On solving the double loading problem using a modified particle swarm optimization

Takwa Tlili, Saoussen Krichen

PII: S0304-3975(15)00478-8
DOI: http://dx.doi.org/10.1016/j.tcs.2015.05.037
Reference: TCS 10236

To appear in: *Theoretical Computer Science*

Received date: 16 September 2012
Revised date: 19 February 2015
Accepted date: 13 May 2015

Please cite this article in press as: T. Tlili, S. Krichen, On solving the double loading problem using a modified particle swarm optimization, *Theoret. Comput. Sci.* (2015), http://dx.doi.org/10.1016/j.tcs.2015.05.037

On solving the double loading problem using a modified particle swarm optimization

Abstract

We consider in this paper a double loading problem (DLP), an NP-hard optimization problem of extreme economic relevance in industrial areas. The problem consists in loading items into bins, then stowing bins in a set of compartments. The main objective is to minimize the number of used bins. We state the mathematical model as well as a modified binary particle swarm optimization with FFD initialization that outperforms state-of-the-art approaches carried out on a large testbed instances.

Keywords: Double loading problem, Compartmentalized Knapsack Problem, Particle swarm optimization, Metaheuristics.

1. Introduction

In the manufacturing and distribution industries the efficient use of transportation devices is of extreme relevance. A high utilization of the transportation capacities offers a significant cost saving as well as a protection of natural resources. During the last decades packing problems have received a great deal of attention due to the wide number of related practical applications as the bin ship loading (Wei-ying et al., 2005), the plane cargo management (De Queiroz & Miyazawa, 2013) and the transportation planning (Zachariadis et al., 2013). The modeling of different practical problems of the optimal employment of transportation devices brings to life numerous variants of packing problems. The compartmentalized Knapsack Problem (CKP), a packing problem version introduced by Hoto et al. (2007), splits the packing into two main steps: *clustering* and *compartmentalized packing* of items. In spite of the importance of the problem, few studies have dealt with it. Malthouse et al. (2012) applied the CKP in the marketing area to select media vehicles for marketing campaigns then, to determine the depth of purchase from each vehicle. Leão et al. (2011)

and Marques & Arenales (2007) proposed different mathematical formulations to handle the constrained CKP.

Numerous metaheuristics were adopted to solve such relevant class of NP-hard problems. More recently, swarm intelligence-based methods have attract more consideration, e.g., Particle Swarm Optimization (PSO) (Hu et al., 2012), Artificial Bee Colony (ABC) (Samanta & Chakraborty, 2011)and Ant Colony Optimization (ACO) (Rada-Vilela et al., 2013). Due to the simplicity and robustness of the PSO, it has increasingly gained popularity.

Specifically, the Particle swarm optimization (PSO) is a bio-inspired method developed by Kennedy and Eberhart (1995) to handle combinatorial optimization problems. It is a population-based evolutionary algorithm simulating the social behavior of bird flocking. A lot of emphasis has been laid on enhancing the conventional PSO to yield a computationally efficient algorithm. In view of the shortcomings of the conventional PSO algorithm, numerous variants have been proposed. Luan et al. (2012) introduced the Robust Particle swarm Optimization (RPSO), in which an optimal solution tolerates the small perturbation on solving robust optimization problems. Chen et al. (2010) proposed a Set-Based Particle Swarm Optimization (S-PSO) that defines the position and velocity using the concept of set and possibility. Modiri & Kiasaleh (2011) suggested a modification of the velocity by either omitting the personal best coefficient or changing the initial velocity. This PSO variant were shown to give better results for benchmark mathematical optimizations. Orthogonal Particle Swarm Optimization (OPSO) has been proposed by Ho et al. (2008) in which an intelligent move mechanism is used to adjust a velocity for each particle.

In this paper, we investigate a variant of the CKP named the double loading problem (DLP). Theoretically spoken, in the DLP, *clustering* step corresponds to regrouping and packing items into bins while respecting the bin's capacity. The *compartmentalized packing* step refers to packing used bins into capacity limited compartments. In practice, the DLP, can also be viewed as a two-step problem: the bin packing problem and the knapsack problem.

We develop, as a solution approach, a modified variant of the PSO proposed by Modiri & Kiasaleh (2011) based on changing the velocity for a fast convergence to best solutions. Since no DLP benchmark is available, first, we test the performance of the proposed method on the first step (bin packing). Second, we generate some classes, inspired by the bin packing instances, to test the entire

DLP.

The present paper is organized as follows: section 2 describes in details the DLP. In section 3, a mathematical model is stated. Section 4 is a basic version statement of the PSO metaheuristic. The adapted new approach MBPSO is denoted in section 5, we also present the solution encoding and the evaluation standards in this same section. Experimental investigations are drawn in section 6 where a set of tests are carried out to measure the efficiency of MBPSO method and a dataset of Benchmark instances are performed to carry on a comparison between our results and some other heuristics from the literature.

2. Problem statement

We are specifically concerned with the DLP, where the number of used bins is to be minimized and system constraints are to be satisfied. The DLP is also considered as a two-step optimization model expressed by a combination of a bin packing subproblem and a knapsack subproblem. The bin packing concerns the step of items loading into bins and the knapsack consists on stowing a maximum number of used bins in compartments of a mean of transport or a pallet. Figure 1 explains clearly the DLP showing the theoretical and the practical sides of the problem.

