A minimalist yet efficient tabu search algorithm for balancing assembly lines with disabled workers.

Mayron César de O. Moreira, Alysson M. Costa

Instituto de Ciências Matemáticas e de Computação - Universidade de São Paulo
{mayron,alysson}@icmc.usp.br

Abstract

We face the problem of programming assembly lines when each task has a different completion time depending on the worker assigned to its execution. This problem appears in the context of sheltered work centers for the disabled, where the workers have disabilities that might impede or difficult the carrying out of some tasks. We develop a tabu search algorithm for solving this problem. In the designing of this algorithm, we have proposed and followed several guidelines intended to make the method as simple, flexible, accurate and fast as possible. We compare the proposed approach with more sophisticated methods available for the same problem in the literature. Although we have not focused on accuracy and speed, as it is usually the case in the Operations Research literature, the results show that the algorithm outperforms other methods even in these criteria.

Keywords: Metaheuristics, assembly lines, disabled workers.

1 Introduction

Recent times have seen an increase in societal awareness on issues related to the well-being of people with disabilities. On May 2008, came into force the *Convention on the Rights of Persons with Disabilities* (CRPD). This convention, which intends "to promote, protect and ensure the full and equal enjoyment of all human rights and fundamental freedoms by all persons with disabilities, and to promote respect for their inherent dignity"(United Nations 2006) has been signed by more than 140 countries. In particular, the participation of Brazil has been ratified by the National Congress in August 2008.

The CRPD has a chapter dedicated to work and employment, in which the "States Parties recognize the right of persons with disabilities to work, on an equal basis with others". In practice, however, the situation of disabled workers is poor, specially in developing countries. As an example, take the results of a research carried out by the Employement Secretary of the city of São Paulo which showed that 90% of the disabled population on working age was unemployed (SERPRO 2004). The reasons for these alarming statistics are numerous and certainly include misinformation, prejudice and lack of consideration of the special needs of these workers. Concerning this last point, it is known that many disabled workers need a longer adaptation period, in which they get used to the work environment and learn the necessary skills. Indeed, Gates (2000) mentions that the integration of disabled workers should be viewed as a social process which success depends not only on the individual disabilities symptoms and characteristics but also on the responsiveness of the work environment. In this respect, the CRPD mentions that each signatory should "promote the acquisition by persons with disabilities of work experience in the open labor market". To cope with this question, some countries propose the creation of Sheltered Work centers for the Disabled (referred to as SWD henceforth). These centers, which serve as a first work-environment for disabled workers, should be characterized by an environment where these workers can gradually get adapted to a working routine and develop their own personal skills, before being fully integrated into the conventional labor market.

Although the main goal of a SWD concerns the adaptation of disabled workers, these centers are also productive unities competing in the market. Therefore, they must be efficient and organize their activities in an adequate manner. The main decision, in many cases, is how to effectively operate the assembly line. This problem classically consists in assigning tasks to workstations, which are disposed in a pre-defined order (usually, one after another in a serial fashion). The tasks commonly have some set of precedence constraints, usually represented by a precedence graph $G = (V, A)$ in which V is the set of tasks and an arc (i, j) is in A if task i must be executed before j and there is no task k that must be executed after i and before j (i.e., i is a direct predecessor of j). The goal is to maximize the throughput of the line. In other words, one must balance the workload between the workstations while respecting the precedence constraints. In SWD, besides assigning tasks to workstations, one must also consider the workers assignment decision, since each worker is now a very unique individual which might execute some tasks as rapidly as a *conventional* worker but might, as well, be very slow when executing some other tasks or even be unable to execute some of the tasks.

Due to this double-assignment characteristic, this problem has been called the Assembly Line Worker Assignment and Balancing Problem (ALWABP). Our goal in this paper is to present a method

to solve the ALWABP that is as simple, flexible, accurate and fast as possible. These characteristics have not been chosen randomly but follow the guidelines presented by Cordeau et al. (2002). Clearly, some of these attributes might be contradictory (accurate methods usually rely on some kind of complex procedures, e.g.), and a trade-off must be found somewhere. This has led us to the choice of a meta-heuristic. More specifically, we have developed a tabu-search algorithm (Glover 1989a, Glover 1989b, Glover & Laguna 1997) which makes extensive use of the idea of varying penalties introduced by Gendreau et al. (1994).

