Heterogeneous workforce in job shop scheduling

Alexander J. Benavides
Instituto de Informática,
Universidad Federal do
Rio Grande do Sul, Brazil
Email: ajbenavides@inf.ufrgs.br

Marcus Ritt
Instituto de Informática,
Universidad Federal do
Rio Grande do Sul, Brazil
Email: marcus.ritt@inf.ufrgs.br

Cristobal Miralles
ROGLE, Departamento de
Organización de Empresas,
Universitat Politècnica de València, Spain
Email: cmiralles@omp.upv.es

***Abstract*—We introduce a new problem named Heterogeneous Job Shop Scheduling Problem (Het-JSSP) that consists of two subproblems: the assignment of workers to the workstations, and the interrelated job shop scheduling problem. Solving the assignment of workers simultaneously increases the difficulty of finding an optimal schedule. We introduce a mathematical model that extends a job shop model to admit the assignment of heterogeneous workers. We also present a construction heuristic based on a random assignment of workers and the shifting bottleneck procedure. Computational tests find good solutions within a short time. This problem was motivated by Sheltered Work centers for Disabled, where the heterogeneity of the workers' capabilities is greater, but production managers of any company that aim to overcome the traditional standardized vision of the workforce may use this approach to maintain high productivity levels while respecting the paces of their heterogeneous workers.**

Keywords*—Scheduling, Integer Programming, Heuristics, Job shop, Heterogenous Workers

I. Introduction

The International Labour Organization estimates that ten percent of the world population (i.e. 700 million people) has some disability, 500 million of them are in working age and suffering a higher unemployment rate than other citizens. Many governments are implementing policies for the integration of persons with disabilities in the workforce, such as reserving a percentage of jobs, or creating Sheltered Work centers for Disabled (SWDs). Spain and other countries have successfully adopted SWDs to facilitate transition jobs for the integration of disabled workers and also as stable workplaces. Even when SWDs receive some institutional support, they need to be efficient and competitive to survive and to grow in the real labor market. They also need to ensure the social integration of their workers considering their limitations and developing their capacities and capabilities [1].

Many approaches and tools of the Operations Research and Management Science (OR/MS) area standardize the processing time of the operation executed in a machine disregarding the skills of the worker that performs it. This unrealistic assumption may cause planning and control problems. For example, production managers need to compensate manually the deviations of the workers performance, or just use a worst case scenario, since their information systems ignore the skills heterogeneity of the workers. This leads to adopt suboptimal solutions and complicates the verification of indicators because the defined scenarios are different from the reality. It would be more reliable to take into account the workers heterogeneity, and this would allow to feed the planning/scheduling process with more realistic data. The OR/MS area is continuously proposing approaches for Design, Planning & Control of productive systems that admit workers heterogeneity, for example by using a stochastic model of processing times [2], or by categorizing skilled and naive workers [3]. Another example is the Assembly Line Worker Assignment and Balancing Problem (ALWABP), that was originally motivated by the workforce diversity of SWDs assembly lines. It focuses on the heterogeneity of the workers with variable processing times, and even considers possible incompatibilities due to workers disabilities, defining a new set of realistic hypotheses. Since the initial paper of Miralles et al. [4], other authors have contributed to this problem by proposing other extensions and methods to solve it [1], [5]–[8]. More recently, this heterogeneous vision has been extended to other scenarios, such as the heterogeneous flow shop scheduling problem (Het-FSSP) [9], that integrates the scheduling problem with the heterogeneity and incompatibilities of the workers on their workstations in a flow shop.

A. Contribution and outline of the paper

Our literature research did not find any reference for solving the worker assignment and job shop scheduling simultaneously when considering the workers heterogeneity. When the workers are heterogeneous, the processing times of an operation performed on a machine will also depend on the worker assigned to it. In the job shop literature, the term "machine" refers to a work center in which a process may be performed automatically or manually. In this scenario, the solution of the problem must include the optimal allocation of the workers besides the usual schedule of jobs that optimizes a certain objective function. Solving the assignment of workers simultaneously increases the number of possible schedules of the normal job shop problem. Thus, advanced methods are necessary to find an optimal schedule that corresponds to an optimal worker allocation. Even when SWDs motivated the initial scenario of this research, this paper aims to provide new models and approaches for ordinary companies that take into account the natural human diversity of workers.

