Hybrid Metaheuristic for the Assembly Line Worker Assignment and Balancing Problem[⋆]

Antonio Augusto Chaves[1], Luiz Antonio Nogueira Lorena[1], and Cristobal Miralles[2]

[1] Laboratory of Computing and Applied Mathematics, National Institute for Space Research, São José dos Campos, Brazil
[2] ROGLE-Departamento Organización de Empresas, Universidad Politécnica de Valencia, Valencia, Spain
{chaves,lorena}@lac.inpe.br, cmiralles@omp.upv.es

Abstract. The Assembly Line Worker Assignment and Balancing Problem (ALWABP) appears in real assembly lines which we have to assign given tasks to workers where there are some task-worker incompatibilities and considering that the operation time for each task is different depending upon who executes the task. This problem is typical for Sheltered Work Centers for the Disabled and it is well known to be NP-Hard. In this paper, the hybrid method Clustering Search (CS) is implemented to solve the ALWABP. The CS identifies promising regions of the search space by generating solutions with a metaheuristic, such as Iterated Local Search, and clustering them into clusters that are then explored further with local search heuristics. Computational results considering instances available in the literature are presented to demonstrate the efficacy of the CS.

1 Introduction

The World Health Organization estimates that 10% of global population, around 610 million people worldwide, is disabled. Of these, 386 million people are within the active labor age range, but experience very high unemployment rates.

Current practices for the treatment of physically and/or mentally handicapped individuals prescribe meaningful job activity as a means towards a more fulfilling life and societal integration [1]. In some countries active policies have been launched by national governments in order to achieve better labor integration of the disabled. These practices have facilitated the creation of Sheltered Work Centers for 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 developed their personal skills, before being fully integrated into the conventional labor market.

⋆ The authors acknowledge FAPESP (process 2008/09242-9) for the financial support given to the development of this work.

This model of socio-labor integration tries to move away from the traditional stereotype that considers disabled people as unable to develop continuous professional work. Just as in any other firm, the SWDs compete in the market and must be flexible and efficient enough to adapt to market fluctuations and changes, the only difference being that the SWD is a Not-For-Profit organization. Thus, the potential benefits that may be obtained from increased efficiency usually implies growth of the SWD. This means more jobs for the disabled and the gradual integration of people with higher levels of disability; which are in fact the primary aims of the SWD.

Miralles et al. [2] revealed how in these centers the adoption of assembly lines provides many advantages, as the traditional division of work into single tasks can become a perfect tool for making certain worker disabilities invisible. In fact, an appropriate task assignment can even become a good therapeutic method for the rehabilitation of certain disabilities. But some specific constraints relative to time variability arise in this centers, in which case the balancing procedures applied in a SWD should be able to reconcile the following objectives:

1) to maximize the efficiency of the line by balancing the workload assigned to each available worker in each workstation;
2) to satisfy and respect the existent constraints in this environment due the human factors when assigning tasks to workers.

After analyzing some SWDs, Miralles et al. [2] observed some characteristics that can be found in this environment, which were the motivation for defining the Assembly Line Worker Assignment and Balancing Problem (ALWABP). This problem is known to be NP-hard and has a important application in the SWDs.

In general, an assembly line consists of a set tasks, each having an operation time, and a 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. The ALWABP consists of assigning tasks to workstations, which are arranged in a predefined order, so that the precedence constraints are satisfied and some give measure of effectiveness is optimized. Therefore, in the SWD some workers can be very slow, or even incapable, when executing some tasks, but very efficient when executing some other tasks. So, the balancing of the line consists of a double assignment: (1) tasks to workstations; and (2) workers to workstations. Always respecting the incompatibilities among tasks and workers.

The main characteristics of the ALWABP have been listed by Miralles et al. [2] as follows:

a) a single product is assembled on the line;
b) task operation times and precedence constraints are known deterministically;
c) there is a serial line layout with k workstations;
d) k workers are available and the operation times of a task depends on the workers executing it;
e) each worker is assigned to only one workstation;

f) each task is assigned to only one workstation, provided that the worker selected for that workstation is capable of performing the task, and that the precedence constraints are satisfied.

When we aim to minimize the number of workstations, the problem is called ALWABP-1; and when the objective is to minimize the cycle time given a set of workstations, the problem is called ALWABP-2, the latter situation being more common in SWDs.

This paper presents an application of the hybrid method Clustering Search (CS) [3,4] to solve the ALWABP-2. The CS consists of detecting promising areas of the search space using a metaheuristic that generates solutions to be clustered. These promising areas should be exploited with local search heuristics as soon as they are discovered. The Iterated Local Search (ILS) [5] was the metaheuristic chosen to generate solutions for the clustering process. In this paper some improvements to the CS are also proposed.