The remainder of this article is organized as follows. In the next Section, we present a brief literature review on the ALWABP. Section 3 is concerned with presenting a mathematical model for the problem. Our tabu search algorithm is described in detail in Section 4. Computational results are presented and discussed in Section 5. This papers ends with some conclusions in Section 6.

2 Literature review

The operation and planning of assembly lines have been extensively studied in the literature. The most classical problem is probably the Simple Assembly Line Balancing Problem (SALBP) (Baybars 1986, Rekiek et al. 2002, Scholl & Becker 2006), which consists in assigning n tasks to workstations with the most usual goals of minimizing the number of needed workstations, m, given a maximum allowed cycle time $\overline{C}$ (SALBP-1) or minimizing cycle time, C, given a fixed number $\overline{m}$ of workstations (SALBP-2). The main characteristics of the SALBP have been listed by Scholl & Becker (2006) as follows:

a) mass-production of one homogeneous product;

b) given production process;

c) paced line with fixed cycle time C;

d) deterministic (and integral) operation times t_j;

e) no assignment restrictions besides the precedence constraints;

f) serial line layout with m stations;

g) all stations are equally equipped with respect to machines and workers;

h) goal of maximizing the line efficiency $E = t_{\text{sum}}/(m \cdot C)$ with total task times $t_{\text{sum}} = \sum_{i=1}^{n} t_i$.

When the problem has the objective function described in h), it is known as SALBP-E, which is a generalized version of the problem encompassing SALBP-1 and SALBP-2. Another important variant is the feasibility version of the problem, SALBP-F, which consists in deciding wether a given solution with $\overline{m}$ workstations and cycle time $\overline{C}$ is feasible. The decision version SALBP-F is NP-complete while the optimization versions SALBP-1, SALBP-2 and SALBP-E are known to be NP-hard.

The main novelty introduced by the ALWABP concerns aspects d) and g). Indeed, due to the own characteristics of the disabled workers of a SWD, the execution times of a task t_j depends on the worker executing it. Therefore, the positions of the workers in the line become a decision to be taken by the line manager.

The ALWABP has been introduced by Miralles et al. (2007, 2008). The authors define the problem and present a case study based on a Spanish SWD. Variants of the ALWABP have also been studied in

the literature: Miralles et al. (2005) analyze the design of U-shaped assembly lines in the context found at SWD while Costa & Miralles (2009) propose models and algorithms to plan job rotation schedules in SWD.

In what concerns solution methods, a branch-and-bound algorithm has been proposed by Miralles et al. (2008). Due to the problem's complexity, the approach can only be applied to small-sized instances. Heuristic methodologies have been investigated by Chaves et al. (2007, 2008). The authors propose a sophisticated hybrid meta-heuristic named Clustering Search (Oliveira & Lorena 2004), which has been applied with success to hard combinatorial problems such as the Traveling Salesman Problem with Profits (Chaves & Lorena 2008), the Capacitated Centred Clustering Problem (Chaves & Lorena 2009) and the Probabilistic Maximal Covering Location-Allocation Problem (Corrêa et al. 2009).

Clustering search works by identifying clusters of good solutions found by another meta-heuristic and proposing intensification procedures at promising solutions belonging to these clusters. As described in (Chaves et al. 2007), the approach relies on four conceptually independent components with different attributions, being them: a search metaheuristic; an iterative clustering; an analyzer module; and a local searcher. The reader is referred to that paper and to the doctoral thesis of Chaves (2009) for a better description of the interrelations and interactions between these components.

Although one can not deny the efficacy of complex approaches such as the Clustering Search algorithm, it is also arguable that the difficulty of understanding and implementing complicated methods such the CS might slow down or even impede its utilization in practice. A nice discussion on this subject has been done by Cordeau et al. (2002) in the context of algorithms for the Vehicle Routing Problem (some additional discussion on this subject is presented in Section 4). For this reason, in this article, our goal was to develop an efficient yet simple method. The remainder of this paper will discuss the proposed method and analyze (by means of computational tests) its efficiency. First, in the following section, we discuss a mathematical formulation for the ALWABP.

