PARTICLE SWARM OPTIMIZATION WITH TURBULENCE FACTOR FOR TWO-DIMENSIONAL GUILLOTINE SINGLE KNAPSACK PROBLEMS

David Álvarez-Martínez
São Paulo State University
College of Engineering of Ilha Solteira - São Paulo – Brazil
david.unesp@gmail.com

Rubén Augusto Romero Lázaro
São Paulo State University
College of Engineering of Ilha Solteira - São Paulo – Brazil
ruben.feis@gmail.com

ABSTRACT

This study presents the (un)constrained (un)weighted k-staged fixed and rotated two-dimensional guillotineable single knapsack problem. A suitable encoding based on slicing tree is presented. A hybrid algorithm based on the Particle Swarm Optimization (PSO) and Genetic Algorithms (GA) is developed. This algorithm used the main characteristics of PSO and introduced the flight turbulence factor through the concept of mutation of GA. The computational results on large sets of test cases show that the methodology has effectiveness and robustness to solve the two-dimensional knapsack problem.

KEYWORDS: two-dimensional guillotine knapsack problem, particle swarm optimization, turbulence factor.

RESUMO

Este trabalho apresenta o problema da mochila bidimensional guilhotinada com suas diversas variações: com demanda de peças restrita e irrestrita, com e sem custos associados nas peças, com padrões de corte de k-estágios, e com e sem rotação de peças. Uma codificação baseada em árvores de corte é apresentada. Um algoritmo híbrido de Enxame de Partículas (PSO) e Algoritmos Genéticos (GA) é desenvolvido. Este algoritmo usa as principais características do PSO e introduz o fator de turbulência de vôo através do conceito de mutação dos GA. Os resultados computacionais sobre um grande conjunto de instâncias de teste mostram que a metodologia proposta é efetiva e robusta para resolver os problemas da mochila bidimensional.

PALAVARAS CHAVE: mochila bidimensional guilhotinada, enxame de partículas, fator de turbulência.

AD & GP - OR in Administration & Production Management

1. Introduction

The *k*-staged two-dimensional guillotineable single knapsack problem is used to solve cutting problems when the material employed is one rectangular piece where the smaller rectangular pieces must be located, knowing their size, their associated cost, and restricting the number of items of the same type. In addition, the cutting pattern of the solution must be guillotine type and the pieces must be generated using at maximum *k*-stages of cut. The characteristics of this problem are:

i) The associated cost can be related or not with the area of the piece that is going to be located; if the cost is equal to the area of the piece the problem will be solved without weights (unweighted version) and if the cost is different to the area of the item the problem will be solved with weights (weighted version).

ii) The orientation of the pieces is fixed, when a piece of length l and width w is different from a piece of length w and width l (fixed version). When the dimensions (l, w) and (w, l) represent the same piece, the problem includes the pieces rotation (rotated version).

iii) The cutting patterns are guillotine type, each cut produces two sub-rectangles. The cuts are done from one edge to the opposite edge of the original rectangle (only guillotine cuts).

iv) There exists a maximum number of stages k, taking the constant $k < \infty$ as the sum of all the vertical and/or horizontal parallel cuts. If k is equal to two, the problem is two-staged (and has many real applications). If k is equal to three, the problem is three-staged. Finally, if a large value is assumed for k, represents a non-staged problem.

v) If there is a maximum limit b_i of the quantity of pieces that will be cut from type i the problem is called restricted (constrained version) and if not exist a limit or it can be assume a large number for b_i, it means that the demand is unrestricted (unconstrained version).

There are two general techniques used to solve constrained problems: top-down and bottom-up approaches. In Christofides and Whitlock (1977) is originally proposed the top-down. The top-down approach requires an enormous amount of memory, due to the fact that, all possible cuts that can be made on the stock plate are enumerated by means of a tree in which branching corresponds to guillotine cuts and the nodes represent sub-rectangles obtained through the guillotine cut, for this reason its implementation is not attractive. The bottom-up approach has been more used, in works as: Wang (1983); Viswanathan and Bagchi (1993); Hifi (1997a); Fayard *et al.* (1998); Hifi and Roucairol (2001); Cui (2007); Morabito and Pureza (2007) and Cui (2008). In this approach, all possible combinations of smaller rectangles are generated to obtain larger rectangles until no more guillotine patterns can be obtained, one of its problems is the huge amount of required time. Different heuristics approximations have been developed to solve the problem; one of the best is the presented by Alvarez-Valdés *et al.* (2002), which presented the GRASP and Tabu Search algorithms with a good performance.