Section II introduces the job shop scheduling problem and reviews the literature. Section III exemplifies the heterogeneous workers assignment and how they involve the job shop scheduling subproblem. In Section IV presents

Table I. An instance of the JSSP.

	Operations Machine (processing time)			
J_1	M_4 (2)	M_2 (2)	M_3 (2)	M_1 (3)
J_2	M_2 (2)	M_1 (1)	M_4 (3)	M_3 (2)
J_3	M_2 (2)	M_4 (2)	M_1 (1)	M_3 (1)
J_4	M_2 (3)	M_1 (1)	M_3 (3)	M_4 (1)

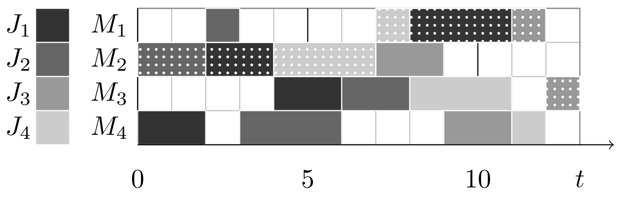

Figure 1. An optimal schedule for the JSSP instance of Table I.

a mathematical model for the situation of heterogeneous workers in a job shop problem, and in Section V we propose a multi-start search method. Section VI present a set of instances for the new problem and computational results from standard solvers and the proposed heuristic. Finally, we analyze and discuss the results and conclude in Section VII.

II. The job shop scheduling problem

The job shop scheduling problem is defined as follows. Given a set of jobs $\mathcal{J} = \{J_1, \dots, J_n\}$, and machines $\mathcal{M} = \{M_1, \dots, M_m\}$, each job J_j consisting of a sequence of m operations $\{o_{1j}, \dots, o_{mj}\}$, each operation o_{ij} needs to be processed on one machine $M(o_{ij}) \in \mathcal{M}$ with processing time p_{ij} without interruptions. Each job must be processed on the machines in its own processing order. Operations of the same job cannot be processed in parallel, and the machines can execute only one operation at a time. The most common objective is to find a schedule that minimizes the maximum completion time among operations or makespan $C_{\max}$.

Table I presents an example instance of a job shop scheduling problem. For each operation, the table shows the machine that processes it, and the processing times in parentheses. Figure 1 shows an optimal schedule for this instance with a makespan of 13 time units. The makespan of a schedule is defined by a *critical path*, which is a sequence of consecutive operations on the same machine or of the same job that leads to the longest overall duration. In the example of the schedule of Figure 1, the critical path is given by the dotted operations.

This problem is not only NP-hard, but it is considered one of the most computationally challenging combinatorial optimization problems [10]. Early approaches to solve it include exact methods such as branch and bound [11]–[14] that were effective only with small instances. In large instances, the computational time of enumerative methods grows exponentially. For this reason, the main efforts nowadays are concentrated in heuristic and meta-heuristic methods and their hybridization. These methods include: shifting bottleneck [15]–[17], simulated annealing [18]–[20], tabu search [21]–[24], genetic algorithms [25]–[28], greedy randomized adaptive search procedure [29] and [30], particle swarm optimization [31] and [32], ant colony optimization [33], artificial bee colony optimization [34], differential evolution [35], global equilibrium search [36], distance based search using the "backbone" and "big valley" properties [37].

III. An example of the heterogeneous job shop scheduling problem

Taking into account the job shop example of Table I, and maintaining the operations precedence and processing order through the machines, we have now four workers that we want to assign to the four machines, each of which has its own pace to accomplish the tasks on each machine. Table II shows custom times for each worker to form an heterogeneous job shop scheduling problem (Het-JSSP). In the example, we choose processing times randomly in the interval $[p, 2p]$, where p is the standardized time from the Table I. Moreover, some workers may be unable to operate some machines (for strategic or therapeutic reasons). In the example, worker W_3 is unable to operate machine M_3. This is represented by a processing time of ∞. The average increase of the processing times (without the incompatibilities) is 1.49, and the expected increase over the makespan should be near of 19 time units.

The average makespan over the 18 different possible worker assignments, maintaining the schedule that is optimal for the original instance, is 20.7 time units, the best assignment has a makespan of 19, and the worst has 23; note that almost all workers combinations produce makespans greaters than expected. The best assignment (assigning workers w_1, w_3, w_2, w_4 to the machines in this order) is shown in Figure 2. On the other hand, if we take into account the workers heterogeneity during the construction of the optimal schedule, we obtain a schedule shown in Figure 3 with a completely different worker assignment. This optimal schedule reduces the makespan to 16, about 19% less than the best assignment of the workers over the optimal job shop schedule.