3 Mathematical model

For the sake of completeness, we present a mathematical model for the ALWABP in this section. This formulation has been proposed by Miralles et al. (2007) and used in (Chaves 2009) to obtain lower bounds for the problem. The formulation uses the following notation:

i,j	indexes for tasks,
w	index for workers,
s	index for workstations,
N	set of tasks,
W	set of workers,
S	set of workstations,
C	cycle time,
p_{wi}	processing time for task i when executed by worker w,
D_j	set of tasks that immediately precede task j in the precedence graph ,
x_{swi}	binary variable. Equal to 1 only if task i is assigned to worker w at workstation s,
y_{sw}	binary variable. Equal to 1 only if worker w is assigned to workstation s,

M constant such that $M \geq \sum_{w \in W} \sum_{i \in N} p_{hi}$.
I_w set of tasks that worker w is not able to execute.

The model can thus be written as:

$$\text{Min } C \tag{1}$$

subject to

$$\sum_{w \in W} \sum_{s \in S} x_{swi} = 1, \qquad \forall i \in N, \tag{2}$$

$$\sum_{s \in S} y_{sw} = 1 \qquad \forall w \in W, \tag{3}$$

$$\sum_{w \in W} y_{sw} = 1 \qquad \forall s \in S, \tag{4}$$

$$\sum_{w \in W} \sum_{s \in S} s x_{swi} \leq \sum_{w \in W} \sum_{s \in S} s x_{swj} \qquad \forall i, j \in N | i \in D_j, \tag{5}$$

$$\sum_{i \in N} p_{wi} x_{swi} \leq C \qquad \forall w \in W, \forall s \in S, \tag{6}$$

$$\sum_{i \in N} x_{swi} \leq M y_{sw} \qquad \forall w \in W, \forall s \in S, \tag{7}$$

$$x_{swi} = 0, \qquad \forall w \in W, \forall s \in S, \forall i \in I_w, \tag{8}$$

$$y_{sw} \in \{0, 1\} \qquad \forall s \in S, \forall w \in W, \tag{9}$$

$$x_{swi} \in \{0, 1\} \qquad \forall s \in S, \forall w \in W, \forall i \in N. \tag{10}$$

Model (1)–(10) focus on minimizing the cycle time for a given number of workstations, i.e., the number of stations is a fixed parameter and the goal is to minimize the cycle time C (ALWABP-2). This is, indeed, the main problem in SWD where one wants to employ as many workers as possible. Constraints (2) guarantee that each task is executed, and that it is done by a single worker, at a single workstation. Constraints (3) and (4) establish a biunivocal relation between workers and workstations at a feasible solution, i.e., every worker is assigned to a single workstation and *vice versa*. Constraints (5) define the precedence relations, while constraints (6) and (7) establish that the cycle time is the sum of the execution times of the tasks at the most charged workstation. Constraints (8) define that a task is not assigned to a worker that is not able to execute it. These constraints could be defined directly in constraints (10), by eliminating the associated variables, but are expressed as a particular set of constraints for convenience, as they will be used to explain the proposed tabu search algorithm in Section 4.

Some improvements can be made on the original model. Indeed, the formulation remains valid if the parameter M has its value reduced to n, the number of tasks. Alternatively, disaggregated constraints on the form:

$$x_{swi} \leq y_{sw}, \qquad \forall i \in N, \forall w \in W, \forall s \in S, \tag{11}$$

could replace aggregated constraints (7). In our tests, we found that it was more efficient to keep the original constraints with the new value of M. Additionally, constraints (6) can be strengthened to:

$$\sum_{w \in W} \sum_{i \in N} p_{wi} x_{swi} \leq C \qquad \forall s \in S, \tag{12}$$

and

$$\sum_{s \in S} \sum_{i \in N} p_{wi} x_{swi} \leq C \qquad \forall w \in W, \tag{13}$$

which proved to be beneficial. Indeed, note that only one set of these constraints is needed.

4 Tabu search algorithm

As discussed earlier, our goal in this paper is to propose an algorithm that is as simple, flexible, accurate and fast as possible. This goes along with the proposal of Cordeau et al. (2002), who suggest the addition of the simplicity and flexibility criteria to the long established custom of evaluating solution methods solely by their accuracy and speed. The rationale and motivation comes from the fact that, in most cases, only (reasonably) simple and flexible algorithms are implemented in practice.

