{t}{2}$ seconds. At the end of such a learning phase, a weight is assigned to each of the three greedy procedures, and such weights are proportional to the average quality of the resulting solutions (i.e., the ones obtained at the end of the descent process). In the second phase, the next greedy algorithm to apply is randomly selected based on those weights. Thus, the greedy procedure leading in average to the best results in the first phase has more chance to be selected in the second phase.

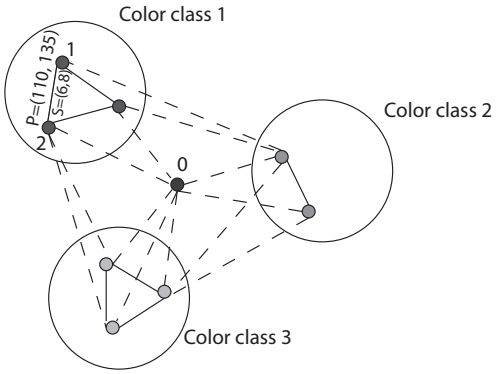

Figure 1: Graph representation of the problem

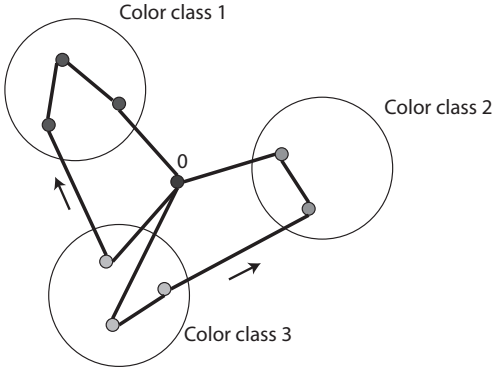

Figure 2: A possible feasible solution for Figure 1

5. COMPUTATIONAL RESULTS

In this section, we first describe the instances used in the computational experiments and then summarize the results and compare those obtained with all the proposed heuristics. Tests were performed on an Intel Quad-core i7 @ 3.4 Ghz with 8 Go DDR3 of ram memory, and the time limit t is 3600 seconds. As TS is not a method with restarts, its results are averaged over five runs.

As (P) is a new problem, we generated dedicated instances. Each instance is characterized by: the number of jobs $n \in \{100, 200, 300, 400, 500\}$, the integer number of machines $m \in [3, 8]$, the number of families $g = \max\{\lceil 0.02 \cdot n \rceil, 2\}$, the family identifier f_j for each job j, the processing times p_j^i of each job j on each machine i uniformly generated within $[100, 200]$, the setup costs s_j^i uniformly generated within $[30, 50]$ for major setups and within $[5, 10]$ for minor setups (we choose $c_j^i = s_j^i$), and the smoothing cost k_f of family f uniformly generated within $[40, 60]$. To ensure the respect of the lexicographic order, we used $\alpha = 1, \beta = 1,000, \gamma = 1,000,000$. In addition, we set $\omega = 1$.

First, we have tested the MILP formulation (1)-(16) with CPLEX 12.4 using AMPL. The problem turns out to be challenging since the formulation does not allow to solve to optimality instances with 30 jobs and 4 machines within a time limit of 10 hours. For instances with 20 jobs and 5 machines, CPLEX finds optimal solutions within 145 minutes, while tabu search is able to do so within 3 minutes.

The computational results obtained with the heuristics on larger instances are summarized in Table 1. The first two columns indicate the values of n and m (observe that there are three instances per couple (n, m)). The third column ($f^\star$) corresponds to the objective function value of the best solution ever found by any of the algorithms. The following column reports the percentage gap between the solution of GrR and $f^\star$. The next columns provide the same information for the other methods. The average results reported in the last line of Table 1, show that TS performs better in terms of average solution quality than the other algorithms and suggest the following order of decreasing solution quality: TS, $LDes$, Des, GrE, GrR, GrF. Not surprisingly the greedy procedures are the fastest ones. For example, GrE needs less than a second if $n = 100$ and less than two minutes if $n = 500$. It is interesting to notice that even for a complex problem like (P), greedy algorithms are still good enough for small instances. On average, it however appears that GrR and GrF are significantly outperformed by the other methods. The best greedy procedure is clearly GrE, which confirms that it is a good choice to use it to generate initial solutions for local search algorithms. This also shows that the selection of the next job to schedule has an important impact on the final solution quality. For larger instances, local search techniques tend to give the best results. It seems that the learning process of $LDes$ is relevant, as it slightly improves Des. Finally, TS finds good-quality solutions even for small instances and is likely to find the best solutions for large instances. Note that TS finds the

best solutions for 22 out of the 30 instances (see the bold numbers), and has a gap above 1% only for four instances. Since, unlike for the other heuristics, the results of TS are averaged over five runs, it can happen that a bold number in the TS column is not the smallest number of the corresponding line. In summary, TS is a good alternative when the solution quality is more important than speed, for example when important savings can be achieved with only a slight improvement in solution quality.