The remainder of the paper is organized as follows. Section 2 reviews previous works about ALWABP. Section 3 describes the CS method and section 4 presents the CS applied to the ALWABP. Section 5 presents the computational results. Conclusions are reported in section 6.

2 Literature Review

The ALWABP was recently introduced by Miralles et al. [2]. The authors presented a mathematical model for ALWABP and a case study based on a Spanish SWD. Miralles et al. [6] proposed a basic branch and bound procedure with three possible search strategies and different parameters for solving the ALWABP.

Chaves et al. [7,8] proposed the use of hybrid method CS for solving the ALWABP, which was implemented using Simulated Annealing (SA) [9] to generate solutions to the clustering process.

There are some other problems with double assignment of tasks and resources to workstations. For example, some cost-oriented models assume that different equipment sets, each with a different cost, can be assigned to a workstation. In this sense, total cost must be minimized by optimally integrating design (selecting the machine type to locate at each activated workstation) and operating issues (assigning tasks to observe precedence constraints and cycle time restrictions). When these decisions are connected, the terms Assembly Line Design Problem (ALDP) [10] or Assembly System Design Problem (ASDP) [11] are frequently used in the literature.

Although ALWABP can be classified with problems of these types, it is not a cost-oriented problem in which there are alternative machines with different associated costs and the total cost has to be minimized. Furthermore, in ALWABP the available resources are constrained: there are unique workers each which can be assigned only once. In some cases workers have similar characteristics; but, even in these cases, an infinite number of workers is not available, as assumed in most ASDP problems.

Variants of the ALWABP have been studied in the literature: Miralles et al. [12] analyze the design of U-shaped assembly lines in the context found at SWD while Costa and Miralles [13] proposes models and algorithms to plan job rotation schedules in SWD.

3 Clustering Search Algorithm

The Clustering Search (CS) [3] is a hybrid method that aims to combine metaheuristics and local search heuristics, in which the search is intensified only in areas of the search space that deserve special attention (promising regions). The CS introduces an intelligence and priority to the choice of solutions to apply local search, instead of choosing randomly or apply local search in all solutions. Therefore, it is expected an improvement in the convergence process associated with a decrease in computational effort as a consequence of a more rational employment of the heuristics.

In fact in this paper the CS method is improved making it more efficient, robust and user-friendly. This is done through the implementation of a random perturbation and a control of the efficiency of local search heuristics. Moreover, the functions of each component of CS was defined more clearly, redesigning the structure of the CS. These improvements change the CS into a method easier to learn and easier to use.

The CS attempts to divide the search space and locate promising search regions by framing them in clusters. For the purposes of the CS, a cluster can be defined by three attributes $C = (c, v, r)$. The center c_i is a solution that represents cluster C_i, identifying its location within the search space. Instead of keeping all solutions generated by the metaheuristic and grouping them into clusters, just a part of these solutions characteristics are inserted to the center. The volume v_i is the number of solutions grouped into the cluster C_i. This volume determines when a cluster becomes promising. The inefficacy rate r_i is a control variable that verifies if the local search is improving the center c_i. The value of r_i indicates the number of consecutive times that the local search was applied to the cluster C_i and did not improve the solution. This attribute avoids that the local search is being performed by more than r_{max} times in bad regions or regions that have already been exploited by the heuristic.

In order to group similar solutions in clusters, the CS needs some form of distance measuring between two solutions. Therefore, for two solutions s_1 and s_2, the distance function $d(s_1, s_2)$ is defined as the number of locations in which s_1 and s_2 differ. For example, the distance of $s_1 = \{1010110\}$ and $s_2 = \{1100100\}$ is three ($d(s_1, s_2) = 3$).

Initially, we need to define the number of clusters ($|C|$). And, the cluster centers should be generated in order to represent different regions of search space. This paper uses a greed method based on maximum diversity to create the initial centers, which generate a large set with n ($n >> |C|$) random solutions and a subset is selected with $|C|$ solutions that have the longest distance among themselves. In this method, the first solution is randomly chosen. The second solution must be the solution that offers the greatest distance for the first one. From the third solution, it is necessary to calculate the sum of distances among each candidate solution and the selected solutions. Then, the solution that has the biggest sum is also selected to be the initial center of one cluster.

The CS is an iterative method which has three main components: a metaheuristic, a clustering process and a local search method. The hybrid strategy of the CS can be described by the flowchart illustrated in figure 1.