Simplicity, flexibility, accuracy and speed are somewhat conflicting objectives. Accurate algorithms tend to be slow and complex; flexible algorithms cannot exploit many problem-related specificities and tend to be less-accurate than algorithms tailored to the specific needs of a class of problems (or even to an instance of the problem), etc. In order to obtain a trade-off between these goals we have developed and tried to pursue the following guidelines:

- (Simplicity) Algorithms should be easy to explain and understand; Algorithms should not rely on the user for the setting of an extensive number of parameters. As a general rule, parameters should be self-adjustable and/or the algorithm should be robust to them.
- (Flexibility) It should be easy to adapt the algorithm so that it copes with (minor) problem particularities to which it was not originally designed.
- (Accuracy) The algorithm should be as accurate as possible, as long as it does not violate the preceding guidelines.
- (Speed) As a rule of thumb, the algorithm should be as fast as possible. However, one should take into consideration not only the *accuracy* guideline mentioned above but also be coherent to the context in which the problem is inserted (real-time, operational, planning, etc.).

The proposed guidelines are quite general and their use is not restricted to any algorithmic approach. We exemplify these concepts with the use of a tabu search algorithm (Glover & Laguna 1997), which main idea is originally simple and can be condensed (in a intentional oversimplified way) as follows: The tabu search algorithm method tries to find a good solution by visiting better neighbors (see paragraph below) of an incumbent solution. The method escapes from local optima by allowing the incumbent solution to deteriorate, if no better neighbor exists. In order to avoid cycling, the algorithm movements cannot be reversed (are tabu) for a certain number of iterations.

In a Tabu search algorithm (as in many other meta-heuristics), one needs to define a neighborhood. We define as a *neighbor* solution, any solution that can be achieved from the current solution by means of one of the simple three movements:

move-task : move one task from one workstation to another.
swap-tasks : swap two tasks between two workstations.
swap-workers : swap two workers between two workstations.

The core of our algorithm has already been described above, with the definition of the basic tabu procedure and of a neighborhood. Some implementation details include the order in which the movements are effected (in our case, we follow the order used for presenting the movements) and an aspiration criterium used to allow some tabu movements (a tabu solution can be chosen as new incumbent if it is feasible and has an objective function that is better than the best known solution found so far).

The algorithm presented so far is simple (it relies on a basic - and classical - tabu structure and on simple and intuitive neighborhoods) and potentially fast. In order to cope with the flexibility and accuracy guidelines we make use of penalties in the objective function. The use of penalties has been a common practice in Operations Research and is core to traditional methods such as Lagrangian Relaxation (Fisher 2004). The specific use of penalties as described in this paper has been proposed by Gendreau et al. (1994) in the context of the Vehicle Routing Problem. The idea maintains the simplicity of the algorithm and consists in allowing infeasible solutions during the search, as long as the infactibilities are penalized in the objective function. There are penalty terms for each subset of important constraints and their values are self-adjustable in the following manner: once a solution respecting a subset of constraints is found, the penalty term associated to these constraints is reduced (divided by 2, e.g.). Analogously, if the current solution violates a subset of constraints, the associated penalty term is increased (doubled, e.g.). The rationale is to force the algorithm into feasible regions of the search space once it has found infeasible solutions for an increased number of iterations and allow it to visit infeasible solutions (as a sort of diversification mechanism) once it has visited feasible solutions for a number of iterations. Note that this is done *automatically* with the dynamic change of the penalties, without the need of defining feasibilization procedures (which can be rather clumsy and difficult to implement).

In the particular case of the algorithm developed here for the ALWABP, we set penalty terms for the precedence constraints (5) and to infeasible (tasks $\times$ worker) assignments, as defined in constraints (8). The rationale behind this choice is that these constraints are hard to respect but can be easily dealt with the proposed penalization strategy. The infactibilities are measured in the following manner:

- Precedence constraints infactibilities associated with task i, Ip_i: number of tasks $j \in D_i$ that are executed before i. (In our implementation, the terms in Ip_i have be weighted by the *distance* of the tasks, in terms of number of workstations, between i and $j \in D_i$.).
- (tasks $\times$ worker) infactibilities associated with the worker in machine s: number of tasks designated to s that cannot be solved by the worker designated to s.