IV. Problem formulation

This section presents a formulation of the heterogeneous job shop scheduling problem. We use the index i for machines, j for jobs, and w for workers. We also use the notation $[n] = \{1, \dots, n\}$, and the precedence relation set $R_j = \{(i, i') | o_{ij} \prec o_{i'j}\}$ for the precedence of all pairs of consecutive operations that belong to job j. The Het-JSSP can be formulated as

$$\begin{aligned}
\textbf{min.}\quad & C_{\max}, && && (1)\\
\textbf{s.t.}\quad & x_{mj} + p_{mj} \le C_{\max}, && \forall j \in [n], && (2)\\
& x_{ij} + p_{ij} \le x_{i'j}, && \forall (i,i') \in R_j, j \in [n], && (3)\\
& x_{ij} + p_{ij} \le x_{ij'} \\
& \qquad + M(1 - y_{ijj'}), && \forall i \in [m], j \ne j' \in [n], && (4)\\
& y_{ijj'} + y_{ij'j} = 1, && \forall i \in [m], j \ne j' \in [n], && (5)\\
& p_{ij} = \sum_{w \in [m]} p_{ijw} z_{iw}, && \forall i \in [m], j \in [n], && (6)
\end{aligned}$$

Table II. TIMES FOR AN INSTANCE OF THE HET-JSSP.

	Worker w_1				Worker w_2				Worker w_3				Worker w_4			
Job	M_1	M_2	M_3	M_4	M_1	M_2	M_3	M_4	M_1	M_2	M_3	M_4	M_1	M_2	M_3	M_4
J_1	4	3	2	4	3	2	2	3	5	2	∞	4	4	4	2	4
J_2	2	2	2	6	1	2	3	5	2	4	∞	5	2	2	2	4
J_3	2	3	1	2	2	2	2	4	2	3	∞	3	1	2	2	3
J_4	1	6	6	2	2	5	4	1	1	3	∞	2	1	3	5	2

(*) The operations were reordered to show the worker assignment, but their precedence is still the same than in Table I.

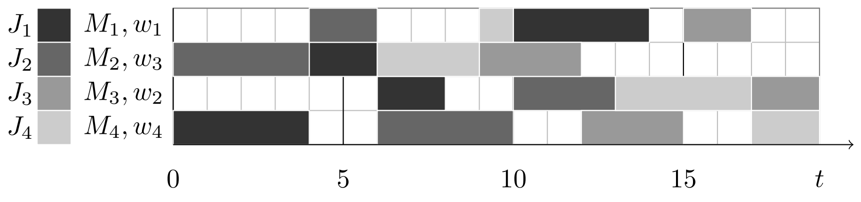

Figure 2. Best schedule obtained by reassigning workers and maintaining the optimal job shop schedule.

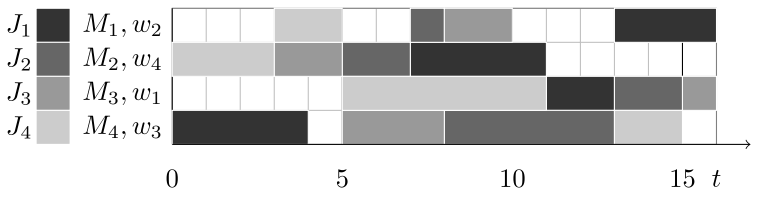

Figure 3. Optimal job shop schedule considering heterogeneous workers

$$\sum_{w \in [m]} z_{iw} = 1, \qquad \forall i \in [m], \quad (7)$$

$$\sum_{i \in [m]} z_{iw} = 1, \qquad \forall w \in [m], \quad (8)$$

$$x_{ij} \geq 0, \qquad \forall i \in [m], j \in [n], \quad (9)$$

$$y_{ijj'} \in \{0,1\}, \qquad \forall i \in [m], j \neq j' \in [n], \quad (10)$$