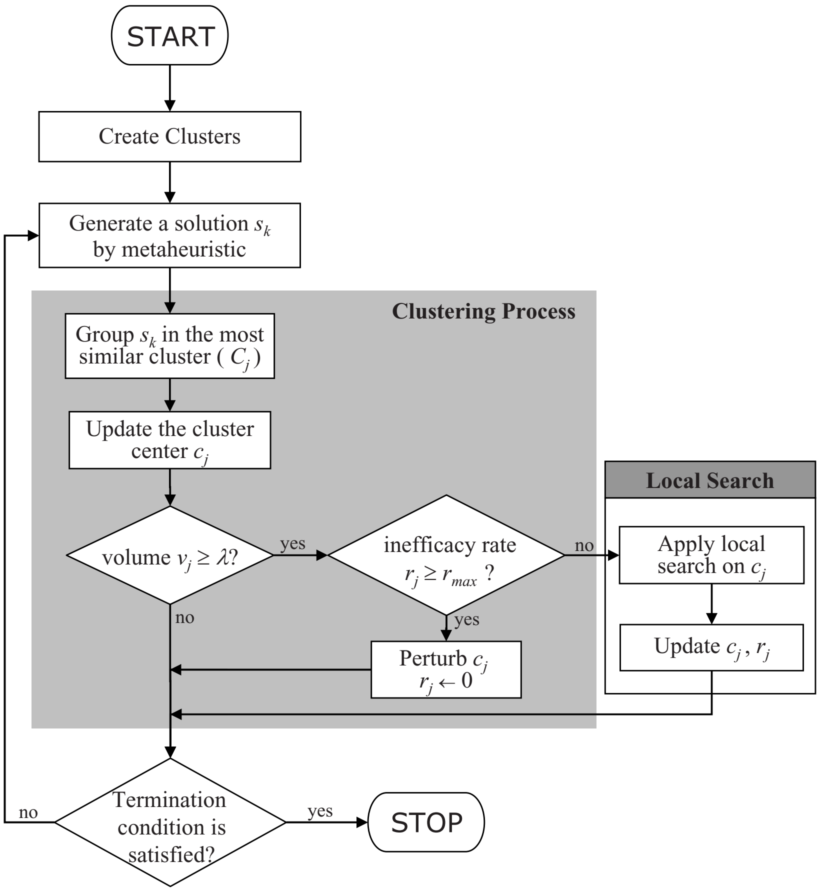

START
Create Clusters
Generate a solution s_k by metaheuristic
Clustering Process
Group s_k in the most similar cluster (C_j)
Update the cluster center c_j
volume $v_j \geq \lambda$?
yes
no
inefficacy rate $r_j \geq r_{max}$?
yes
no
Perturb c_j $r_j \leftarrow 0$
Local Search
Apply local search on c_j
Update c_j, r_j
Termination condition is satisfied?
no
yes
STOP

Fig. 1. Flowchart of the CS

A metaheuristic works as a solution generator to the clustering process. The algorithm performs independently of the others components and must be able to provide a great number of different solutions for enabling a broad analysis of the search space.

In each iteration of CS, a solution s_k is generated by a metaheuristic and sent to the clustering process. This solution is grouped in the most similar cluster

C_j, that is the cluster with the smallest distance d between the center c_j and the solution s_k. The clustering process aims to direct the search for supposedly promising regions.

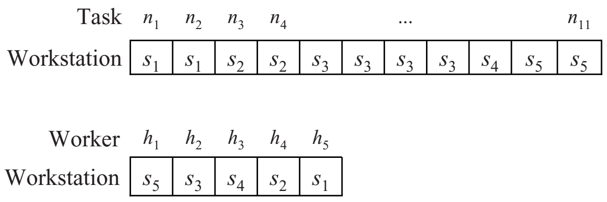
The insertion of a new solution into a cluster should cause a perturbation in its center. This perturbation is called assimilation and consists of updating the center c_j with attributes of solution s_k. This process is the Path-Relinking method [14], which generates several solutions along the path that connects the center c_j and the solution s_k. This process is responsible for intensify and diversify the search into the cluster, for the reason that the new center is the best-evaluated solution obtained along the path, even if it is worse than the current center.

Then, the volume v_j is analyzed. If it has reached the threshold λ, this cluster may be a promising region. And, if the local search has achieved success in recent r_{max} applications in this promising cluster ($r_j < r_{max}$), the center c_j is better investigated with local search to accelerate the convergence process. Otherwise, if $r_j \geq r_{max}$ a random perturbation is performed in c_j, allowing the center go to another region of search space. The local search obtains success in a cluster j when one finds a solution better than the best solution obtained so far in this cluster (c_j^*).

The local search is a problem-specific local search heuristic that provides the exploitation of promising region framed by the cluster. The heuristic is applied on the center c_j and this is updated if the heuristic finds a better solution. The Variable Neighborhood Descent (VND) [15] can be used in this component, to analyze a large number of solutions around the cluster.