In contrast, for the unconstrained problems, Gilmore and Gomory (1965; 1966) proposed a recursive exact algorithm based on dynamic programming. Their algorithm is applicable to both weighted and unweighted versions. Herz (1972) proposed a recursive tree search method, his method is more effective than Gilmore and Gomory's algorithm for the unweighted problem, but does not apply to weighted cases. Beasley (1985) proposed an algorithm that is a modified version of Gilmore and Gomory's algorithm. Hifi and Zissimopoulos (1996) proposed a recursive exact algorithm that uses a dynamic programming procedure and efficient lower and upper bounds. Young-Gun and Kang (2002) improved Hifi and Zissimopoulos' recursive algorithm by applying a more efficient upper bound. This is presently the most efficient exact algorithm for the unconstrained problem. Young-Gun and Kang (2003)

proposed an algorithm based on Christofides and Whitlock (1977) that starts from an initial solution of good quality and uses the bottom-up algorithm as strategy to generate the branches, decreasing the number of nodes to explore.
Recently, various authors have been working with particle swarm optimization (PSO), due to its performance for solving continuous optimization as well as discrete optimization problems, Zhan *et al.* (2009). Different studies have successfully employed PSO and its variants to optimize process parameters of various manufacturing processes such as grinding Xie *et al.* (2002), welding Kennedy and Mendes (2002), boring Hu and Eberhart (2002), turning Clerc (1999) and cache memory tuning Carlisle and Dozier (2000).

This article uses a binary tree encoding, called slicing tree structure, which decomposes the problem into smaller packing problems to solve them through of hybrid metaheuristic algorithms that combine: the main characteristics of the PSO and the flight turbulence factor through the concept of mutation of GA (using an adaptation of the equation for updating the threshold variable from threshold accepting algorithm as mutation operator). In order to proof the efficiency and quality of the results obtained with the proposed methodology, study cases of the specialized literature were used. Solutions of good quality were obtained, and some of them have never been reported. The structure of this article is as follows: problem description, a general description of the methodology used to solve the *k*-staged two-dimensional guillotineable single knapsack problem, results analysis and conclusions.

2. Problem Description

The *k*-staged two-dimensional guillotineable single knapsack problem is formally defined as a finite set of n rectangular items of given dimensions (length l_i and width w_i, where i=1,2,…,n), a demand b_i and an associated profit c_i, which are cut from a rectangular initial plate with length L and width W. An item (l_i, w_i) is equal to a piece (w_i, l_i). The objective is to find a cutting pattern of the plate maximizing the profit function, see the equation (1), where z_i is a binary variable that indicates if the piece i would be cut or not.

$$max \quad \sum_{i=1}^{n} c_i \cdot z_i \tag{1}$$

Subject to:

- The packed (cut) pieces cannot trespass the limits of the plate.
- The pieces cannot overlap to each other.
- Only cuts of guillotine type are permitted.
- The cutting pattern should be obtained on k stages of cut.

Are also taken in count the variants of the problem, where: the associated costs are related or not to the area of the pieces (weighted and unweighted versions). The pieces can have or not fixed orientation (fixed and rotated versions). There is or not a demand of pieces of the same type (constrained and unconstrained versions). Furthermore, all these variants are solved for different values of *k*-stages (two-staged, three-staged and non-staged versions). In order to simplify, the notation presented by Hifi and Roucairol (2001) was used to enumerate the proposed problems on this study, where the most common *k* values are two and three.

• FCU_kSTDC: represents the Fixed Constrained Unweighted *k*-Staged Two-Dimensional Cutting problem;

• RCW_kSTDC: denotes the Rotated Constrained Weighted *k*-Staged Two-Dimensional Cutting problem;

• FCU_nSTDC: corresponds to the Fixed Constrained Unweighted Non-Staged Two-Dimensional Cutting problem;

• RCW_nSTDC: denotes the Rotated Constrained Weighted Non-Staged Two-Dimensional Cutting problem.

• FUU_kSTDC: represents the Fixed Unconstrained Unweighted *k*-Staged Two-Dimensional Cutting problem;

• RUW_kSTDC: denotes the Rotated Unconstrained Weighted *k*-Staged Two-Dimensional Cutting problem;

• FUU_nSTDC: corresponds to the Fixed Unconstrained Unweighted Non-Staged Two-Dimensional Cutting problem;

• RUW_nSTDC: denotes the Rotated Unconstrained Weighted Non-Staged Two-Dimensional Cutting problem.