$$z_{iw} \in \{0,1\}, \qquad \forall i, w \in [m]. \quad (11)$$

This formulation uses the auxiliary variables x_{ij} and p_{ij} to represent the starting time and processing time of the operation of job j that must be executed on machine i. Constraints (2)–(5) solve the job shop subproblem for a fixed worker allocation. Constraint (2) defines $C_{\max}$ as the latest completion time, and constraints (3) and (4) set the starting time of all operations according to the precedences. The precedence between two operations of a job j that are executed consecutively on machines i and i' is fixed by constraint (3). The constant M should be sufficiently large (e.g. $\sum_{i\in[m]} \sum_{j\in[n]} \max_{w\in[m]} p_{ijw}$) in constraint (4), we introduce a binary decision variable $y_{ijj'}$ for each pair of operations of different jobs j and j' that must be processed on the same machine i, and whose precedence is not fixed. If j precedes j' on machine i, then $y_{ijj'}$ equals 1. Constraint (6) defines the processing time of an operation depending on the worker assigned to its machine, based on the processing time p_{ijw} of the operation o_{ij} when executed by worker w. We introduce a binary assignment variable z_{iw} which equals 1 if worker w is assigned to machine i. Constraints (7) and (8) define the assignment of workers to machines.

Algorithm 1 A multi-start local search for the Het-JSSP.

```
Output: Best solution found $S_{best}$
1: repeat
2:     Create a new solution $S_{new}$
3:     Apply a local search to $S_{new}$
4:     Update $S_{best}$ with $S_{new}$ if necessary
5: until stopping criterion is satisfied
6: return $S_{best}$
```

V. A MULTI-START LOCAL SEARCH METHOD FOR THE HET-JSSP

A multi-start local search heuristic for the heterogeneous job shop scheduling problem is shown in Algorithm 1. In a multi-start local search, various different initial solutions are generated by a constructive heuristic, and improved by a local search. Multi-start heuristics are efficient for problems where solutions are constructed easily, and neighbourhoods are difficult to be explored largely by local search methods. A more detailed description of multi-start methods is available in [38].

A. Construction method

Our construction method is described in Algorithm 2. It first generates a random worker assignment and sets the processing times for the incumbent JSSP. After that, it creates a solution for the incumbent job shop scheduling problem with the shifting bottleneck procedure (SBP). The SBP was proposed by Carlier et.al. [11] for the job shop scheduling problem. The SBP schedules iteratively the machine that is considered a bottleneck. To do this,

Algorithm 2 A construction heuristic for the Het-JSSP.

```
Output: Constructed solution
1: Assign workers randomly
2: repeat
3:     for all unscheduled machines do
4:         Solve the one-machine sequencing problem
5:     Schedule the bottleneck machine
6:     Reoptimize all the previously scheduled machines
7: until all machines are scheduled
8: return constructed solution
```

it solves the one-machine sequencing problem with release and due dates that minimizes the maximum lateness for each unscheduled machine, and schedules the machine with the worst maximum lateness (i.e. the bottleneck machine). After each machine is scheduled, the procedure applies a reoptimization over all the previously scheduled machines. The reoptimization consists in trying to reschedule each previously scheduled machine one by one, and updates the partial schedule if there is a better one. This process is repeated until all machines are scheduled.

B. Improvement method

A local search algorithm explores the neighbourhood of a solution looking for better ones. In each step, a local search chooses the best neighbour to replace the current solution. It stops when there are no better solutions in the neighbourhood. The local search applied in our heuristic uses the N5 neighbourhood proposed by Nowicki and Smutnicki [21]. N5 reduces the moves to the two first or the two last consecutive operations of a block on a critical path, excluding the two first and the two last operations of the critical path. For example, the neighbourhood of the schedule in Figure 1 contains three solutions, obtained by swapping the job operation pairs (J_1, J_4) on M_2, (J_4, J_1) on M_1, or (J_1, J_3) on M_1.

VI. Computational Experiments

We created instances for the Het-JSSP based on the job shop instances of Taillard [39], and on the instance *ft10* [40]. We modify the processing times p_{ij} of the original instances in two ways: First, we introduce a fixed percentage of incompatibilities to represent the case of workers that are unable to operate some machines. Second, we increase and multiply the processing times to simulate the different paces of workers. Based on experiences with existing SWDs, we generated instances with 0%, 10% and 20% of incompatibilities per worker, and we increased the standard time p randomly in the interval $[p, 2p]$ or $[p, 5p]$ for each of the m workers. We refer to instances in the tables below by the name of the base instance, followed by an i for processing times in $[p, 2p]$ or an I for processing times in $[p, 5p]$, and the percentage of incompatibilities. The instances as well as detailed computational results are available at http://www.inf.ufrgs.br/algopt/hetJS/