4 CS Algorithm for ALWABP

A solution of the ALWABP is represented by two vectors. The first vector represents the task/workstation assignments and the second vector represents the worker/workstation assignments. Figure 2 shows an example of one solution with 11 tasks, 5 workers and 5 workstations.



Task
n_1 n_2 n_3 n_4 ... n_{11}
Workstation
s_1 s_1 s_2 s_2 s_3 s_3 s_3 s_3 s_4 s_5 s_5
Worker
h_1 h_2 h_3 h_4 h_5
Workstation
s_5 s_3 s_4 s_2 s_1

Fig. 2. An example of the solution representation

In the particular case of the CS developed for the ALWABP, we make use of penalties in the objective function. Let C_{time} be the cycle time and f_p and f_t be the infactibilities measures of the precedence constraints violations and the infeasible task/worker assignments; ω and δ be the multipliers for the values f_p and f_t. The objective function of ALWABP is defined as follows:

$$f(s) = C_{time} + (\omega * f_p + \delta * f_t) \tag{1}$$

The distance of two solutions (d) is the number of tasks assigned to different workstations between them. So, the distance increases when there is a large number of allocations to different workstations between the solutions.

4.1 Iterated Local Search

The CS uses the Iterated Local Search (ILS) [5] to generate solutions to the clustering process. ILS consists in the iterative application of a local search procedure to starting solutions that are obtained by the previous local optimum through a solution perturbation.

To apply an ILS algorithm, four components have to be specified: a procedure that generates an initial solution s_0, a perturbation that modifies the current solution $\hat{s}$ leading to some intermediate solution s', a local search procedure that returns an improved solution $\hat{s}'$, and an acceptance criterion that decides to which solution the next perturbation is applied.

In this paper we propose a method to generate an initial solution without violating the precedence network. The method is based in the Kilbridge-Wester heuristic [16] and it considers the number of tasks that precede each task. A ordered list is built, in ascending order, with tasks that have the smallest number of predecessors. If two or more tasks have the same number of predecessors, the order is chosen randomly. Then, we select a task of the list (starting from first up to last) and assigned it on an available workstation. The first tasks should be assigned to the first workstations, and so on. Thus, the workstations have the same number of assigned tasks. Finally, workers are randomly assigned to workstations. Therefore, the initial solution probably have only infeasibilities between tasks and workers.

In order to escape from local optima and to explore new regions of the search space, ILS applies perturbations to the current solution. A crucial issue concerns the strength of the perturbation. If it is too strong, ILS may behave like a random restart resulting in a very low probability of finding better solutions. On the other hand, if the perturbation is not strong enough, the local search procedure will rapidly go back to a previous local optimum.

We defined three types of perturbation movements that are randomly applied in our ILS procedure. The first consists in swapping two tasks between two workstations. The second moves one task from one workstation to another. And the third is obtained by swapping two workers between two workstation.

The strength of the perturbation is the number of solution components (tasks and workers) that are modified, and it is randomly defined in each iteration. A percentage β ($\beta \in [0.25, 0.75]$) of the number of tasks are altered by the perturbation.

We select the *Swap Worker* and *Swap Task* heuristics as the local search of ILS. This heuristics performs a descent from a solution s' until it reaches a local minimum ($\hat{s}'$). Further details of these methods are presented in section 4.3.

The acceptance criterion is biased towards diversification of the search since the best solution resulting from the local search phase is accepted if it improves the local minimum encountered so far or with a probability of 5%.

The condition used to stop the algorithm was the maximal number of iterations, which was defined as 5000 iterations. This is the termination condition of the CS.

4.2 Clustering Process

At each iteration of CS, the solution $\hat{s}'$ is grouped into the closest cluster C_j; that is, the cluster that minimizes the distance between the solution and the cluster center. The volume v_j is increased in one unit and the center c_j should be updated with new attributes of the solution $\hat{s}'$ (assimilation process).

The assimilation process uses the path-relinking method. The procedure starts by computing the symmetric difference between the center c_j and the solution $\hat{s}'$ $\Delta(c_j, \hat{s}')$; i.e., the set of moves needed to reach $\hat{s}'$ from c_j. A path of solutions is generated, linking c_j and $\hat{s}'$. At each step, the procedure examines all moves $m \in \Delta(c_j, \hat{s}')$ from the current solution s and selects the one that results in the best-cost solution, applying the best move to solution s. The set of available moves is updated. The procedure terminates when 30% of the solutions in the path have been analyzed, this is to stop the center from moving too far. The new center c_j is the best solution along this path. In this paper, one move is to swap the workstation in which one task of c_j is assigned by the workstation of this task in the $\hat{s}'$.

After performing the path-relinking, we must conduct an analysis of the volume v_j, verifying if this cluster can be considered promising. A cluster becomes promising when its volume reaches the threshold λ ($v_j \geq \lambda$). The value of λ was set equal to 10.