The main objective of the DLP is to minimize the number of used bins while respecting the bin's capacity and the compartment's capacity.

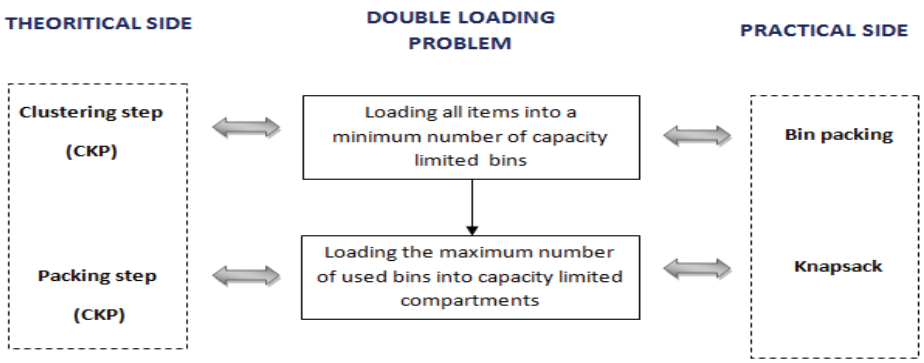

Figure 1: DLP definition

3. Mathematical formulation

3.1. Sets and notation

N	The total number of items to be packed
M	The total number of bins available
P	The total number of compartments
w_i	The weight of item i
v_i	The volume of item i
C_j	The maximum capacity of bin j
V_j	The volume of bin j
T_k	The maximum capacity of compartment k

3.2. Binary decision variables

$$X_{ij} = \begin{cases} 1 & \text{if the item } i \text{ is placed in bin } j \\ 0 & \text{otherwise} \end{cases}$$

$$Y_{jk} = \begin{cases} 1 & \text{if the bin } j \text{ is stowed in compartment } k \\ 0 & \text{otherwise} \end{cases}$$

$$n_j = \begin{cases} 1 & \text{if the bin } j \text{ is used} \\ 0 & \text{otherwise} \end{cases}$$

3.3. Model

Given a number M of different bins with volume V_j and capacity C_j $(j = 1, \ldots, M)$. Let N be a set of items $(i = 1, \ldots, N)$. Each item i has a volume v_i and a weight w_i.

This problem is about loading all items into bins, then allocating the bins into P compartments $(k = 1, \ldots, P)$ of capacity T_k. Our main objective is minimizing the number of used bins.

We formulate the DLP by the following mathematical model:

(1)	$Min\ Z(X) = \sum_{j=1}^{M}\ n_j$	The objective (1) states that the number of used bins is to be minimized.
(2)	$\sum_{j=1}^{M} X_{ij} = 1; \forall i$	Constraints (2) ensure that every item is packed in exactly one bin.
(3)	$\sum_{k=1}^{P} Y_{jk} = n_j; \forall j$	The set of constraints (3) states that every used bin should be loaded in exactly one compartment.
(4)	$\sum_{i=1}^{N} w_i X_{ij} \leq C_j n_j; \forall j;$	Constraints (4) impose that the capacity of bins should be respected.
(5)	$\sum_{j=1}^{M} \sum_{i=1}^{N} w_i X_{ij} \leq T_k; \forall k$	The set of constraints (5) guarantee the respect of the compartments' capacities.
(6)	$\sum_{i=1}^{N} v_i X_{ij} \leq V_j n_j; \forall j;$	Constraints (6) impose that the maximum volume of bins should be respected.

3.4. An illustrative example

The mathematical model was tested using IBM ILOG CPLEX 12.1 (a Branch & Cut algorithm) running on a 2 GHz Intel Core Duo processor with 3 GB RAM under Windows XP.

Let us consider an example of $N = 10$ items to be packed into $M = 3$ bins and $P = 2$ compartments. The parameters settings are tabulated in table 4.

Number	Items		bins		Compartments
	w_i	v_i	C_j	V_j	T_k
1	5	8	50	60	70
2	20	15	100	100	180
3	15	10	150	80	-
4	10	15	-	-	-
5	15	15	-	-	-
6	7	9	-	-	-
7	10	15	-	-	-
8	20	20	-	-	-
9	50	25	-	-	-
10	25	20	-	-	-

Table 1: Parameters' settings of the example

After an average of 10 Branch & Cut iterations, the feasible solution is $Z(X) = 2$ (illustrated in figure 2).

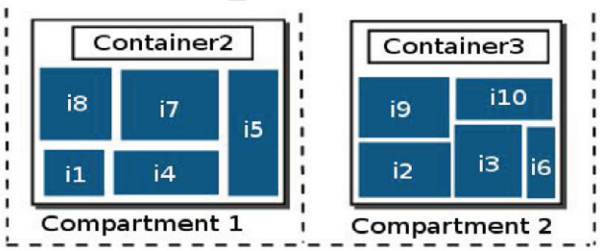

Figure 2: A feasible solution for the DLP solved by CPLEX

4. Solution solving methodology

As the DLP is NP-hard, we propose to solve it using a metaheuristic approach. Indeed, based on the fact that the MBPSO is efficient for generating high quality solutions, we propose to adapt it for the DLP. The MBPSO features will be state, at first, to highlight the ability of each operator wile constructing solutions and moving from one swarm to another. All such information is provided in the subsequent sections.