Our multi-start local search was implemented in C++, and compiled with GNU C++ 4.7.3 with optimization level 2 (-O2). We use the SBP implementation of Applegate and Cook [12]. The stopping criterion was one thousand iterations. We compare the solutions obtained by the multi-start local search to those obtained by solving the mathematical model of Section IV. The mathematical model has been solved with CPLEX 12.5 running with a single thread and a time limit of one hour. All computational tests were executed on a PC with an AMD Opteron processor running at 2.9 GHz and 64 GB of main memory.

Table III and IV present results for the CPLEX solver with the proposed model and for the multi-start local search for the instances based on the first fifty Taillard instances. Table III presents results for instances with processing times in $[p, 2p]$, and Table IV for instances with processing times in $[p, 5p]$. Each line of the tables reports 10 instances, grouped by size and percentage of incompatibility. All tables report the size of the instances $(n \times m)$, the average of the best known upper bounds $(\overline{ub})$ obtained by this or by previous preliminary experiments, the average of the relative deviation in percent $(\overline{\%R})$ of a solution with makespan $C_{\max}$ from the best known solution $\%R = 100\% \times (C_{\max} - ub)/ub$, the average of the percentage relative gap $(\overline{gap\%})$ between the lower and upper bounds found by CPLEX, and the average time $\bar{t}$ in seconds of the multi-start search. For the multi-start search we present the average deviation for the best solution found $(\overline{\%R})$ and for all independent solutions find on each iteration of construction and local search $(\overline{\%R_I})$.

CPLEX was able to create solutions but not to prove the optimality of any instance within the time limit of one hour. CPLEX finds the best known upper bound in 100 of the instances, 51 among the 60 instances of 15 jobs and 49 among the 120 instances of 20 jobs. The optimal value for the original *ft10* is found in less than 10 minutes, and its optimality is proven in less than 15 minutes, but CPLEX cannot prove optimality even for Het-JSSP instances of size 10×10 based on *ft10*, and the average CPLEX gap for these instances is 21% after one hour time limit. This is expected, since the number of possible solutions is $m!$ times larger due to the worker assignment. CPLEX only finds better solutions for the small instances, and the gap between the CPLEX lower and upper bound increases with the size of the instance from 30% to 61%.

The multi-start search reaches the best known values in the other 200 instances, all the 120 instances of 30 jobs using less than 12 minutes, 71 among the 120 instances of 20 jobs using less than 5 minutes, and 9 among the 90 instances of 15 jobs using less than 1.1 minutes. The average deviation of the best found solution $\overline{\%R}$ decreases from 2.3% to 0% for the instances with processing times $[p, 2p]$, and from 6.4% to 0% for the instances with processing times $[p, 5p]$. The difficulty of the problem grows with the size of the instance, and that the multi-start approach shows scalability for bigger instances. The average deviation of all the created solutions $\overline{\%R_I}$ decreases from 12.4% to 7.1% for the instances with processing times $[p, 2p]$, and from 21.4% to 9.7% for the instances with processing times $[p, 5p]$. The local search improves less than 5% of the constructed solutions, and the relative improvement is less than 0.7%, but also the average time that it consumes is less than 4% of the total running time.

Table III. Average results for the Taillard instances with processing times between t and $2t$.

			CPLEX		Multi-start		
Instances	n× m	$\overline{ub}$	$\overline{\%R}$	$\overline{gap\%}$	$\bar{t}$	$\overline{\%R}$	$\overline{\%R_I}$
tai01i00 - tai10i00	15×15	1783.2	0.26	30.22	61.9	2.22	12.34
tai01i10 - tai10i10	15×15	1794.8	0.07	30.25	61.3	1.70	11.61
tai01i20 - tai10i20	15×15	1797.8	0.76	30.58	62.0	1.80	11.33
tai11i00 - tai20i00	20×15	2095.1	2.77	42.37	112.8	0.18	8.82
tai11i10 - tai20i10	20×15	2088.9	3.79	42.18	112.5	0.19	9.25
tai11i20 - tai20i20	20×15	2096.2	3.80	41.93	114.4	0.00	8.96
tai21i00 - tai30i00	20×20	2494.1	1.65	38.15	256.9	0.64	8.05
tai21i10 - tai30i10	20×20	2496.6	2.20	38.25	253.8	0.40	7.97
tai21i20 - tai30i20	20×20	2494.5	2.18	37.93	254.6	0.12	8.07
tai31i00 - tai40i00	30×15	2769.9	16.80	60.35	298.5	0.00	7.78
tai31i10 - tai40i10	30×15	2771.7	16.16	59.92	299.6	0.00	7.65
tai31i20 - tai40i20	30×15	2786.0	17.14	60.29	298.6	0.00	7.14
tai41i00 - tai50i00	30×20	3119.3	26.81	59.39	649.1	0.00	7.84
tai41i10 - tai50i10	30×20	3128.7	21.42	57.69	639.2	0.00	7.46
tai41i20 - tai50i20	30×20	3125.6	20.12	56.94	633.5	0.00	7.56
			9.06	45.76		0.48	8.79