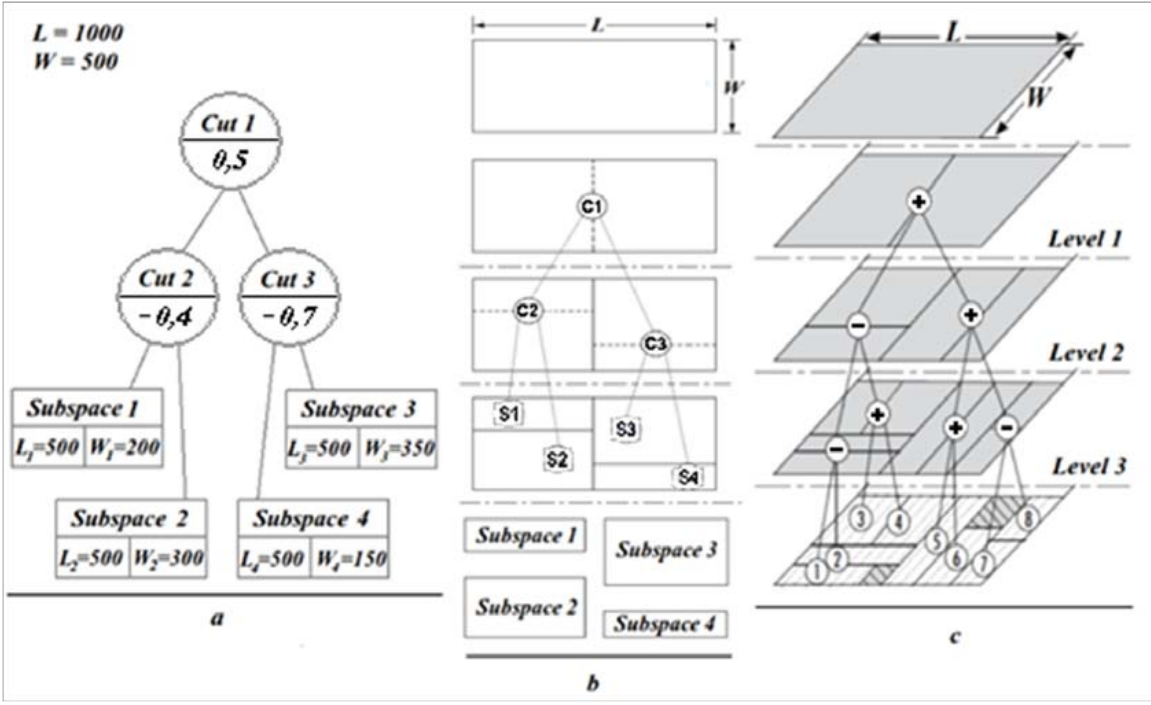

Figure 1. Illustration of slicing trees. (a) Numeric example of a slicing tree of two levels. (b) Slicing tree and subspaces generated (c) Slicing tree of three levels.

3. Methodology

The encoding and the solution methodology used on this study are presented below.

3.1 Encoding

Wong *et al.* (1988) presented a data codification for the floorplan design problem called slicing tree. A slicing tree is defined as: a tree with root, where each intern node (parental node) represents the position and how the cut will be done on the material (horizontal or vertical), meanwhile the leaves nodes (terminal nodes) represent the dimensions of the sub-spaces generated for cutting the grouped pieces. Therefore, the slicing tree contains on its internal nodes

the information about the orientation of the cut (perpendicular to the length or to the width) and the distance where the cut should be done. By other hand, its leaves nodes contain the dimensions of the resulting subspaces.

A cut is represented by a number between [-1, 1], where the sign represents the orientation of the cut and its value the distance for making the cut, thus, if the cut number 1 is equal to -0.13, represents a perpendicular cut to the width of the plate with a distance of 13% of it. The Figure 1 shows the proposed encoding.

One of the main advantages of using the slicing tree is the generation of cutting patterns guillotine type. Different proposed methodologies have corroborated the effectiveness of the slicing tree codification, especially the ones presented by Berthold (1995); Cui (2007); and Cui (2008).

3.2. Objective Function Calculation

After obtaining the sub-spaces, the placement of the pieces should be performed, this process is developed through a constructive best-fit algorithm, keeping the guillotine type restrictions and packing the biggest quantity of pieces for each sub-space. The constructive best-fit algorithm consists on finding the set of identical pieces that maximizes the area of the subspace *j* (represented by its length and width respectively) with the best associated profit. The calculation of the objective function consists on applying the equation (2) to each subspace, then the value of the objective function is the sum of the areas (or associated profits) packed on the plate.

$$max\left\{min\left\{\frac{L_j}{l_i}\cdot\frac{W_j}{w_i}, b_i\right\}, c_i\right\}; i = 1, 2, \cdots, n \qquad 2)$$

Different proposals that use the slicing tree encoding try to find the optimal tree, having this process a difficult solution, as: Berthold (1995) and Cui (2007; 2008). In contrast, in this work is restricted and reduced the number of trees during the optimization, after making a statistical study, the slicing tree is defined as a complete binary tree with three levels.