Table 1: Computational results

n	m	$f^\star$	GrR	GrE	GrF	Des	$LDes$	TS
100	3	4,328,786,571	5.45	1.18	6.39	0.03	**0.00**	1.09
100	3	4,245,912,803	5.83	0.53	5.47	0.56	0.34	**0.31**
100	3	4,426,440,790	5.11	0.08	5.11	0.09	0.09	**0.12**
100	4	3,391,950,505	5.53	**0.00**	20.29	**0.00**	**0.00**	4.14
100	4	3,092,572,911	6.71	0.73	6.77	0.74	0.72	**0.44**
100	4	3,191,912,833	7.55	0.17	7.72	0.17	0.17	**0.35**
200	4	6,590,860,334	10.09	2.39	8.57	0.03	**0.00**	0.78
200	4	6,498,914,381	10.15	0.53	10.08	0.08	0.18	**0.59**
200	4	7,393,134,327	1.83	3.26	0.95	0.30	0.13	**0.32**
200	5	5,379,374,024	9.86	10.96	8.37	1.86	2.11	**0.59**
200	5	5,324,665,984	9.77	7.14	8.55	2.74	3.29	**0.43**
200	5	5,069,039,647	11.15	2.55	10.49	2.54	2.54	**0.47**
300	5	7,599,797,593	11.88	9.13	12.31	9.48	9.98	**0.49**
300	5	8,807,918,149	8.02	4.18	0.92	**0.00**	0.19	1.06
300	5	7,681,543,177	13.15	0.82	23.28	0.73	0.84	**0.22**
300	6	6,020,430,266	13.61	0.15	12.30	**0.00**	0.19	0.36
300	6	6,776,973,326	12.06	7.58	8.41	20.26	**0.00**	2.47
300	6	7,497,225,254	12.06	2.92	22.08	0.45	0.50	**0.32**
400	6	8,145,401,006	15.14	0.23	15.24	0.05	0.17	**0.29**
400	6	11,859,026,355	10.86	0.42	15.15	**0.00**	0.11	0.31
400	6	9,037,098,794	12.73	1.47	8.42	0.29	0.76	**0.36**
400	7	7,454,156,110	13.48	5.63	11.76	5.32	5.49	**0.46**
400	7	7,116,299,800	13.48	7.49	13.62	2.72	2.41	**0.28**
400	7	6,983,854,842	16.71	0.66	16.97	**0.00**	0.10	0.39
500	7	9,029,251,346	15.40	1.72	16.64	1.31	1.34	**0.29**
500	7	9,042,559,903	13.64	1.42	14.76	0.59	1.06	**0.12**
500	7	9,058,870,658	13.01	5.49	12.78	2.87	2.94	**0.54**
500	8	7,691,897,956	15.54	8.99	16.63	2.46	2.30	**0.28**
500	8	7,448,592,060	15.83	0.50	15.55	0.47	0.21	**0.09**
500	8	7,469,847,076	15.39	1.76	16.87	0.73	0.51	**0.20**
Avg			11.03	3.00	11.75	1.90	1.29	0.60

6. CONCLUSIONS

We proposed a new multi-machine multi-objective job scheduling problem with interesting applications, for example in the car industry. The problem includes the following realistic features: jobs of various families, different machines (or production resources), makespan minimization, machine eligibility, machine and job dependent setup times and costs, as well as smoothing costs. We gave a MILP formulation and developed and compared several heuristics for this problem, ranging from simple greedy procedures to a tabu search metaheuristic. When the number of jobs increases, tabu search tends to provide the best solutions and is very competitive in terms of solution quality and computing time ratio.

Future work includes an improved tabu search algorithm with diversification and intensification mechanisms, as well as additional metaheuristics (as genetic or hybrid algorithms). Experiments on special cases of the problem with available benchmark instances would also be interesting since they would allow comparison of the proposed heuristics with existing ones.

7. REFERENCES

[1] A. Allahverdi, C. Ng, T. Cheng, and M. Kovalyov. A survey of scheduling problems with setup times or costs. *European Journal of Operational Research*, 187(3):985 – 1032, 2008.

[2] C. Becker and A.Scholl. A survey on problems and methods in generalized assembly line balancing. *European Journal of Operational Research*, 168(3):694 – 715, 2006.

[3] J.-F. Cordeau, G. Laporte, and F. Pasin. Iterated tabu search for the car sequencing problem. *European Journal of Operational Research*, 191(3):945 – 956, 2008.

[4] B. Estellon, F. Gardi, and K. Nouioua. Two local search approaches for solving real-life car sequencing problems. *European Journal of Operational Research*, 191(3):928 – 944, 2008.

[5] M. Gendreau and J.-Y. Potvin. *Handbook of Metaheuristics*. International Series in Operations Research & Management Science. Springer, 2010.

[6] I. P. Gent. Two results on car-sequencing problems. *Report APES-02-1998*, 1998.

[7] G. Laporte. Fifty years of vehicle routing. *Transportation Science*, 43(4):408 – 416, Nov. 2009.

[8] M. Pinedo. *Scheduling: Theory, Algorithms, and Systems, third edition*. Prentice Hall, 2008.

[9] C. C. Ribeiro, D. Aloise, T. F. Noronha, C. Rocha, and S. Urrutia. A hybrid heuristic for a multi-objective real-life car sequencing problem with painting and assembly line constraints. *European Journal of Operational Research*, 191(3):981 – 992, 2008.

[10] H. D. Sherali and P. J. Driscoll. On Tightening the Relaxations of Miller-Tucker-Zemlin Formulations for Asymmetric Traveling Salesman Problems. *Operations Research*, 50(4):656 – 669, 2002.

[11] C. Solnon, V. Cung, A. Nguyen, and C. Artigues. The car sequencing problem: Overview of state-of-the-art methods and industrial case-study of the ROADEFŠ 2005 challenge problem. *European Journal of Operational Research*, 191(3):912 – 927, 2008.

[12] S. Webster, P. D. Jog, and A. Gupta. A genetic algorithm for scheduling job families on a single machine with arbitrary earliness/tardiness penalties and an unrestricted common due date. *International Journal of Production Research*, 36 (9):2543 – 2551, 1998.