Table IV. Average results for the Taillard instances with processing times between t and $5t$.

			CPLEX		Multi-start		
Instances	n× m	$\overline{ub}$	$\overline{\%R}$	$\overline{gap\%}$	$\bar{t}$	$\overline{\%R}$	$\overline{\%R_I}$
tai01I00 - tai10I00	15×15	3407.9	0.19	35.83	62.7	5.86	20.65
tai01I10 - tai10I10	15×15	3396.6	0.41	35.22	62.4	6.33	21.35
tai01I20 - tai10I20	15×15	3448.8	0.06	34.91	62.2	4.75	19.47
tai11I00 - tai20I00	20×15	4028.4	1.27	46.99	115.0	1.72	15.49
tai11I10 - tai20I10	20×15	4029.7	1.56	46.61	117.0	1.66	15.45
tai11I20 - tai20I20	20×15	4065.5	2.48	46.59	116.7	1.03	14.48
tai21I00 - tai30I00	20×20	4766.4	1.59	42.53	261.8	3.31	15.13
tai21I10 - tai30I10	20×20	4802.1	0.70	41.74	270.2	2.95	14.37
tai21I20 - tai30I20	20×20	4822.4	0.48	41.30	269.7	2.74	13.86
tai31I00 - tai40I00	30×15	5445.2	15.38	63.47	302.8	0.00	10.83
tai31I10 - tai40I10	30×15	5428.1	16.42	63.40	303.9	0.00	10.94
tai31I20 - tai40I20	30×15	5407.4	15.06	62.32	304.5	0.00	11.31
tai41I00 - tai50I00	30×20	6186.2	23.13	62.09	711.6	0.00	10.06
tai41I10 - tai50I10	30×20	6208.6	19.69	60.83	682.1	0.00	9.70
tai41I20 - tai50I20	30×20	6178.2	21.66	60.91	711.9	0.00	10.39
			8.00	49.65		2.02	14.23

This indicates that even the constructive heuristic alone creates good quality solutions, but it needs a local search to guarantee that the final solution is a local minimum. This construction method is thorough, therefore we may use a swifter construction, and invest better this time in other local search and diversification methods.

VII. Concluding remarks

We have proposed an extension to the job shop scheduling problem that considers heterogeneous workers with their assignment and incompatibilities. We gave a mathematical model of the problem, and proposed a multi-start local search to solve it.

The Het-JSSP is considerably harder than a traditional job shop problem. The Het-JSSP mathematical model cannot be solved by standard software even for small instances of size 10×10. On the other hand, computational tests show that the problem can be solved satisfactorily by a multi-start local search.

Our proposal may be beneficial to the operations management area, when all (or some) workers perform their tasks with different execution times, and when both the optimal schedule and the optimal assignment of workers vary depending on each other.

The inclusion of workers heterogeneity within a job shop system is an important step for those companies that desire the integration of a percentage of disabled workers as part of their policies of Corporate Social Responsibility without loosing productivity.

Acknowledgment

We are grateful for the support by CAPES-Brazil and MEC-Spain (project CAPES-DGU258-12/PHB-0012-PC), by FAPERGS (project 12/2046-6), by Petrobras, by CNPq (project 478847/2013-0), and by the National Center for Supercomputing at UFRGS.

References

[1] C. Miralles, J. Marin-Garcia, G. Ferrus, and A. Costa, "OR/MS tools for integrating people with disabilities into employment. a study on valencia's sheltered work centres for disabled," *Int. Trans. Oper. Res.*, vol. 17, pp. 457–473, 2010.