3.3 Optimization Scheme

The proposed codification in this study guarantees the feasibility of the guillotine type constraints and the maximum number of cutting stages. The optimization scheme for the knapsack problems is described below.

Particle Swarm Optimization and Turbulence Factor

The algorithm proposed on this work use different metaheuristic optimization techniques, combining the features of the particle swarm optimization and the mutation operator from the genetic algorithms. The first one is considered the main algorithm and the concept of mutation is used as flight turbulence factor.

The PSO algorithm uses a simple mechanism that emulates the behavior of sets of birds and fishes when together are looking for food or running away from predators, as they have a common goal, the algorithm has the objective function.

This algorithm can be computationally inefficient because may easily get trapped into local optima when solves cutting or packing problems, which have multimodal solution spaces, however, accelerates the convergence speed and skip the premature local optima, included a parameter control of the algorithm, improved of the topological structure and the combination of the search operators have become on the three most prominent and promising aims. In the presented work all the three aspects are taken into account.

In order to improve the efficiency and accelerate the search process is fundamental to determinate the state of the evolution and the best values for the parameters, aiming to avoid possible local optima in the convergence state, Zhan *et al.* (2009).

Bratton and Kennedy (2007) redefined the standard for PSO by adopting *lbest* (local best) topology, where each particle has access to the information from its immediate neighbors only, also: Zhan *et al.* (2009) and Xie *et al.* (2002), and with success: Kennedy and Mendes (2002); and Mendes *et al.* (2003), however, *lbest* topology should not be considered as an optimal choice in all situations is statement by Bratton and Kennedy (2007).

The algorithm has been modified introducing operators such as selection by Angeline (1998), crossing by Chen *et al.* (2007), mutation by Andrews (2006), local search by Liang and Suganthan (2005), restart of some operators by Carlisle and Dozier (2000) and Hu and Eberhart (2002), and the restart of all the operators of the process by Hu and Eberhart (2002) and Clerc (1999). All this hybrid operations are usually implemented during each generation as Andrews, (2006), inside a fixed interval as Liang and Suganthan (2005) or simply controlled by a defined fitness function for each specific case.

In PSO, a set of particles represent potential solutions (in this work is represented by a slicing tree, i.e., a cutting pattern), where each particle *i* is associated to two vectors, the speed vector $V_i = [v_i^1, v_i^2, \cdots, v_i^D,]$ and the position vector $X_i = [x_i^1, x_i^2, \cdots, x_i^D,]$, where D is the number of characteristics that determine the dimension of the solution space. The speed and position of each particle are initialized with random values between [-1, 1] (encoding reasons) with a uniform distribution. During the evolution process, the speed and the position of the *i-th* particle are updated through equations (3) and (4).

$$v_i^k = wv_{i-1}^k + c_1 rand_1^k\left(pbest_i^k - x_i^k\right) + c_2 rand_2^k\left(gbest_i^k - x_i^k\right) \quad (3)$$

$$x_i^k = x_{i-1}^k + v_i^k \quad (4)$$

Where w is the inertia weight, c_1 and c_2 are the acceleration coefficients and, $rand_1^k$ and $rand_2^k$ are random numbers uniformly distributed between [0, 1] for the *k-th* dimension.

The steps of the algorithm are:

i) The process starts with a population of particles with certain position and speed in the space of the problem with dimension *D*. The population is generated randomly.

ii) For each particle is evaluated the objective function (*fitness* function).

iii) The *fitness* of the particle is compared with its *pbest* (the best obtained solution with the particle). If its current value is better, the *pbest* will be equal to the *fitness* value of the particle and the location of the *pbest* will be equal to the current location of the *D*-dimensional space.

iv) The *fitness* of the particle is compared with the best population *fitness*. If the current value is better than the *gbest* (the best reached position by the swarm), then the *gbest* will be updated.

v) Modify the speed and position according to the equations (3) and (4) respectively.

vi) Return to the step (ii) until the stopping criterion is reached (maximum number of generational cycles)

The mutation operator commonly used in the genetic algorithms is introduced in the proposed algorithm trying to emulate the flight turbulence factor, in this work is used an adaptation of the equation to updating the threshold variable of the Threshold Accepting algorithm by Glass and Potts (1996). See more about turbulence factor in Fieldsend and Singh (2002).