In this section, the conventional particle swarm optimization (PSO), the binary PSO as well as the modified PSO are discussed for solving the DLP.

4.1. Particle swarm optimization approach

In nature, a swarm of birds looking for food, starts by flying around randomly with no particular destination until one of birds found the best source. Analogically, PSO initializes a population of random particles, everyone is considered as a potential solution (Perez and Behdinan, 2007). Firstly, particles fly spontaneously through the problem space at a random velocities. Next, velocities are updated based on the best previous particle's experience and the best previous group's experience, i.e. the behavior inside a population is a compromise between individual and collective memory. This can be described by a swarm intelligence system in which the share of information among individuals is the basic idea.

The PSO script is described as follows.

Given a d-dimensional search space and a swarm of S particles ($p = 1, \ldots, S$).

- To each particle p in generation t corresponds a position-vector $X_p^t = (x_{p1}^t, x_{p2}^t, \ldots, x_{pd}^t)$ and a velocity-vector $V_p^t = (v_{p1}^t, v_{p2}^t, \ldots, v_{pd}^t)$, where x_{pn}^t represents the location and v_{pn}^t is the flying velocity of particle p in generation t in the nth dimension of the search space ($n = 1, \ldots, d$).
- Particles memorize every reached position and save the one with the best fitness. This individual best position is denoted by $B_p^t = (b_{p1}^t, b_{p2}^t, \ldots, b_{pd}^t)$.
- Particles record the whole best position's fitness until generation t. This global best position is referred to as $G^t = (g_1^t, g_2^t, \ldots, g_d^t)$.

At each generation t a particle p takes one of those decisions:

- Keeping its strategy of search.
- Coming back to its previous individual best position B_p^{t-1}.
- Coming back to the global best position G^{t-1}.

The new velocity and position are updated, respectively, by the following formulas:

$$v_{pn}^{t} = iw\ v_{pn}^{t-1} + \lambda_1\ (b_{pn}^{t-1} - x_{pn}^{t-1}) + \lambda_2\ (g_{n}^{t-1} - x_{pn}^{t-1}) \quad (1)$$

$$x_{pn}^{t} = x_{pn}^{t-1} + v_{pn}^{t} \quad (2)$$

$$p = 1, \ldots, S\ ;\ n = 1, \ldots, d\ ;\ t = 1, \ldots, T \quad (3)$$

where:

$\lambda_1 = c_1 \times r_1$ and $\lambda_2 = c_2 \times r_2$

- iw : The inertia weight that controls the influence of the precedent velocity on the current velocity
- c_1 : The first acceleration coefficient given to the attraction to the previous best location of the current particle
- c_2 : The second acceleration coefficient given to the attraction to the previous best location of the particle neighborhood
- r_1, r_2: Two random variables uniformly distributed in [0, 1], i.e. $r_1, r_2 \sim U(0, 1)$
- T_{max} : The maximum number of generations prefixed by the decision maker

4.2. Binary particle swarm optimization

The original variant of PSO is continuous, it deals with real-valued optimization problems. However, for some optimization problems with binary decision variables, it is more suitable to use a binary version of PSO (BPSO) proposed by Kennedy and Eberhart (1997).

In a discrete search space:

- Particles' moves are restricted to 0 or 1 on each dimension, i.e. particle's position is a binary variable $x_{pn}^{t} \in \{0, 1\}$.
- Each particle's velocity is considered as a probability to change a position from 0 to 1 and vice versa. For this reason, a mapping needs to be defined in order to switch velocities from real-valued form to the probability form ($v_{pn}^{t} \in [0, 1]$). This normalization is done due to a sigmoid function (4).

$$Sig(v_{pn}^{t}) = \frac{1}{1 + e^{-v_{pn}^{t}}} \quad (4)$$

The equation of updating positions (2) is then replaced by the following formula:

$$x_{pn}^{t} = \begin{cases} 1 & \text{if } Sig(v_{pn}^{t}) \geq rnd() \\ 0 & \text{otherwise} \end{cases} \tag{5}$$

Where rnd() is a uniform random number in the range [0,1].

4.3. Modified binary particle swarm optimization

The Modified Binary Particle Swarm Optimization (MBPSO), extended from the BPSO, is based on a modified velocity calculation. In MBPSO, the individual best positions B_p^t and the global best position G^t are updated as in continuous version. Whereas, there is a modification in the velocity function propose by Modiri & Kiasaleh (2011) and the formula of position's update.
The idea of the velocity function modification is to omit personal best coefficient. The proposed velocity function (6) is proved to give better results for solving benchmark mathematical optimizations (Modiri & Kiasaleh, 2011).
The particle's velocity is updated as follows:

$$v_{pn}^{t} = iw\ v_{pn}^{t-1} + \lambda_2\ (g_n^{t-1} - x_{pn}^{t-1}) \tag{6}$$

The MBPSO flowchart description is depicted in figure 3 illustrating the DLP solving procedure.

4.3.1. Initial population

To accelerate the convergence of the algorithm, we generate an initial population using the First Fit Decreasing (FFD) heuristic. The details of the FFD algorithm is depicted in Algorithm 1.