[2] E. Erel, I. Sabuncuoglu, and H. Sekerci, "Stochastic assembly line balancing using beam search," *Int. J. Prod. Res.*, vol. 43, no. 7, pp. 1411–1426, 2005.

[3] A. Corominas, R. Pastor, and J. Plans, "Balancing assembly line with skilled and unskilled workers," *Omega*, vol. 36, no. 6, pp. 1126–1132, 2008.

[4] C. Miralles, J. P. Garcia-Sabater, C. Andrés, and M. Cardos, "Advantages of assembly lines in Sheltered Work Centres for Disabled. A case study," *Int. J. Prod. Res.*, vol. 110, no. 2, pp. 187–197, 2007.

[5] C. Blum and C. Miralles, "On solving the assembly line worker assignment and balancing problem via beam search," *Comput. Oper. Res.*, vol. 38, no. 1, pp. 328–339, 2011.

[6] M. C. O. Moreira, M. Ritt, A. M. Costa, and A. A. Chaves, "Simple heuristics for the assembly line worker assignment and balancing problem," *J. Heuristics*, vol. 18, no. 3, pp. 505–524, 2012.

[7] A. Corominas, L. Ferrer, and R. Pastor, "Assembly line balancing: general resource-constrained case," *Int. J. Prod. Res.*, vol. 49, no. 12, pp. 3527–3542, 2012.

[8] L. Borba and M. Ritt, "A heuristic and a branch-and-bound algorithm for the assembly line worker assignment and balancing problem," *Comput. Oper. Res.*, vol. 45, pp. 87–96, 2014.

[9] A. J. Benavides, M. Ritt, and C. Miralles, "Flow shop scheduling with heterogeneous workers," *Eur. J. Oper. Res.*, vol. 237, no. 2, pp. 713–720, 2014.

[10] J. Lenstra and A. Rinnooy Kan, "Computational complexity of discrete optimization problems," *Annals of Discrete Mathematics*, vol. 4, pp. 121–140, 1979.

[11] J. Carlier and E. Pinson, "An algorithm for solving the job-shop problem," *Manag. Sci.*, vol. 35, no. 2, pp. 164–176, 1989.

[12] D. Applegate and W. Cook, "A computational study of the job-shop scheduling problem," *ORSA J. Comput.*, vol. 3, no. 2, pp. 149–156, 1991.

[13] P. Brucker, B. Jurisch, and B. Sievers, "A branch and bound algorithm for the job-shop scheduling problem," *Discrete App. Math.*, vol. 49, no. 1, pp. 107–127, 1994.

[14] J. Carlier and E. Pinson, "A practical use of Jackson's preemptive schedule for solving the job-shop problem," *Ann. Oper. Res.*, vol. 26, pp. 269–287, 1990.

[15] J. Adams, E. Balas, and D. Zawack, "The shifting bottleneck procedure for job shop scheduling," *Manag. Sci.*, vol. 34, no. 3, pp. 391–401, 1988.

[16] E. Balas, J. Lenstra, and A. Vazacopoulos, "The one-machine problem with delayed precedence constraints and its use in job shop scheduling," *Manag. Sci.*, vol. 41, no. 1, pp. 94–109, 1995.

[17] E. Balas and A. Vazacopoulos, "Guided local search with shifting bottleneck for job shop scheduling," *Manag. Sci.*, vol. 44, no. 2, pp. 262–275, 1998.

[18] P. Van Laarhoven, E. Aarts, and J. Lenstra, "Job shop scheduling by simulated annealing," *Operations research*, vol. 40, no. 1, pp. 113–125, 1992.

[19] T. Yamada and R. Nakano, "Job-shop scheduling by simulated annealing combined with deterministic local search," *Metaheuristics: Theory and applications*, pp. 237–248, 1996.

[20] M. Kolonko, "Some new results on simulated annealing applied to the job shop scheduling problem," *Eur. J. Oper. Res.*, vol. 113, no. 1, pp. 123–136, 1999.

[21] E. Nowicki and C. Smutnicki, "A fast taboo search algorithm for the job shop problem," *Manag. Sci.*, vol. 42, no. 6, pp. 797–813, 1996.

[22] ——, "An advanced tabu search algorithm for the job shop problem," *Journal of Scheduling*, vol. 8, no. 2, pp. 145–159, 2005.