The turbulence mechanism consists in permitting big changes during the first iterations, like the threshold accepts the lost of quality for the objective function at the beginning of the process (due to the relaxation of the thresholds). With the advance of the process, turbulence will become more deterministic. The turbulence factor is defined as the modification of a node's value from the tree, trough the mechanism shown in the equation (5).

$$node\ i = node\ i + \left(rand - \frac{1}{2}\right)\cdot\left(\sqrt{1 - \frac{k}{TotalIterations}} + \varepsilon\right) \quad (5)$$

Equation (5) is composed by: the current value of the node *i* from the tree, *rand* is a random number with uniform distribution in the interval $[0,1]$, k is the current iteration, *TotalIterations* is the number of cycles and ε is the minimum percentage to generate a change on the tree, where $\varepsilon = 100/max(L,W)$. In this work is introduced the mutation operator used for the genetic algorithms, named as mutation rate, allowing mutation of the particles in each iteration. The figure 2 show the optimization algorithm proposed, those are used to search the optimum values of the nodes from the slicing of a specific size (level).

Solution Methodology

A scheme of the methodology is presented on the figure 3. The steps that must be done are described and depending of the instance of the problem some steps are omitted. Moreover, the types of searches that are presented on this work are only different on the number of levels of the slicing tree, this permits relate these levels with the cutting constraints by stages, i.e., for the two-staged problems the methodology is executed until step 2, for the three-staged problems until step 3 and finally for the non-staged problems is executed the whole methodology.

Initialize population and parameters
For *l =1* to Total Cycles
For *i =1* to Population Size
If $f(\overline{x_i}) < f(\overline{pbest_i})$ then
$\overline{pbest_i} = \overline{x_i}$
If $f(\overline{x_i}) < f(gbest)$ then
$gbest = \overline{x_i}$
End if
End if
$\overline{v_i} = w\overline{v_{i-1}} + c_1 rand_1 \ (\overline{pbest_i} - \overline{x_i}) + c_2 rand_2 \ (gbest_i - \overline{x_i})$
$\overline{x_i} = \overline{x_{i-1}} + \overline{v_i}$
If *rand* < Mutation Rate
$c\_rand = \lfloor rand \cdot D \rfloor$
$x_i^{c\_rand} = x_i^{c\_rand} + (rand - 1/2) \cdot \left(\sqrt{1 - l/Total\ Cycles} + \varepsilon\right)$
End if
Next *i*
Next l

Figure 2. PSO with Turbulence Factor algorithm (A_{PSO+TF})

Calibration of parameters

The parameter adjustment is very important in order to have good results with the metaheuristic techniques. Different approaches were presented to make the parametrization. In general, there are no exact and efficient methods to make the parameter adjustment of the different metaheuristic techniques, commonly these algorithms are parameterized through the combination of an exhaustive search and a statistical analysis to the quality of the results.

Step 1. Use the optimization algorithm to search over a slicing tree of one level.

Step 2. Use the optimization algorithm to search over a slicing tree of two levels.

Step 3. Use the optimization algorithm to search over a slicing tree of three levels.

Step 4. Deepening, the eight resulting subspaces from the optimal tree in the step 3 are sent to the step 2 in order to make an improvement process to the solution obtained.

Figure 3. Scheme of the solution methodology.

To make the parameterization, different studies suggest: classify the test problems (if they exist) by its complexity (mathematical or computational), choose one representing problem (candidate) from each class, make an adjustment to the parameters for each candidate trough an exhaustive mesh search and finally recombine the obtained parameters for each class picking the best combination.

This process requires a great computational effort because the parameter adjustment through a mesh search represents another optimization process of almost the same complexity to the problem of this study because each parameter belongs to a big range of values.

Zhan *et al.*, (2009) presented a reduced range of values for the parameters of PSO algorithm. Using the presented ranges in Zhan *et al.* (2009) the size of the mesh is considerably reduced. In the Table 1 the values of the parameters are shown.

Parameters	Value
Population Size	100
Number of Cycles - Level 1	50
Number of Cycles - Level 2	75
Number of Cycles - Level 3	100
Number of Deepening Cycles	100
c1 (Individual Knowledge)	2.05
c2(Collective Knowledge)	2.05
w (Inertia)	0.71
Mutation Rate	0.03

Table 1. Parameters and values for the algorithms.

4. Results Analysis

The test systems used in this study were taken from the specialized literature; both approximate and exact methodologies are used in the solution of the mentioned problems. The selected problems are diverse in terms of the mathematical complexity and were specially designed for each type of problem. Different studies have used these test cases to prove the performance of the proposed methodologies.