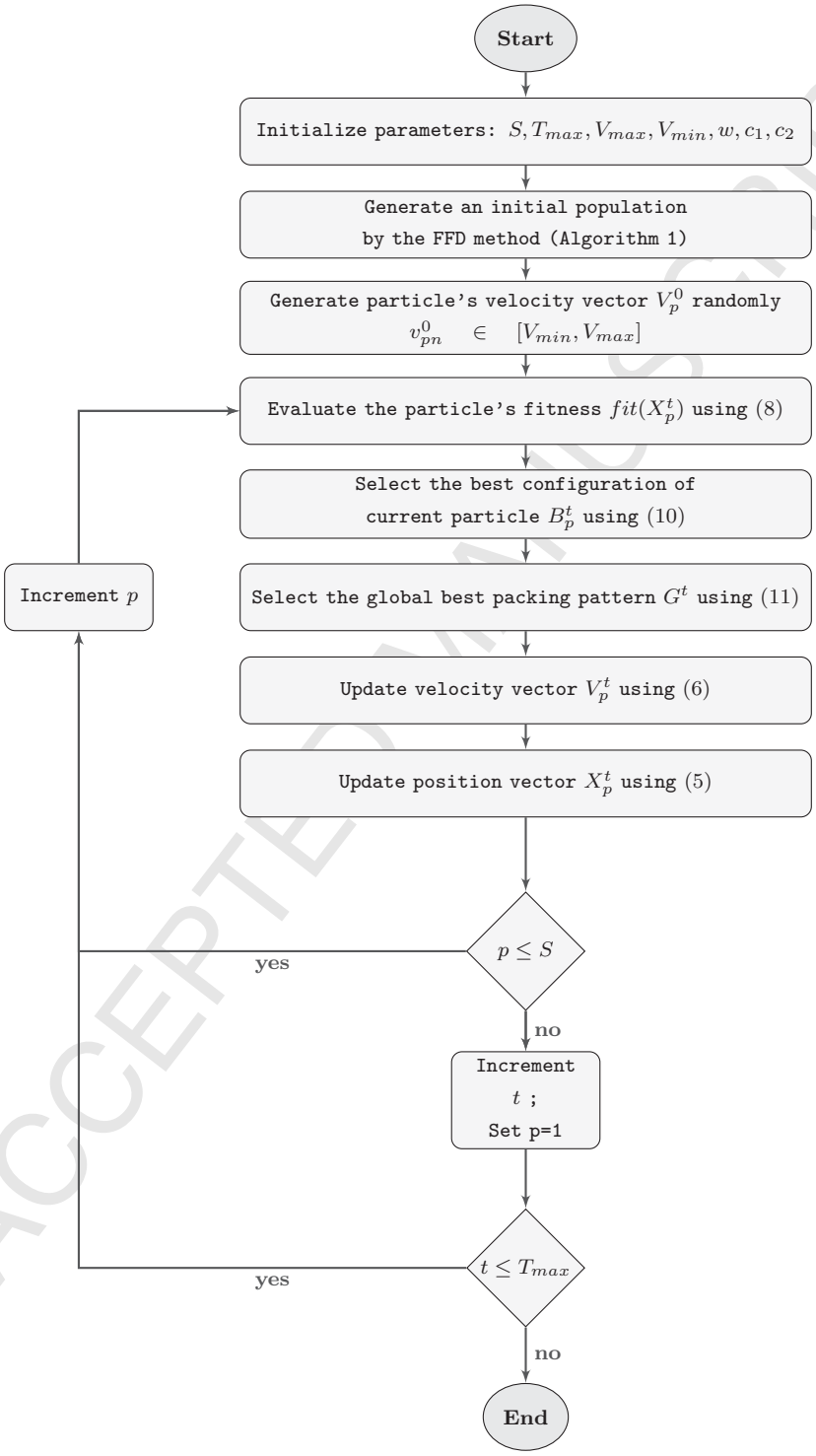

Figure 3: The flowchart of the MBPSO algorithm

Algorithm 1 FFD algorithm for DLP

```
Sort items in decreasing order according to their weights;
while (Items list is not empty) do
  Load the item in the first suitable bin that satisfy all constraints;
  if (The item cannot fit into any already used bin) then
    Open new bin;
    Remove placed item from item list;-+
  end if
end while
Sort used bins in decreasing order by their weights ;
while (Compartments are not full) do
  Pack bin in the first feasible compartment;
  if (The bin cannot be packed into any compartment) then
    Use new compartment;
    Remove placed bins from the bin list;
  end if
end while
```

4.3.2. Encoding of a particle

Let us consider a set of N items. Each item i ($i = 1, \ldots, N$) should be packed into one of M bins. Then, each full bin j ($j = 1, \ldots, M$) is loaded in a compartment k ($k = 1, \ldots, P$).