[23] ——, "Some new ideas in TS for job shop scheduling," in *Metaheuristic Optimization via Memory and Evolution*, R. Sharda, S. Voß, C. Rego, and B. Alidaee, Eds. Springer, 2005, pp. 165–190.

[24] J. Watson, A. Howe, and L. Darrell Whitley, "Deconstructing Nowicki and Smutnicki's i-TSAB tabu search algorithm for the job-shop scheduling problem," *Comput. Oper. Res.*, vol. 33, no. 9, pp. 2623–2644, 2006.

[25] T. Yamada and R. Nakano, "Scheduling by genetic local search with multi-step crossover," *Parallel Problem Solving from Nature IV*, pp. 960–969, 1996.

[26] J. Gonçalves, J. de Magalhães Mendes, and M. Resende, "A hybrid genetic algorithm for the job shop scheduling problem," *Eur. J. Oper. Res.*, vol. 167, no. 1, pp. 77–95, 2005.

[27] J. F. Gon calves, M. G. C. Resende, and J. J. M. Mendes, "A biased random-key genetic algorithm with forward-backward improvement for the resource constrained project scheduling problem," *J. Heuristics*, 2010.

[28] J. F. Gonçalves and M. G. C. Resende, "An extended akers graphical method with a biased random-key genetic algorithm for job-shop scheduling," *Int. Trans. Oper. Res.*, vol. 21, no. 2, pp. 215–246, 2014.

[29] R. Aiex, S. Binato, and M. Resende, "Parallel {GRASP} with path-relinking for job shop scheduling," *Parallel Computing*, vol. 29, no. 4, pp. 393 – 430, 2003, parallel computing in numerical optimization.

[30] S. Fernandes and H. Lourenço, "A simple optimised search heuristic for the job shop scheduling problem," in *Recent Advances in Evolutionary Computation for Combinatorial Optimization*, ser. Studies in Computational Intelligence, C. Cotta and J. van Hemert, Eds. Springer Berlin Heidelberg, 2008, vol. 153, pp. 203–218.

[31] T.-L. Lin, S.-J. Horng, T.-W. Kao, Y.-H. Chen, R.-S. Run, R.-J. Chen, J.-L. Lai, I. Kuo *et al.*, "An efficient job-shop scheduling algorithm based on particle swarm optimization," *Exp. Syst. Appl.*, vol. 37, no. 3, pp. 2629–2636, 2010.

[32] R. Zhang and C. Wu, "A neighbourhood property for the job shop scheduling problem with application to hybrid particle swarm optimization," *IMA Journal of Management Mathematics*, vol. 24, no. 1, pp. 111–134, 2013.

[33] K. Huang and C. Liao, "Ant colony optimization combined with taboo search for the job shop scheduling problem," *Comput. Oper. Res.*, vol. 35, no. 4, pp. 1030–1046, 2008.

[34] A. Banharnsakun, B. Sirinaovakul, and T. Achalakul, "Job shop scheduling with the best-so-far abc," *Engineering Applications of Artificial Intelligence*, vol. 25, no. 3, pp. 583–593, 2012.

[35] A. Ponsich and C. A. Coello Coello, "A hybrid differential evolution—tabu search algorithm for the solution of job-shop scheduling problems," *Applied Soft Computing*, vol. 13, no. 1, pp. 462–474, 2013.

[36] P. M. Pardalos and O. V. Shylo, "An algorithm for the job shop scheduling problem based on global equilibrium search techniques," *Computational Management Science*, vol. 3, no. 4, pp. 331–348, 2006.

[37] P. M. Pardalos, O. V. Shylo, and A. Vazacopoulos, "Solving job shop scheduling problems utilizing the properties of backbone and "big valley"," *Computational Optimization and Applications*, vol. 47, no. 1, pp. 61–76, 2010.

[38] R. Martí, M. G. Resende, and C. C. Ribeiro, "Multi-start methods for combinatorial optimization," *Eur. J. Oper. Res.*, vol. 226, no. 1, pp. 1 – 8, 2013.

[39] E. Taillard, "Benchmarks for basic scheduling problems," *Eur. J. Oper. Res.*, vol. 64, no. 2, pp. 278–285, 1993.

[40] H. Fischer and G. Thompson, *Probabilistic Learning combinations of local job-shop scheduling rules.* -: Englewood Cliffs (Prentice Hall), 1963, pp. 225–251.