Then, if the volume v_j reached λ and the local search has been obtained success in this cluster ($r_j < r_{max}; r_{max} = 5$), an exploitation is applied in center c_j by local search heuristics. Otherwise, if the inefficacy rate r_j is greater than 5, we must apply a random perturbation to the center, swapping 30% of assigments task/workstation.

4.3 Local Search

The VND method is implemented as local search of CS, intensifying the search in neighborhood of a promising cluster C_j. Three types of moves are relevant in a VND for ALWABP: swap tasks, shift tasks and swap workers. So, our VND procedure uses three heuristics based on these moves, which seek to improve the center of a promising cluster.

The VND improvement methods are:

- *Swap Tasks*: to perform the best move, swapping two tasks that have been assigned to different workstations;
- *Shift Task*: to perform the best move, removing one task from a workstation and assigning it to another;
- *Swap Workers*: to perform the best move, swapping the workstation assignments of two workers.

```
algorithm CS
  create the initial clusters of CS
  { metaheuristic – ILS}
  generate an initial solution $s_0$
  $\hat{s} \leftarrow$ LocalSearch ($s_0$)
  while termination condition not satisfied do
      $s' \leftarrow$ Perturbation ($\hat{s}$)
      $\hat{s}' \leftarrow$ LocalSearch ($s'$)
      $\hat{s} \leftarrow$ AcceptanceCriterion ($\hat{s}, \hat{s}'$)
      { clustering process }
      find the most similar cluster $C_j$ to the solution $\hat{s}'$
      insert $\hat{s}'$ into $C_j$ ( $v_j \leftarrow v_j + 1$ )
      update the center $c_j$ ( $c_j \leftarrow$ PR ($c_j$, $\hat{s}'$) )
      if $v_j \geq \lambda$ then
          $v_j \leftarrow 0$
          if $r_j \geq r_{max}$ then
              apply a random perturbation in $c_j$
              $r_j \leftarrow 0$
          else
              { local search }
              $\hat{c}_j \leftarrow$ VND ( $c_j$ )
              if $f(\hat{c}_j) < f(c_j)$ then
                  $c_j \leftarrow \hat{c}_j$
              end if
              if $f(\hat{c}_j) < f(c_j^*)$ then
                  $r_j \leftarrow 0$
                  $c_j^* \leftarrow \hat{c}_j$
              else
                  $r_j \leftarrow r_j + 1$
              end if
          end if
      end if
  end while
```

Fig. 3. CS algorithm for the ALWABP

If some heuristic obtains a better solution, VND returns to the first heuristic and the search continues from the better solution; otherwise, it changes to the next heuristic. The VND stopping condition is that there are no more improvements to be made to the incumbent solution. The center c_j is updated if the new solution is better than the previous one. On the other hand, the inefficacy rate r_j is reset if and only if the VND improves the best center found so far in this cluster (c_j^*). Otherwise, r_j is increased in one unit.

Figure 3 presents the CS pseudo-code.

5 Computational Results

The CS algortithm was coded in C++ and computational tests were executed on a 2.6 GHz Pentium 4 with 1 GB RAM. In order to test the CS, we have used the problem sets proposed by Chaves et al. [7] and available in the literature. These sets are composed of four families: Roszieg, Heskia, Wee-Mag and Tonge. The ALWABP benchmark is composed of 320 instances (80 in each family), enabling to extract conclusions about the overall behavior of CS against different kind of problem. The characteristics for each problem set (number of tasks (|N|), number of workers (|W|) and the order strength (OS) of the precedence network) are listed in table 1. The OS measures the relative number of precedence relations. That is, problems with a large strength are basically expected to be more complex than such with small OS values [17].

Table 1. ALWABP: Instances characteristics

Family	**|N|**	**|W|**	**OS**
Roszieg	25	4 (group 1-4) or 6 (group 5-8)	71,67
Heskia	28	4 (group 1-4) or 7 (group 5-8)	22,49
Tonge	70	10 (group 1-4) or 17 (group 5-8)	59,42
Wee-Mag	75	11 (group 1-4) or 19 (group 5-8)	22,67

Two exact approaches were tested in Chaves et al. [7,8]: a branch and bound algorithm proposed by Miralles et al. [6] and the commercial solver CPLEX 10.1 [18]. The results of CPLEX were better than the results of branch and bound algorithm. CPLEX found the optimal solutions only for instances of the smaller Roszieg and Heskia families and had memory overload in instances of the Tonge e Wee-Mag families, finding very poor feasible sub-optimal solutions. Sometimes CPLEX failed in finding feasible solutions even after several hours of computational time for the larger instances. Chaves et al. [7,8] also present the results of a version of the CS algorithm for ALWABP, and we use these results for comparison. The tests were also executed on a 2.6 GHz Pentium 4 with 1 GB RAM.