Each particle is encoded in (N, M, P) dimensions as shown in figure 4.

$$X_p^t = \begin{pmatrix} X_{111} & X_{121} & \ldots & X_{1j1} & \ldots & X_{1M1} \\ X_{211} & X_{221} & \ldots & X_{2j1} & \ldots & X_{2M1} \\ \vdots & \vdots & \ldots & \vdots & \ldots & \vdots \\ X_{i11} & X_{i21} & \ldots & X_{ij1} & \ldots & X_{iM1} \\ \vdots & \vdots & \ldots & \vdots & \ldots & \vdots \\ X_{i1k} & X_{i2k} & \ldots & X_{ijk} & \ldots & X_{iMk} \\ \vdots & \vdots & \ldots & \vdots & \ldots & \vdots \\ X_{N1P} & X_{N2P} & \ldots & X_{NjP} & \ldots & X_{NMP} \end{pmatrix}$$

Figure 4: Particle's encoding

It is a binary particle in which a variable X_{ijk} takes 0 if item i is packed neither in bin j nor in compartment k, otherwise it is equal to 1.

Example

Figure 5 is the encoding of the CPLEX example presented in figure 2.

$$X = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 1 \end{pmatrix}$$

Figure 5: Particle's encoding for the example

4.3.3. Evaluation of a particle

In order to simulate the natural survival process of the fittest in biological world, the fitness evaluation function assigns to each particle of the swarm a value reflecting their relative superiority or inferiority (Iyer and Saxena, 2004). This function is considered not only as a quality measure of a given solution but also an indication of particle's reconciliation from the best solution.

As shown in the mathematical formulation, it is about treating a constrained optimization problem. The most common approach in solving this type of problems is the use of a penalty function, which adds a penalty to the objective function in order to decrease the quality of an infeasible areas of the search space (Abadi and Jalili, 2008). There exist three principal forms of penalties that differ in some details namely static penalty (Homaifar et al., 1994), dynamic penalty (Joines and Houck, 1994) and death penalty (Michalewicz, 1995). We employ the dynamic-additive form seen that the particle's evaluation occurs dynamically at each iteration.

To handle the DLP, we evaluate each particle's fitness $fit(X)$ by referring to the objective function $F(X)$ and also the dynamic penalty factor $P(X)$.

Where:

$$fit(X) = \begin{cases} F(X) & \text{if } X \text{ is feasible} \\ F(X) + P(X) & \text{elsewhere} \end{cases} \quad (7)$$

If no constraint violation occurs, the penalty term $P(X)$ will be null and the evaluation will be done by the objective function. Snyman et al. (1994) stated the formulation (8) for dynamic fitness function to evaluate individuals at each generation t:

$$fit(X) = F(X) \; + \; (C \times t)^{\alpha} \quad \sum_{q=1}^{Q} f_j^{\beta}(X) \quad (8)$$

Where: C, α, and β are the penalty's parameters. According to the same reference $C = 0.5, \alpha = \beta = 2$ and the function $f_j(X)$ $(j = 1, \ldots, Q)$ measures the violation of the j-th constraint.

$$f_j(X) = \begin{cases} Max\{0, g_j(X)\} & \text{if } 1 \leq j \leq q \\ \mid h_j(X) \mid & \text{if } q+1 \leq j \leq Q \end{cases} \quad (9)$$

With:
$g_j(X)$ and $h_j(X)$ are sets of constraints taking the following forms:

$$g_j(X) \leq 0 \qquad \text{and} \qquad h_j(X) = 0$$

The formula of the individual best positions B_p^t:

$$B_p^t = \begin{cases} X_p^t; & \text{if } t = 0 \\ X_p^{t^*} / t^* = Min(X_p^t); & \text{if } t = 1, \ldots, T_{max} \end{cases} \quad (10)$$

The formula of the global best positions G^t:

$$G^t \quad \leftarrow \quad \{B_{p^*}^t \quad / p^* = Min(B_p^t)\}; \quad \forall p \in S \quad (11)$$

5. Experimental design

The MBPSO heuristic is implemented in Java language (NetBeans IDE 6.9) based on a 2 GHz Intel Core Duo processor with 3 GO RAM under Linux.
Since no test instances were available for the DLP, we generate a dataset to demonstrate the abilities of this approach and we report the empirical results. Besides, we perform the MBPSO over 1210 benchmark instances on the bin

packing step of the DLP. An experimental analysis of results are occurred in addition to a comparison with state-of-the-art solution approaches.
To show the effectiveness of the MBPSO, we perform a benchmark test developed by Scholl et al. (1997) and available on line at
`http://www.wiwi.uni-jena.de/Entscheidung/binpp/index.htm`.
The benchmark counts 1210 instances distributed into three sets:

- **S1:** 720 instances divided into 36 classes of 20 instances. Those classes differ by the weight of items $w_i \in [1, 100], [20, 100] or [30, 100]$, the bin capacity $C \in \{100, 120, 150\}$ and the number of items $N \in \{50, 100, 200, 500\}$.
- **S2:** 480 instances consists in 48 classes of 10 instances characterized by a bin capacity $C = 1000$, a number of items $N \in \{50, 100, 200, 500\}$ and an average weight of items $a_i \in \{\frac{C}{3}, \frac{C}{5}, \frac{C}{7}, \frac{C}{9}\}$.
- **S3:** 10 hard instances with $C = 100000$, $N = 200$ and items are drawn from a uniform distribution on $[20000, 35000]$. This set is the most difficult one.

5.1. Applied metrics

We perform the same metrics used by Scholl et al. (1997) to compare SMBPSO results to other heuristics taken from literature. Let z be the found solution and z^* the best-known solution.