For the constrained version of the problem, twenty-six test cases were selected for the non-staged problem. Fifteen cases for the weighted version, CHW1 and CHW2 are taken from Christofides and Whitlock (1977); TH1 and TH2 from Tschöke and Holthöfer (1996) and the instances CW1- CW11 are taken from Hifi (1997a). Eleven cases for the unweighted version, CU1- CU11 are taken from Hifi (1997a). Sixty-three test cases were selected for the two-staged problem. Fifty cases for the unweighted version, 2SCUI1- 2SCUI50 are taken from Cui (2007). Thirteen cases for the weighted version, taken from Hifi and Roucairol (2001). Forty test cases were selected for the three-staged problem. Twenty cases for the unweighted version and twenty cases for the weighted version, 3SCUI1- 3SCUI20 are taken from Cui (2008).

For the unconstrained version of the problem, twenty-seven test cases were selected for the non-staged problem, thirteen cases for the unweighted version, GCUT1-GCUT13 take from Cintra *et al.* (2008) and fourteen test cases were selected for the weighted version, (W1-W3 and UW1-UW11) take from Song *et al.* (2010). Thirty test cases were selected for the two-staged version, fifteen cases for the weighted version (UW1-UW11 and UWL1-UWL4) and fifteen cases for the unweighted version (UU1-UU11 and UUL1-UUL4). The database is presented by Hifi (2001) and its available online Hifi (1997b). Twenty-two (UW1-UW11 and UU1-UU11) are the same that were used for the three-staged version.

In order to obtain a sample of the algorithm's performance, it was executed 30 times with each test instance. The algorithm was developed using Delphi 7.0 ® and executed with a Pentium IV ® processor of 3,0 GHz and a RAM memory of 1 GB. With the publication of the

paper, these results will be made available on Internet at http://www.dee.feis.unesp.br/lapsee/arquivos/sistemastestes/resultspsowithturbulencefactor.html.

For the unconstrained non-staged weighted rotated problem, the unconstrained two-staged weighted and unweighted fixed problem, the constrained non-staged and three-staged weighted and unweighted rotated problem and the constrained two-staged unweighted rotated problem, there are no published results to compare. On the other hand, the best known solution (best reported solution in the literature) was chosen for the results of the remaining problems, and in case of having a draw on the results, the solution obtained in the shortest time was chosen.

As observation, for the two-staged weighted problem the answers were compared against Cui (2007), a methodology that generates patterns of two segments which is more restricted, being an unfair comparison. This one was used due to the fact that in the literature does not have another reference to make a comparison. The large-scale instances of the unconstrained three-staged weighted and unweighted, fixed and rotated problem were not compared, due to the fact that Hifi and Roucairol (2001) used in their work but they did not reported results for those.

The table 2 shows the number and average of optima reached by each algorithm, the optima represents the best reported solutions on the specialized literature. For the non-staged problem with item rotation, for the weighted rotated two-staged problem and for the rotated three-staged problem, the comparison is not done, due to that do not exist references to validate the results. The average of optima reached presented in the table 2 shows the robustness of the methodology presented. In general, the A_{PSO+TF} algorithm obtained good results for the different variants of the *k*-staged two-dimensional guillotine knapsack problem (see figure 4).

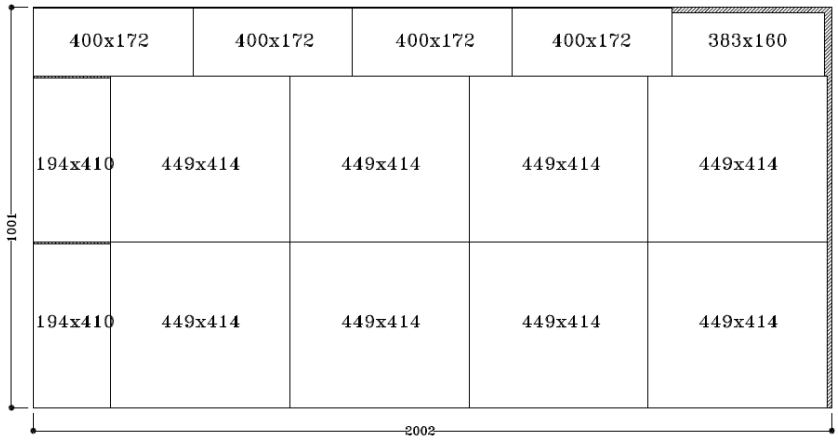

Figure 4. Proposed solution for the 2SCUI1 case that belongs to the constrained two-staged unweighted fixed problem, Cui (2007) reaches an objective value of 98.033%, meanwhile, the presented value on this work reaches 98.93%.