Tables 2 - 5 present the results for each set of instances. In the tables, the results are averaged in each line for each group of 10 instances with same

Table 2. ALWABP: Results for the Roszieg family of instances

			CS				**ILS**			**Chaves et al. [7,8]**			
Family	**Group**	**best-known**	best	avrg	t_b(s)	t(s)	best	avrg	t(s)	best	avrg	t_b(s)	t(s)
Roszieg	1	20.1	20.1	20.2	0.8	3.8	20.1	20.4	3.4	20.1	20.2	2.2	5.2
	2	31.5	31.5	32.5	0.9	3.7	32.4	38.9	3.2	31.5	34.3	2.0	5.1
	3	28.1	28.1	28.5	0.6	3.8	28.2	28.7	3.3	28.1	28.1	2.0	5.2
	4	28.0	28.0	28.0	0.2	3.8	28.0	28.1	3.3	28.0	28.1	1.9	5.2
	5	9.7	9.7	10.7	1.3	5.2	10.3	11.8	4.6	9.7	10.2	3.5	6.0
	6	11.0	11.0	12.1	1.4	5.2	11.5	14.3	4.5	11.0	11.9	3.6	6.0
	7	16.0	16.0	16.9	1.5	5.2	16.5	18.7	4.6	16.0	16.2	3.5	6.0
	8	15.1	15.1	15.6	1.9	5.2	15.3	17.7	4.6	15.1	15.4	3.4	6.0
	average	**19.94**	**19.94**	**20.57**	**1.09**	**4.48**	**20.29**	**22.32**	**3.94**	**19.94**	**20.57**	**2.75**	**5.59**

Table 3. ALWABP: Results for the Heskia family of instances

			CS				**ILS**			**Chaves et al. [7,8]**			
Family	**Group**	**best-known**	best	avrg	t_b(s)	t(s)	best	avrg	t(s)	best	avrg	t_b(s)	t(s)
Heskia	1	102.3	102.3	102.8	1.3	5.0	102.3	103.0	6.3	102.3	103.5	3.026	5.802
	2	122.6	122.6	123.8	1.4	5.0	122.7	124.2	6.3	122.6	123.7	2.482	5.736
	3	172.5	172.5	175.5	1.7	5.0	172.6	176.4	6.4	172.5	173.1	2.62	5.779
	4	171.2	171.2	171.7	1.4	5.1	171.3	171.8	6.4	171.2	171.8	2.668	5.789
	5	34.9	34.9	37.8	4.4	9.0	35.3	38.6	7.5	34.9	36.4	4.584	7.448
	6	42.6	42.6	44.7	3.4	9.0	43.6	45.7	7.5	42.6	44.3	4.232	7.386
	7	75.2	75.2	77.7	2.9	9.0	76.7	78.6	7.5	75.2	76.4	3.554	7.361
	8	67.2	67.2	70.7	3.6	9.0	68.1	72.4	7.5	67.2	70.2	4.054	7.376
	average	**98.56**	**98.56**	**100.59**	**2.51**	**7.00**	**99.08**	**101.35**	**6.92**	**98.56**	**99.92**	**3.40**	**6.58**

characteristics. The tables list for each method the best solution found (*best*), the average solution found out of the 20 runs (*avrg*) and the total computational time (t) in seconds. For our CS method and CS proposed by Chaves et al. [7,8], the tables also list the time in which the best solution was found (t_b).

The results in the tables show the efficacy of the proposed CS. We can observe that our CS and the method proposed by Chaves et al. [7,8] found the best-known solutions for all instances of the Roszieg and Heskia families, and these are optimal solutions as proved by the solver CPLEX. However, our CS finds the best solution in shorter time. These instances are easy to solve, but the tests are important to show that CS is able to obtain good solutions in terms of quality, as a result of the comparison with the optimal solutions.

For the Tonge family, the proposed method found best-known solutions in 76 of 80 instances. The solutions are better than the previously best-known solution in 73 tested instances. It is interesting to note that the solution found by our CS are about 29% better than the best solutions obtained by Chaves et al. [7,8]. Our method also were better in terms of average solutions (590% better), mainly, beacuse the method of Chaves et al. [7,8] found infeasible solutions in some instances.