- Hits: The percentage of the number of best known solutions reached by the MBPSO.
- Absolute Deviation (AD): The deviation from the best-known solution;

$$AD = z - z^* \tag{12}$$

- Relative Deviation (RD): It measures the amount of variation between z and z^*. It is expressed as a percentage;

$$RD = \frac{z - z^*}{z^*} \tag{13}$$

- AVG: The average absolute (relative) deviation from optimality.
- MAX: The maximum absolute (relative) deviation from optimality.

5.2. Empirical results

In table 2 we report the MBPSO results on 36 classes from S1. When observing table 2, we can point out that for 33 classes we obtain 100% of optimality. For the rest of classes, we reach 95% of best known solutions. Results of failed instances are detailed in table **??**.

Table 3 illustrates the computational results of testing the benchmark set S2 to 48 classes. MBPSO reaches the 100% of best-known solutions in 41 classes. For the rest of classes, hits are of 90%, AVG-AD are in the range $[0.1, 0.5]$, MAX-AD $\in [1, 5]$, AVG-RD $\in]0, 0.06]$ and MAX-RD $\in]0, 0.073]$.

In terms of optimality percentage, the results of S1 (91.66%)) outperforms those of S2 (85.41%).

5.3. Comparison with state-of-the art approaches

In order to show the effectiveness of MBPSO results to data sets, the quality of the solutions is compared based on the average number of bins and the number of optimal instances.

Table 4 compares the best number of bins averaged over the instances of each test problem against previous solution approaches proposed in Kim & Wy (2010). We point out that our algorithm outperforms the other approaches in all sets. The general average number of bins (82.67) is the nearest value from the best average.

Table 5 shows the results of set 3 considered as the hardest class of scholl benchmark. Our algorithm obtains the optimal solution in 80% of the instances. The MBPSO average number of bins outperforms the results of FFD, QICSABP, GGA1 and GGA2. The proposed algorithm found similar average result as GPSO developed by Kashan et al., (2013).

Inst. class	Kim & Wy (2010)				This paper	Best average
	NFD-L2F	FFD-L2F	BFD-L2F	BBB-FFD-L2F	MBPSO	
S1	109.3	110.6	110.6	109	**108.87**	108.85
S2	42.6	42.7	42.7	42.6	**42.21**	42.17
S3	56.5	56.6	56.6	56.5	**56.4**	56.2
Average	83.09	83.91	83.91	82.91	**82.67**	82.65

Table 4: Comparison based on the average number of bins

Instance Class	Hits (%)	AD		RD (%)		Instance Class	Hits (%)	AD		RD (%)	
		AVG	MAX	AVG	MAX			AVG	MAX	AVG	MAX
$N_1C_1W_1$	100	0	0	0	0	$N_3C_1W_1$	100	0	0	0	0
$N_1C_1W_2$	100	0	0	0	0	$N_3C_1W_2$	100	0	0	0	0
$N_1C_1W_4$	100	0	0	0	0	$N_3C_1W_4$	100	0	0	0	0
$N_1C_2W_1$	100	0	0	0	0	$N_3C_2W_1$	100	0	0	0	0
$N_1C_2W_2$	100	0	0	0	0	$N_3C_2W_2$	100	0	0	0	0
$N_1C_2W_4$	100	0	0	0	0	$N_3C_2W_4$	100	0	0	0	0
$N_1C_3W_1$	100	0	0	0	0	$N_3C_3W_1$	100	0	0	0	0
$N_1C_3W_2$	100	0	0	0	0	$N_3C_3W_2$	95	0.1	1	0.0052	0.026
$N_1C_3W_4$	100	0	0	0	0	$N_3C_3W_4$	100	0	0	0	0
$N_2C_1W_1$	100	0	0	0	0	$N_4C_1W_1$	100	0	0	0	0
$N_2C_1W_2$	100	0	0	0	0	$N_4C_1W_2$	100	0	0	0	0
$N_2C_1W_4$	100	0	0	0	0	$N_4C_1W_4$	100	0	0	0	0
$N_2C_2W_1$	100	0	0	0	0	$N_4C_2W_1$	100	0	0	0	0
$N_2C_2W_2$	100	0	0	0	0	$N_4C_2W_2$	95	0.05	1	0.00015	0.003
$N_2C_2W_4$	100	0	0	0	0	$N_4C_2W_4$	100	0	0	0	0
$N_2C_3W_1$	100	0	0	0	0	$N_4C_3W_1$	100	0	0	0	0
$N_2C_3W_2$	100	0	0	0	0	$N_4C_3W_2$	100	0	0	0	0
$N_2C_3W_4$	100	0	0	0	0	$N_4C_3W_4$	95	0.25	5	0.028	0.023