An additional penalty, associated with the number of iterations the algorithm spent without using the swap-workers movement, was also set. The idea of this additional penalty was to promote diversification in the solution, since the swap-workers movement was, otherwise, rather rarely used. This penalty was set in the simple following manner: movements not associated with swap-workers were penalized by a factor γ. After each iteration without a swap-worker movement, γ was increased and, eventually, the algorithm would chose to swap workers (γ was then reduced to its initial value).

A pseudo-code is presented in Algorithm 1. In this algorithm, for a solution s, $f(s)$ represents the objective function value (considering the penalties) and $N(s)$ represents the set of Neighbors of s.

Algorithm 1 Tabu search algorithm for the ALWABP

1: $s =$ initial (possibly infeasible) solution obtained randomly;
2: $\overline{s} = s$;
3: $p_t = 100, p_w = 100$;
4: **while** stopping criterion not reached **do**
5: **if** s is feasible **then**
6: $p_t = 0.8p_t; p_w = 0.8p_w$; update γ;
7: **else**
8: $p_t = 1.2p_t; p_w = 1.2p_w$; update γ;
9: **end if**
10: $f(b) = \infty$;
11: **for** $j \in N(s)$ **do**
12: **if** j is feasible and $f(j) < f(\overline{s})$ **then**
13: apply aspiration criterium: $\overline{s} = j$; $s = j$; **go to** 4;
14: **end if**
15: **if** $f(j) < f(b)$ and j is not tabu **then**
16: $b = j$;
17: **end if**
18: **end for**
19: $s = b$;
20: Define movement tabu for $\frac{|N|}{4}$ iterations (move-task and swap-task movements) or $\frac{|N|}{2}$ iterations (swap-workers movement).
21: **end while**

Some additional comments are in order. In line 3 we define initial values for the penalties associated with precedences, p_t, and with (task $\times$ worker) infeasibility, p_w. We set the initial values at 100 in the example, but the algorithm proved robust to this parameter. Moreover, the dynamic change of the parameters proved to be an efficient way of switching between feasible and infeasible regions and, therefore, thoroughly exploring the search space. The stopping criterium was a pre-defined number of iterations (10.000, in our tests).

As intended, the algorithm has been made as simple as possible. It is important to notice that it is also flexible, in the sense that new problem characteristics could be incorporated with the use of the same penalty scheme developed for the precedence and (worker $\times$ task) infeasibility constraints. In order to evaluate the speed and accuracy of the method, we have compared it, by means of a series of computational tests, with the elaborate clustering search algorithm of Chaves et al. (2007). These results are presented in the following section.

5 Computational results

In order to test the proposed algorithm in terms of accuracy and speed, we have used 4 groups of instances available in the literature. These instances have been used in (Chaves et al. 2007, Chaves et al. 2008, Chaves 2009) and, therefore, allow the comparison of our method to the Clustering Search algorithm of Chaves et al. (2007, 2009). The instances are grouped in four families: Roszieg, Heskia, Wee-mag and Tonge. The characteristics for each group of instances (number of tasks, number of workers and the order strength of the precedence network) are listed in Table 1.

We have run the tabu search algorithm 20 times for each instance with a random start solution at each run, following the strategy used by Chaves (2009). We have also run the tabu search algorithm

Family	$\|N\|$	$\|W\|$	Order Strength
Roszieg	25	4 (groups 1-4) or 6 (groups 5-8)	71.67
Heskia	28	4 (groups 1-4) or 6 (groups 5-8)	22.49
Wee-mag	70	10 (groups 1-4) or 17 (groups 5-8)	22.67
Tonge	75	11 (groups 1-4) or 19 (groups 5-8)	59.42

Tabela 1: Instance characteristics

once starting with a (possibly infeasible) greedy solution (Moreira & Costa 2009). Tables 2-5 present the results for each family of instances. In the tables, the results are averaged in each line for each group of 10 instances. The tables list for each method (CS - Clustering Search, TS - Tabu Search with random initial solution, TSG - Tabu Search with greedy initial solution) the best solution found, *best*, the total computational time, $t(s)$, and the time in which the best was found, $t_b(s)$. For methods CS and TS, the tables also list the average solution found out of the 20 runs (*avrg*).