Table 4. ALWABP: Results for the Tonge family of instances

			CS				ILS			Chaves et al. [7,8]			
Family	**Group**	**best-known**	best	avrg	t_b(s)	t(s)	best	avrg	t(s)	best	avrg	t_b(s)	t(s)
Tonge	1	96.7	96.7	116.6	64.0	122.2	120.0	135.3	102.5	107.5	732.8	40.6	58.0
	2	116.0	116.0	141.8	64.6	122.6	151.8	174.8	101.7	141.8	826.0	41.8	59.3
	3	167.1	167.7	199.4	66.2	122.8	214.6	236.5	102.6	179.5	778.7	39.4	58.0
	4	174.0	174.0	206.0	65.7	123.0	220.7	244.3	102.6	206.4	755.2	38.1	57.9
	5	41.3	41.3	51.3	101.1	183.0	64.2	71.1	151.1	71.9	877.7	55.7	83.9
	6	48.5	48.5	61.6	105.3	184.7	74.2	87.4	151.4	83.9	913.2	56.3	86.0
	7	77.8	77.8	93.0	100.1	184.5	113.7	129.3	151.4	132.6	912.0	53.6	84.9
	8	77.5	77.9	95.6	100.3	184.9	116.1	131.3	151.5	113.8	875.9	59.1	84.9
	average	**99.86**	**99.99**	**120.64**	**83.42**	**153.48**	**134.41**	**151.25**	**126.86**	**129.68**	**833.93**	**48.06**	**71.61**

Table 5. ALWABP: Results for the Wee-Mag family of instances

			CS				ILS			Chaves et al. [7,8]			
Family	**Group**	**best-known**	best	avrg	t_b(s)	t(s)	best	avrg	t(s)	best	avrg	t_b(s)	t(s)
Wee-Mag	1	29.0	29.0	32.7	94.3	163.2	31.2	35.5	51.0	33.6	59.0	52.8	69.5
	2	34.6	34.6	38.4	91.4	160.8	37.4	41.0	51.2	38.5	45.9	59.0	71.3
	3	50.8	50.8	56.7	96.0	160.4	54.3	61.3	51.0	56.2	67.1	57.3	69.9
	4	49.6	49.6	55.6	103.9	158.8	52.7	58.9	51.2	55.2	73.5	53.1	69.8
	5	13.1	13.1	20.9	141.2	248.6	16.7	45.4	64.8	19.4	504.2	59.0	99.4
	6	14.6	14.6	18.2	155.2	249.0	18.7	23.7	65.0	21.8	544.9	66.8	102.2
	7	21.2	21.2	27.1	148.0	244.8	25.1	34.1	64.6	30.6	370.0	69.7	100.9
	8	21.6	21.6	26.8	140.6	243.4	24.9	33.7	64.7	27.9	443.6	68.9	101.2
	average	**29.31**	**29.31**	**34.56**	**121.31**	**203.65**	**32.63**	**41.70**	**57.92**	**35.40**	**263.51**	**60.82**	**85.52**

Finally, for the last and larger class of instances (Wee-Mag), our CS found the best-known solutions for all instances and improved the best-known solution so far in 77 instances. The improvement of our CS in terms of quality of the solutions was about 20%. Again, the average solutions of our CS were also much better than the results of Chaves et al. [7,8] (660% better).

We can observe that ILS without the clustering process gave solutions in poorer quality than CS in all problem families. These results show the importance of the clustering process and the local search method for the CS. Indeed, the best solution is always found first by path-relinking or VND.

The proposed method was robust, producing average solutions close to best solutions. The computational times of our CS were competitive, finding good solutions in few seconds for the smaller instances and in a reasonable time for the larger instances. However, the improvements proposed for the CS became it slower than the CS proposed by Chaves et al. [7,8], this can be observed by analyzing of the computational time for the Tonge and Wee-Mag families.

The convergence of CS had an interesting behavior, allowing to find good solution in 25% of total computational time for the Roszieg family (best case) and in 60% of the total computational time for the Wee-Mag family (worst case). The features and parameters settings of CS are implemented in order to speed up the optimization process and avoid premature convergence to local optimum.

6 Conclusions

This paper presents a solution for the Assembly Line Worker Assignment and Balancing Problem (ALWABP) using the Clustering Search (CS) algorithm. The CS has been applied with success in some combinatorial optimization problems, such as the pattern sequencing [3], the prize collecting traveling salesman [19], the flowshop scheduling [20], the capacitated p-median [4], and others.

The idea of the CS is to avoid applying a local search heuristic to all solutions generated by a metaheuristic. The CS detects the promising regions in the search space and applies local search only in these regions. Then, to detect promising regions becomes an interesting alternative, preventing the indiscriminate application of the heuristics.

This paper reports results of different classes of instances to the ALWABP found by the CS. The CS got the best-known solution in 314 of 320 instances, and it defined new best solutions for 306 instances. The results show that our CS approach is competitive for solving the ALWABP.

We proposed some improvements for the CS, making it more efficient, robust and user-friendly. The results found by this new version are better than the CS proposed by Chaves et al. [7,8], in terms of quality of solution and in terms of robustness of the method.