Table 2: Computational results for set 1

ACCEPTED MANUSCRIPT

Instance Class	Hits (%)	AD		RD (%)		Instance Class	Hits (%)	AD		RD (%)	
		AVG	MAX	AVG	MAX			AVG	MAX	AVG	MAX
$N_1W_1B_1$	100	0	0	0	0	$N_3W_1B_1$	100	0	0	0	0
$N_1W_1B_2$	100	0	0	0	0	$N_3W_1B_2$	90	0.3	3	0.04	0.046
$N_1W_1B_3$	100	0	0	0	0	$N_3W_1B_3$	100	0	0	0	0
$N_1W_2B_1$	100	0	0	0	0	$N_3W_2B_1$	90	0.3	3	0.06	0.073
$N_1W_2B_2$	100	0	0	0	0	$N_3W_2B_2$	90	0.2	2	0.01	0.048
$N_1W_2B_3$	100	0	0	0	0	$N_3W_2B_3$	100	0	0	0	0
$N_1W_3B_1$	100	0	0	0	0	$N_3W_3B_1$	90	0.1	1	0.00034	0.034
$N_1W_3B_2$	100	0	0	0	0	$N_3W_3B_2$	100	0	0	0	0
$N_1W_3B_3$	100	0	0	0	0	$N_3W_3B_3$	100	0	0	0	0
$N_1W_4B_1$	100	0	0	0	0	$N_3W_4B_1$	100	0	0	0	0
$N_1W_4B_2$	100	0	0	0	0	$N_3W_4B_2$	100	0	0	0	0
$N_1W_4B_3$	100	0	0	0	0	$N_3W_4B_3$	100	0	0	0	0
$N_2W_1B_1$	100	0	0	0	0	$N_4W_1B_1$	90	0.5	5	0.075	0.03
$N_2W_1B_2$	100	0	0	0	0	$N_4W_1B_2$	100	0	0	0	0
$N_2W_1B_3$	100	0	0	0	0	$N_4W_1B_3$	100	0	0	0	0
$N_2W_2B_1$	100	0	0	0	0	$N_4W_2B_1$	90	0.3	3	0.02	0.029
$N_2W_2B_2$	100	0	0	0	0	$N_4W_2B_2$	90	0.2	2	0.0076	0.019
$N_2W_2B_3$	100	0	0	0	0	$N_4W_2B_3$	100	0	0	0	0
$N_2W_3B_1$	100	0	0	0	0	$N_4W_3B_1$	100	0	0	0	
$N_2W_3B_2$	100	0	0	0	0	$N_4W_3B_2$	100	0	0	0	
$N_2W_3B_3$	100	0	0	0	0	$N_4W_3B_3$	100	0	0	0	0
$N_2W_4B_1$	100	0	0	0	0	$N_4W_4B_1$	100	0	0	0	0
$N_2W_4B_2$	100	0	0	0	0	$N_4W_4B_2$	100	0	0	0	0
$N_2W_4B_3$	100	0	0	0	0	$N_4W_4B_3$	100	0	0	0	0

Table 3: Computational results for set 2

ACCEPTED MANUSCRIPT

Instance	Layeb & Boussalia (2012)		Kashan et al. (2013)			This paper	Best Bins
	FFD	QICSABP	GPSO	GGA1	GGA2	MBPSO	
HARD0	59	59	56	56	56	56	56
HARD1	60	60	57	57	57	57	57
HARD2	60	60	57	57	57	57	56
HARD3	59	59	56	56	57	55	55
HARD4	60	60	57	58	58	57	57
HARD5	59	59	56	57	57	56	56
HARD6	60	59	57	57	58	58	57
HARD7	59	59	55	55	56	55	55
HARD8	60	59	57	57	58	57	57
HARD9	60	59	56	57	57	56	56
Average	59.6	59.3	**56.4**	56.7	57.1	**56.4**	56.2

Table 5: Comparison of hard instances of set 3

Table 6 shows that MBPSO reach the best known solutions for 717 instances while other approaches found worst values (615, 672, 692 and 712). For sets 2 and 3, the hybrid algorithm proposed by Yesil et al. (2013) outperforms our approach in two instances for each set. Whereas the MBPSO shows best results compared to the rest of methods.
Globally, the MBPSO outperforms the four approaches by solving 1198 instances over a set of 1210 instances which is the best value shown in table 6.

Inst. class	Jiang et al. (2011)	Quiroz et al. (2013)	Yesil et al. (2013)	Abidi et al. (2013)	This paper
S1 (720)	672	692	712	615	**717**
S2 (480)	343	450	**475**	315	473
S3 (10)	4	8	**10**	0	8
All (1210)	1019	1150	1197	930	**1198**

Table 6: Comparison based on the number of solved instances to optimality

6. Conclusion

We studied in this paper a DLP that involves an objective that minimizes the number of used bins for loading the whole set of items while respecting structural constraints which should be respected such as the capacity weight and volume of bins. It was shown the packing problems are NP-Hard problem

therefore we started by stating a mathematical formulation and we resorted to a heuristic approach to solve the problem in a reasonable computational time. We developed a MBPSO approach for the DLP and experimented it on a large variety of instances including some benchmarks. The empirical results show clearly that, compared with the state-of-the-art approaches outperforms in almost all cases.

References

Abadi, M., & Jalili, S. (2008). Using Binary Particle Swarm Optimization for Minimization Analysis of Large-Scale Network Attack Graphs. Scientia Iranica, 15(6), 605-619.

Abidi, S., Krichen, S., Alba, E., & Molina, J. M. (2013, April). Improvement heuristic for solving the one-dimensional bin-packing problem. In Modeling, Simulation and Applied Optimization (ICMSAO), 5th International Conference on (pp. 1-5). IEEE.