Two notes are in order. First, we mention that the *branch-and-cut* solver of the commercial package IBM ILOG CPLEX 11.2 was only able to efficiently solve instances of the smaller Roszieg and Heskia families and performed very poorly (sometimes failing in finding feasible solutions even after several hours of computational time) for the larger instances. The second note concerns the Clustering Search algorithm. Chaves (2009) developed several versions of the CS, using different meta-heuristics. The best results were obtained when the CS used a Simulated Annealing algorithm as a solution generator and, therefore, we use these results for comparison. The reader interested in a more detailed description of the results obtained by CPLEX or in the results of the CS with other meta-heuristics is referred to (Chaves 2009).

The results in the tables show the efficacy of the proposed methodology. Indeed, although we did not give priority to accuracy and speed during the design process, the results show that our method often outperforms the sophisticated clustering search algorithm of Chaves et al. (2007) even in these criteria. Indeed, for families Roszieg and Heskia, the quality of the solutions obtained is virtually the same for all methods although the proposed tabu search finds the best solution in only a small fraction of the time needed by the clustering search (the CS results were obtained with a Pentium 4 2.6 GHz processor with 1 GB RAM while our results were obtained with a Intel Core 2 Duo processor T5450, 1.66 GHz, 3 GB RAM).

For the instances in the Wee-mag family, the CS could find best solutions that were in average 9% better than those found by the TS and 7% better than the solutions found by TSG. It is interesting to note, however, that the solution found by TSG (in a single run) is about 9% better than the average solution obtained by the CS (out of 20 runs). The performance of the proposed TSG is indeed, remarkable for all instances tested: in a single run it obtains solutions that are competitive with the best solution found in 20 runs for the CS or the TS.

Finally, for the last and larger class of instances (Tonge), the proposed methods present better results in terms of best solutions found (the best solutions found by our TS were about 17% better than the best solutions found by the CS), in terms of the average solution (40% better) and computational times to converge (about 50% smaller). Indeed, for these larger instances even the **average** solution found (out of 20 runs) with the TS was usually of better quality than the **best** solution found by the CS out of 20 runs. This was always true for the larger instances (those of groups 5-8, which have 19

		CS				TS				TSG		
Family	**Group**	best	avrg	t(s)	$t_b(s)$	best	avrg	t(s)	$t_b(s)$	best	t(s)	$t_b(s)$
Roszieg	1	20.10	20.19	5.51	2.12	20.10	20.10	3.64	0.16	20.10	3.66	0.05
	2	31.50	33.31	5.42	2.06	31.50	31.52	3.69	0.38	31.50	3.59	0.10
	3	28.10	28.14	5.49	2.15	28.10	28.11	3.71	0.16	28.10	3.73	0.30
	4	28.00	28.12	5.49	1.85	28.00	28.00	3.78	0.03	28.00	3.73	0.02
	5	9.70	10.16	6.21	3.27	9.70	9.73	4.42	0.60	9.70	4.47	0.42
	6	11.00	11.88	6.18	3.55	11.00	11.01	4.45	0.57	11.00	4.45	0.53
	7	16.00	16.27	6.22	3.45	16.00	16.00	4.45	0.23	16.00	4.44	0.29
	8	15.10	15.36	6.23	3.32	15.10	15.15	4.43	0.69	15.20	4.50	0.19
	Average	**19.94**	**20.43**	**5.84**	**2.72**	**19.94**	**19.95**	**4.07**	**0.35**	**19.95**	**4.07**	**0.24**

Tabela 2: Results for the Roszieg family of instances

		CS				TS				TSG		
Family	**Group**	best	avrg	t(s)	$t_b(s)$	best	avrg	t(s)	$t_b(s)$	best	t(s)	$t_b(s)$
Heskia	1	102.30	103.34	6.20	3.01	102.30	102.79	4.57	0.59	102.90	4.72	0.37
	2	122.60	123.50	6.15	2.61	122.60	122.77	4.59	0.35	122.80	4.63	0.74
	3	172.50	173.23	6.18	2.62	172.50	172.60	4.56	0.78	172.50	4.53	0.51
	4	171.20	171.71	6.19	2.87	171.20	171.78	4.58	0.49	171.70	4.65	0.79
	5	35.00	36.12	7.58	4.59	35.00	36.77	5.81	1.71	37.90	5.90	1.20
	6	42.90	44.24	7.58	4.08	43.10	44.62	5.85	0.58	44.00	6.08	1.63
	7	75.20	76.17	7.56	3.52	75.20	76.59	5.78	0.43	76.10	5.79	0.61
	8	67.20	70.09	7.58	4.20	67.30	70.49	5.82	1.09	69.10	5.82	1.30
	Average	**98.61**	**99.80**	**6.88**	**3.44**	**98.65**	**99.80**	**5.19**	**0.75**	**99.63**	**5.27**	**0.89**