In practical sense, the short computational times achieved enable rapid balancing of the assembly line. According to Miralles et al. [6], this is very important if we take into account that, due to the high absenteeism and the periodic physical and psychological checking of workers, the SWD manager knows only at the beginning of every working day which workers are available. Therefore approaches like CS, which provide good solutions in short computational times, are very desirable for the aids to set up an assembly line early in the day.

Apart from implementation in industrial software, further studies can be done analyzing other metaheuristics to generate solutions for the CS clustering process (e.g., Ant Colony System, Tabu Search and Genetic Algorithm), and also exploring other problems to which CS could be applied.

References

1. Bellamy, G.T., Horner, R.H., Inman, D.P.: Vocational habilitation of severely retarded adults: a direct service technology. University press, Baltimore (1979)
2. Miralles, C., García-Sabater, J., Andrés, C., Cardós, M.: Advantages of assembly lines in sheltered work centres for disabled: a case study. International Journal of Production Economics 110(1-2), 187–197 (2007)
3. Oliveira, A.C.M., Lorena, L.A.N.: Hybrid evolutionary algorithms and clustering search. In: Grosan, C., Abraham, A., Ishibuchi, H. (eds.) Hybrid Evolutionary Systems - Studies in Computational Intelligence, pp. 81–102. Springer, Heidelberg (2007)
4. Chaves, A.A., Lorena, L.A.N., Corrêa, F.A.: Clustering search heuristic for the capacitated p-median problem. Advances in Soft Computing 44, 136–143 (2008)

5. Lourenço, H., Martin, O., Stützle, T.: Iterated local search. In: Glover, F., Kochenberger, G.A. (eds.) Handbook of Metaheuristics. International Series in Operations Research & Management Science, vol. 57, pp. 320–353. Springer, New York (2003)
6. Miralles, C., García-Sabater, J.P., Andrés, C., Cardós, M.: 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(3), 352–367 (2008)
7. Chaves, A.A., Miralles, C., Lorena, L.A.N.: Clustering search approach for the assembly line worker assignment and balancing problem. In: Proceedings of 37th International Conference on Computers and Industrial Engineering, Alexandria, Egypt, pp. 1469–1478 (2007)
8. Chaves, A.A., Miralles, C., Lorena, L.A.N.: Uma meta-heurística híbrida aplicada ao problema de balanceamento e designação de trabalhadores em linha de produção. In: Proceedings of XL Simpósio Brasileiro de Pesquisa Operacional, João Pessoa, Brazil, pp. 143–152 (2008)
9. Kirkpatrick, S., Gellat, D.C., Vecchi, M.P.: Optimization by simulated annealing. Science 220, 671–680 (1983)
10. Baybars, I.: A survey of exact algorithms for the simple assembly line balancing problem. Management Science 32, 909–932 (1986)
11. Pinnoi, A., Wilhelm, W.E.: A family of hierarchical models for assembly system design. International Journal of Production Research 35, 253–280 (1997)
12. Miralles, C., García-Sabater, J.P., Andrés, C.: Application of u-lines principles to the assembly line worker assignment and balancing problem: a model and a solving procedure. In: Proceedings of the Operational Research Peripatetic Postgraduate Programme International Conference (ORP3), Valencia, Spain (2005)
13. Costa, A.M., Miralles, C.: Job rotation in assembly lines employing disabled workers. International Journal of Production Economics (forthcoming)
14. Glover, F.: Tabu search and adaptive memory programing. In: Barr, R.S., Helgason, R.V., Kennington, J.L. (eds.) Interfaces in Computer Science and Operations Research, pp. 1–75. Kluwer, Dordrecht (1996)
15. Mladenovic, N., Hansen, P.: Variable neighborhood search. Computers and Operations Research 24, 1097–1100 (1997)
16. Kilbridge, M., Wester, L.: The balance delay problem. Management Science 8(1), 69–84 (1961)
17. Scholl, A.: Data of assembly line balancing problems. University Press 16, TU Darmstadt (1993)
18. ILOG France: Ilog Cplex 10.0: user's manual (2006)
19. Chaves, A.A., Lorena, L.A.N.: Hybrid metaheuristic for the prize collecting travelling salesman problem. In: van Hemert, J., Cotta, C. (eds.) EvoCOP 2008. LNCS, vol. 4972, pp. 123–134. Springer, Heidelberg (2007)
20. Filho, G.R., Nagano, M.S., Lorena, L.A.N.: Evolutionary clustering search for flowtime minimization in permutation flow shop. In: Bartz-Beielstein, T., Blesa Aguilera, M.J., Blum, C., Naujoks, B., Roli, A., Rudolph, G., Sampels, M. (eds.) HCI/ICCV 2007. LNCS, vol. 4771, pp. 69–81. Springer, Heidelberg (2007)