Chen, W. N., Zhang, J., Chung, H. S. H., Zhong, W. L., Wu, W. G., & Shi, Y. H. (2010). A novel set-based particle swarm optimization method for discrete optimization problems. Evolutionary Computation, IEEE Transactions on, 14(2), 278-300.

de Queiroz, T. A., & Miyazawa, F. K. (2013). Two-dimensional strip packing problem with load balancing, load bearing and multi-drop constraints. International Journal of Production Economics, 145(2), 511-530.

do Prado Marques, F., & Arenales, M. N. (2007). The constrained compartmentalised knapsack problem. Computers & operations research, 34(7), 2109-2129.

Homaifar, A., Qi, C. X., & Lai, S. H. (1994). Constrained optimization via genetic algorithms. Simulation, 62(4), 242-253.

Hoto, R., Arenales, M., & Maculan, N. (2007). The one dimensional Compartmentalised Knapsack Problem: A case study. European Journal of Operational Research, 183(3), 1183-1195.

Hu, M., Wu, T., & Weir, J. D. (2012). An intelligent augmentation of particle swarm optimization with multiple adaptive methods. Information Sciences, 213, 68-83.

Iyer, S. K., & Saxena, B. (2004). Improved genetic algorithm for the permutation flowshop scheduling problem. Computers & Operations Research, 31(4), 593-606.

Joines, J. A., & Houck, C. R. (1994). On the use of non-stationary penalty functions to solve nonlinear constrained optimization problems with GA's. In Evolutionary Computation, IEEE World Congress on Computational Intelligence., Proceedings of the First IEEE Conference on (pp. 579-584).

Kashan, A. H., Kashan, M. H., & Karimiyan, S. (2013). A particle swarm optimizer for grouping problems. Information Sciences, 252, 81-95.

Kennedy, J., & Eberhart, R. C. (1997). A discrete binary version of the particle swarm algorithm. In Systems, Man, and Cybernetics, Computational Cybernetics and Simulation., IEEE International Conference on (Vol. 5, pp. 4104-4108).

Kenndy, J., & Eberhart, R. C. (1995). Particle swarm optimization. In Proceedings of IEEE International Conference on Neural Networks (Vol. 4, pp. 1942-1948).

Kim, B.-I. & Wy, J. (2010) Last two fit augmentation to the well-known construction heuristics for one-dimensional bin-packing problem: an empirical study. The International Journal of Advanced Manufacturing Technology, 50, 1145–1152.

Langeveld, J., & Engelbrecht, A. P. (2012). Set-based particle swarm optimization applied to the multidimensional knapsack problem. Swarm Intelligence, 6(4), 297-342.

Layeb, A., & Boussalia, S. R. (2012). A novel quantum inspired cuckoo search algorithm for bin packing problem. International Journal of Information Technology and Computer Science (IJITCS), 4(5), 58.

Leão, A. A., Santos, M. O., Hoto, R., & Arenales, M. N. (2011). The constrained compartmentalized knapsack problem: mathematical models and solution methods. European Journal of Operational Research, 212(3), 455-463.

Luan, F., Choi, J. H., & Jung, H. K. (2012). A particle swarm optimization algorithm with novel expected fitness evaluation for robust optimization problems. Magnetics, IEEE Transactions on, 48(2), 331-334.

Malthouse, E. C., Qiu, D., & Xu, J. (2012). Optimal selection of media vehicles using customer databases. Expert Systems with Applications, 39(17), 13035-13045.

Michalewicz, Z. (1995). A Survey of Constraint Handling Techniques in Evolutionary Computation Methods. Evolutionary Programming, 4, 135-155.

Modiri, A., & Kiasaleh, K. (2011). Modification of real-number and binary PSO algorithms for accelerated convergence. Antennas and Propagation, IEEE

Transactions on, 59(1), 214-224.

Perez, R. E., & Behdinan, K. (2007). Particle swarm approach for structural design optimization. Computers & Structures, 85(19), 1579-1588.

Quiroz, M., Cruz-Reyes, L., Torres-Jiménez, J., & Melin, P. (2013). Improving the performance of heuristic algorithms based on exploratory data analysis. In Recent Advances on Hybrid Intelligent Systems (pp. 361-375). Springer Berlin Heidelberg.

Rada-Vilela, J., Chica, M., Cordón, Ó., & Damas, S. (2013). A comparative study of multi-objective ant colony optimization algorithms for the time and space assembly line balancing problem. Applied Soft Computing, 13(11), 4370-4382.

Samanta, S., & Chakraborty, S. (2011). Parametric optimization of some non-traditional machining processes using artificial bee colony algorithm. Engineering Applications of Artificial Intelligence, 24(6), 946-957.

Scholl, A., Klein, R., & Jürgens, C. (1997). BISON: A fast hybrid procedure for exactly solving the one-dimensional bin packing problem. Computers & Operations Research, 24(7), 627-645.

Wei-Ying, Z., Yan, L., & Zhuo-Shang, J. (2005). Model and algorithm for container ship stowage planning based on bin-packing problem. Journal of Marine Science and Application, 4(3), 30-36.

Yesil, C., Turkyilmaz, H. & Korkmaz, E. E. (2012) A New Hybrid Local Search Algorithm on Bin Packing Problem. International Conference on Hybrid Intelligent Systems.

Zachariadis, E. E., Tarantilis, C. D., & Kiranoudis, C. T. (2013). Integrated distribution and loading planning via a compact metaheuristic algorithm. European Journal of Operational Research, 228(1), 56-71.