Tabela 3: Results for the Heskia family of instances

		CS				TS				TSG		
Family	**Group**	best	avrg	t(s)	$t_b(s)$	best	avrg	t(s)	$t_b(s)$	best	t(s)	$t_b(s)$
Wee-mag	1	31.90	36.94	76.73	55.94	35.30	38.44	57.89	39.93	36.30	57.06	16.29
	2	37.90	42.65	76.26	55.96	41.10	45.33	58.22	35.66	43.80	60.29	20.33
	3	55.90	63.87	76.51	56.31	58.40	64.82	58.36	29.66	63.10	57.53	16.78
	4	53.60	62.49	76.46	54.66	56.10	61.33	57.91	35.80	62.50	58.08	25.08
	5	17.00	21.50	111.38	76.32	19.90	23.83	71.72	43.77	15.50	71.29	0.02
	6	19.50	24.33	111.63	75.72	23.30	28.56	71.28	40.11	18.20	72.92	0.01
	7	28.90	36.24	111.87	78.79	32.60	39.15	71.14	26.96	26.70	72.22	0.01
	8	27.50	35.46	111.60	81.91	31.70	39.09	71.19	47.89	26.50	70.45	0.01
	Average	**34.03**	**40.43**	**94.06**	**66.95**	**37.30**	**42.57**	**64.71**	**37.47**	**36.58**	**64.98**	**9.81**

Tabela 4: Results for the Wee-mag family of instances

		CS				TS				TSG		
Family	**Group**	best	avrg	t(s)	$t_b(s)$	best	avrg	t(s)	$t_b(s)$	best	t(s)	$t_b(s)$
Tonge	1	109.60	155.40	61.98	42.97	100.10	108.30	47.77	21.29	106.50	48.53	22.76
	2	133.10	207.85	61.95	41.79	117.70	128.04	47.96	26.43	128.60	47.83	16.07
	3	183.30	264.60	61.82	44.29	171.50	187.42	47.95	28.87	189.00	47.52	17.39
	4	192.10	273.01	61.89	43.27	178.30	194.07	47.80	17.69	191.00	47.99	17.40
	5	55.60	95.68	96.55	67.39	41.70	47.62	58.59	27.47	46.10	58.30	16.13
	6	70.40	133.44	96.34	63.51	47.70	54.60	58.99	32.12	52.00	58.14	16.94
	7	108.00	184.45	96.22	64.94	75.50	82.10	58.74	32.25	81.20	58.07	14.76
	8	118.60	183.52	96.13	64.79	75.80	86.06	58.73	37.49	82.30	58.45	32.74
	Average	**121.34**	**187.24**	**79.11**	**54.12**	**101.04**	**111.02**	**53.32**	**27.95**	**109.59**	**53.10**	**19.27**

Tabela 5: Results for the Tonge family of instances

workers), for which the best and average solutions of the proposed TS were 28% and 20% better than the best solutions found by the CS of Chaves (2009).

6 Conclusions

A considerable part of the world population presents some kind of disability. Establishing conditions for these citizens to work is a task that is increasingly present in governments and corporations agendas. In this article, we consider the problem of programming assembly lines in sheltered work centers for the disabled. These centers might serve as a first step towards the integration of disabled workers in the conventional market. The problem of finding good assembly lines programming in this context is a hard combinatorial problem. We propose a tabu search algorithm that tries to incorporate four main characteristics: simplicity, flexibility, accuracy and speed. Although full priority was not given to accuracy and speed, as it is usually the case in the Operations Research literature, our results show that the method outperforms more sophisticated methods even in these two criteria.

Acknowledgments

This work was supported by CNPq, under grant "Jovens Pesquisadores 561672/2008-3". This support is gratefully acknowledged. The authors also thank an anonymous referee whose comments have helped improve both the content and the presentation of this article.