5. Conclusions

The two-dimensional guillotineable knapsack problem has been solved with all its variants: constrained, unconstrained, weighted, unweighted, rotated, fixed, two-staged, three-staged and non-staged. Using an optimization algorithm based on: particle swarm optimization, variable neighborhood search and genetic algorithms, showing effectiveness and robustness in this kind of problems.

The slicing tree encoding from the floorplan design problem was adapted in this work, creating a simple encoding (for the cutting and packing problems) based on the *divide and conquer* strategy. This encoding proposal presented a satisfactory performance for this type of problems because it reduces the search space with a low risk of losing good quality solutions.

Problem	Number of Instances	Number of Optima Reached
FUU_nSTDC	13	13
RUU_nSTDC	13	13
FUW_nSTDC	14	14
RUU_2STDC	15	15
RUW_2STDC	15	15
FUU_3STDC	11	5
RUU_3STDC	11	5
FUW_3STDC	11	11
RUW_3STDC	11	11
FCU_nSTDC	11	6
FCW_nSTDC	15	10
FCW_2STDC	13	13
FCU_3STDC	20	15
FCW_3STDC	20	19
Average of Optima Reached (%)		84.4372
Total Computing Time (sec)		745,280

Table 2. Number and average of optima reached by each algorithm.

The optimization algorithm combines the main features of particle swarm optimization, variable neighborhood search and genetic algorithms. The first one is considered the main algorithm, the second one was used as a limiter of the characteristics of the particles that must be updated, and the last one is the flight turbulence factor represented through the concept of mutation of genetic algorithms, using an adaptation of the equation for updating the threshold variable from threshold accepting algorithm as mutation operator.

The computational times obtained using the proposed methodology, in some cases were better than the ones reported on the specialized literature but due to the differences between the processors architecture and the programming languages used, is not possible to make a final conclusion for the methodologies. In general terms the computing times were reasonable.

References

Alvarez-Valdés, R.; Parajón, A.; Tamarit, J. M.: A tabu search algorithm for large-scale guillotine (un)constrained two-dimensional cutting problems. *Computers & Operations Research*. 29, 925–947, (2002)

Andrews, P. S.: An investigation into mutation operators for particle swarm optimization, *in Proc. IEEE Congr. Evol. Comput.*, Vancouver, BC, Canada, 1044– 1051, (2006)

Angeline, P. J.: Using selection to improve particle swarm optimization, *in Proc. IEEE Congr. Evol. Comput.*, Anchorage, AK, 84–89, (1998)

Beasley, J. E.: Algorithms for unconstrained two-dimensional guillotine cutting. *J. Oper. Res. Soc*., 36, 297–306, (1985)

Berthold, K.: Guillotineable bin packing: A genetic approach. *European J. of Oper. Res.* 84, 645–661, (1995)

Bratton, D.; Kennedy, J.: Defining a standard for particle swarm optimization, *in Proceedings of the IEEE Swarm Intelligence Symposium*, 120-127, (2007)

Carlisle, A.; Dozier, G.: Adapting particle swarm optimization to dynamic environments, *in Proc. Int. Conf. Artif. Intell.*, Las Vegas, NV, 429–434, (2000)

Chen, Y. P.; Peng W. C.; Jian, M. C.: Particle swarm optimization with recombination and dynamic linkage discovery, *IEEE Trans. Syst., Man, Cybern. B*, 37, 1460–1470, (2007)

Christofides, N.; Whitlock, C.: An algorithm for two-dimensional cutting problems. *Operations Research*. 25, 30–44, (1977)

Cintra, G.F.; Miyazawa, F.K.; Wakabayashi, Y.; Xavier E.C.: Algorithms for two-dimensional cutting stock and strip packing problems using dynamic programming and column generation, *European Journal of Operational Research*, 191, 61–85, (2008)

Clerc, M.: The swarm and the queen: Toward a deterministic and adaptive particle swarm optimization, *in Proc. IEEE Congr. Evol. Comput*., 1951–1957, (1999)

Cui, Y.: An exact algorithm for generating homogenous two-segment cutting patterns. *Engineering Optimization*. 39, 365–380, (2007)

Cui, Y.: Heuristic and exact algorithms for generating homogenous constrained three-staged cutting patterns. *Computers & Operations Research*. 35, 212–225, (2008)

Fayard, D.; Hifi, M.; Zissimopoulos, V.: An efficient approach for large-scale two-dimensional guillotine cutting stock problems. *J. of the Operational Research Society*. 49, 1270–1277, (1998)