Referências

Baybars, I. (1986). A survey of exact algorithms for the simple assembly line balancing problem, *Management Science* **32**: 909–932.

Chaves, A. A. (2009). *Uma meta-heurística híbrida com busca por agrupamentos aplicada a problemas de otimização combinatória*, PhD thesis, Instituto Nacional de Pesquisas Espaciais.

Chaves, A. A. & Lorena, L. A. N. (2008). Hybrid metaheuristic for the prize collectin travelling salesman problem, *Lecture Notes in Computers Science* **4972**: 123–134.

Chaves, A. A. & Lorena, L. A. N. (2009). Clustering search algorithm for the capacitated centred clustering problem, *Computers & Operations Research* (forthcoming) .

Chaves, A. A., Miralles, C. & Lorena, L. A. N. (2007). Clustering search approach for the assembly line worker assignment and balancing problem, *Proceedings of the 37th International Conference on Computers and Industrial Engineering*, Alexandria, Egypt, pp. 1469 – 1478.

Chaves, A. A., Miralles, C. & Lorena, L. A. N. (2008). Uma metaheurística híbrida aplicada ao problema de balanceamento e designação de trabalhadores em linhas de produção, *Anais do XL SBPO*, João Pessoa, pp. 1479–1490.

Cordeau, J.-F., Gendreau, M., Laporte, G., Potvin, J.-Y. & Semet, F. (2002). A guide to vehicle routing heuristics, *Journal of the Operational Research Society* **53**: 512–522.

Corrêa, F. A., Chaves, A. A. & Lorena, L. A. N. (2009). Hybrid heuristics for the probabilistic maximal covering location-allocation problem, *Operational Research* forthcoming .

Costa, A. M. & Miralles, C. (2009). Job rotation in assembly lines employing disabled workers, *International Journal of Production Economics* (forthcoming) .

Fisher, M. L. (2004). The Lagrangian relaxation method for solving integer programing problems, *Management Science* **50**: 1861–1871.

Gates, L. B. (2000). Workplace accommodation as a social process, *Journal of Occupational Rehabilitation* **10**: 85–98.

Gendreau, M., Hertz, A. & Laporte, G. (1994). A tabu search heuristic for the vehicle routing problem, *Management Science* **40**: 1276–1290.

Glover, F. (1989a). Tabu search - part I, *ORSA Journal on Computing* **1**: 190–206.

Glover, F. (1989b). Tabu search - part II, *ORSA Journal on Computing* **2**: 4–32.

Glover, F. & Laguna, M. (1997). *Tabu Search*, Kluwer Academic Publications.

Miralles, C., Garcia-Sabater, J. P. & Andrés, C. (2005). Application of U-lines principles to the assembly line worker assignment and balancing problem (UALWABP). A model and a solving procedure, *Proceedings of the Operational Research Peripatetic Postgraduate Programme International Conference (ORP3)*, Valencia, Spain.

Miralles, C., Garcia-Sabater, J. P., Andrés, C. & Cardos, M. (2007). Advantages of assembly lines in sheltered work centres for disabled. A case study, *International Journal of Production Economics* **110**: 187–197.

Miralles, C., Garcia-Sabater, J. P., Andrés, C. & Cardos, M. (2008). Branch and bound procedures for solving the assembly line worker assignment and balancing problem: Application to sheltered work centres for disabled, *Discrete Applied Mathematics* **156**: 352–367.

Moreira, M. C. O. & Costa, A. M. (2009). Simple heuristics for the assembly line and worker assignment balancing problem, Working Paper .

Oliveira, A. & Lorena, L. (2004). Detecting promising areas by evolutionary clustering search, *Advances in Artificial Intelligence, Springer Lecture Notes in Artificial Intelligence Series* **3171**: 385–394.

Rekiek, B., Dolgui, A., Delchambre, A. & Bratcu, A. (2002). State of art of optimization methods for assembly line design, *Annual Reviews in Control* **26**: 163–174.

Scholl, A. & Becker, C. (2006). State-of-the-art exact and heuristic solution procedures for simple assembly line balancing, *European Journal of Operational Research* **168**: 666–693.

SERPRO (2004). http://www.serpro.gov.br/noticias-antigas/noticias-2004/20041221_02 (Last visited on April 7, 2009).

United Nations (2006). Convention on the rights of persons with disabilities.