Fieldsend, J. E. and Singh, S.: A multi-objective algorithm based upon particle swarm optimization, an efficient data structure and turbulence. in *Proc. 2002 U.K. Workshop on Computational Intelligence*, Birmingham, U.K., 37–44, (2002)

Gilmore, P. C.; Gomory, R. E.: Multistage cutting problems of two and more dimensions. *Operations Research*, 13, 94-120, (1965)

Gilmore, P. C.; Gomory, R. E.: The theory and computation of knapsack functions. *Oper. Res.*, 14, 1045-1074, (1966)

Glass C. A.; Potts, C. N.: A comparison of local search methods for flow shop scheduling, *Annals of Operations Research*, 63, 489-509, (1996)

Herz, J. C.: A recursive computing procedure for two-dimensional stock cutting. *I.B.M. Ji Res. Dev.*, 16, 462-469, (1972)

Hifi, M.: An improvement of Viswanathan and Bagchis exact algorithm for constrained two-dimensional cutting stock. *Computers & Operations Research*. 24, 727–736, (1997a)

Hifi, M.: *Problem instances for the 2D Cutting/Packing Problems*, [on line], <ftp://cermsem.univ-paris1.fr/pub/CERMSEM/hifi/2Dcutting/>. (1997b).

Hifi, M.: Exact algorithms for large-scale unconstrained two and three staged cutting problems. *Computational Optimization and Applications*. 18, 63–88, (2001)

Hifi, M.; Roucairol, C.: Approximate and exact algorithms for constrained (un)weighted two-dimensional two-staged cutting stock problems. *J. of Comb. Optimization*. 5, 465–494, (2001)

Hifi, M.; Zissimopoulos, V.: A recursive exact algorithm for weighted two-dimensional cutting. *European J. Oper. Res.*, 91, 553–564, (1996)

Hu, X.; Eberhart, R. C.: Adaptive particle swarm optimization: Detection and response to dynamic systems, *in Proc. IEEE Congr. Evol. Comput.*, Honolulu, HI, 1666–1670, (2002)

Kennedy J.; Mendes, R.: Population structure and particle swarm performance, Proceedings of the *Congress on Evolutionary Computation (CEC 2002)*, Honolulu, HI, USA, 1671–1676, (2002)

Liang, J. J.; Suganthan, P. N.: Dynamic multi-swarm particle swarm optimizer with local search, *in Proc. IEEE Congr. Evol. Comput.*, 522–528, (2005)

Mendes, R.; Kennedy, J.; Neves, J.: Avoiding the pitfalls of local optima: how topologies can save the day, *in Proc. Conf. Intelligent Systems Application to Power Systems*, (2003).

Morabito, M.; Pureza, V.: Generation of constrained two-dimensional guillotine cutting patterns via dynamic programming and and/or-graph search. *Produção*. 17, 033–051, (2007)

Song, X.; Chu, C.B.; Lewis, R.; Nie, Y.Y.; Thompson J.: A worst case analysis of a dynamic programming-based heuristic algorithm for 2D unconstrained guillotine cutting, *European Journal of Operational Research*, 202, 368–378, (2010)

Tschöke, S.; Holthöfer, N.: A new parallel approach to the constrained two-dimensional cutting stock problem. Technical Report, University of Padeborn, D.C.S. 33095 Padeborn. Germany, (1996)

Viswanathan, K.V.; Bagchi, A.: Best-first search methods for constrained two-dimensional cutting stock problems. *Operations Research*. 41, 768–776, (1993)

Wang, P. Y.: Two algorithms for constrained two-dimensional cutting stock problems. *Operations Research*. 32, 573–586 (1983)

Wong, D. F.; Leong, H. W.; LIU, C. L.: *Simulated Annealing for VLSI Design*. Kluwer Academic Publishers, (1988)

Xie, X.; Zhang, W.; Yang, Z.: Adaptive particle swarm optimization on individual level, *in Proc. Int. Conf. Signal Process.*, 1215–1218, (2002)

Young-Gun, G.; Kang, M.-K.: A new upper bound for unconstrained two-dimensional cutting and packing. *J. Oper. Res. Soc.*, 53, 587–591, (2002)

Young-Gun, G.; Kang, M.-K.; Seong, J.: A best-first branch and bound algorithm for unconstrained two-dimensional cutting problems. *Oper. Res. Letters*, 31, 301–307, (2003)

Zhan, Z.; Zhang, J.; Li, Y.; Chung, H.S.H.: Adaptive particle swarm optimization. *IEEE Transactions on System, Man and Cybernetics – B*, 39, 1362–1381, (2009)