Lecturer Sajjad AGHAREZAEI, MSc
E-mail: Sajjadagharezaee1993@gmail.com
"Qom" University of Technology
Lecturer Mehdi FALAMARZI, MSc
E-mail: Mehdi.falamarzi1371@gmail.com
"Qom" University of Technology

PARTICLE SWARM OPTIMIZATION ALGORITHM FOR THE PREPACK OPTIMIZATION PROBLEM

Abstract. *Packing problem is one of the most well-known problems in inventory control situations. There are some methods for solving such problems one of which is the meta-heuristic algorithm. The PSO method is a procedure that solves problems with a heuristic procedure. In this study, some realistic assumptions are considered and each particle has its speed. Then, these particles transfer in the solution space, and after every iteration, fitness function for each particle is calculated. We intend to solve the packaging problem by using PSO in this investigation. It is considered that the operable cost in this problem is considerable. In this case, we have the instance of the warehouse that should manage a wide range of various shops, requiring a given group of items. Consequently, this paper is solved a Mixed-integer linear programming problem for the pre-pack optimization problem and results shows the presented method could reduce the optimal amount in reasonable time effort.*

Keywords: *particle swarm optimization, algorithm, packing problems, Prepack optimization, meta-heuristics.*

JEL Classification: C61

1. Introduction

Nowadays, packing problems have a valuable effect on industrial environments and many researchers have investigated this problem since 1960. In the mentioned problem, the number of items should be packed and consequently are sent to each shop. As can be predicted, there are some constraints and objectives that have to be noticed. Most of the research that has taken to account for this problem, have considered the minimum number of bins as an objective function (Fischetti et. al., 2015) studied the packing problem with transportation time for the first time (Fischetti et. al., 2015). The subsequent problem is called the Bin Packing Problem, which has

DOI: 10.24818/18423264/53.2.19.17

been generally considered in the literature, both in its one-dimensional version (Martello et al., 1990) and in its higher-dimensional (variants et. al., 2002).
In logistic applications some sort of packing problems with various realistic consumption exist (Iori et. al., 2007) one of the most well-known problems is where some customers should be supplied with one supplier, whereas (Monaci, 2004;Correia et. al., 2008).addressed the problem in which different kinds of bins are obtainable and their costs are absolutely different. In this study, we have considered a specific packing problem, where the required cost of packing items is considerable, or even higher than items price (Fischetti et. al., 2015). In this regard, the warehouse must arrange each customer's requirements (e.g., stores), each demanding a specified set of products (Fischetti et. al., 2015). In this research, an automatic packing system is utilized and products are sent automatically to each customer. For better control on systems costs, the number of pre-defined configurations is designed (Fischetti et. al., 2015).

2. Background and related work

Creating pre-defined configurations for packing items has some advantages some of which are easy transportation and the minimum number of transferred products, but as can be predicted, this limitation can reduce flexibility in the supply chain and in some situations it would be possible that customers' requirements are not satisfied properly (Fischetti et. al., 2015). In order to modify Particle swarm optimization efficiency, numerous PSO alternatives were prepared in literature. These methods contain (Chen et. al., 2013): 1- setting optimal parameters, which can achieve best values in the minimum computational effort, 2- constructing innovative neighborhood solution, 3- crossing PSO with lower search methods, and 4- using multi-swarm methods. An entire analysis on the PSO algorithm was newly proposed by Banks et al. (Banks et. al., 2007; Banks et. al., 2008).A lot of procedure is suggested to develop the Particle swarm optimization presentation. Mentioned methods can be elaborated as follow: (i) learning plans, (ii) multi-swarm arrangements, (iii) neighborhood topologies, (iv) combinations with other swarm intellects (SI) procedures, and (v) parameter setting.

2.1 The first particle swarm algorithm

In 1995, Eberhart and Kennedy first developed an algorithm for particle swarm optimization as an uncertain search method for practical optimization. This algorithm is stimulated by the movement of birds looking for food. Numbers of birds in this procedure are searching foods in the search space. There is one portion of food in the search space. Each solution that called a particle in the algorithm is alike to a bird in the bird movement procedure. Consequently, each particle has a fitness value calculated by a fitness function. Particles that are near to foods have more fitness values. In this algorithm, each particle has its speed too. By continuing to look for optimal particles in the search space, the particle continues to near itself to the best

DOI: 10.24818/18423264/53.2.19.17

values. In this way, the collection of particles at the start of the process is created accidentally and by upgrading the generations, they try to find an optimal solution.

In PSO algorithm there is two best value. First, one is the best fitness value which is found so far and is called pbest. The second one that is presented by gbest is the best solution which is found by the population. Finally, the speed and location of the particle are calculated with the following equations.

$$v(t+1) = v(t) + C_1 * rand(t) * (p_{best}(t) - position(t)) + C_2 * rand(t) * (g_{best}(t) - position(t)) \quad (1)$$

$$position(t+1) = position(t) + v(t+1) \quad (2)$$

The right side of equation (1) contains three portions: the first portion is the existing speed of the particle and the second part and third is the rate of change in the speed of the particle and its direction towards the best value. If we do not consider the first part in this equation, then the speed of the particles is determined only with respect to the current position. Therefore, the effect of its current and randomly speed is eliminated. In this means, the best part of the group stops in place and the others transfer towards that particle. In fact, group transfer of birds without the first part of Equation (1) will be a procedure in which the search space reduces and searches are formed around the local solution.
On the other hand, if the first part of the equation is considered, searching in global space can be occurred. The parameter is given by whether or not to set parameters and to zero in three different following modes:
1- We do not set any of these two parameters to zero, which is an algorithm for particles swarm optimization with speed inertia, collective intelligence and nostalgia.
2- Zero the weight of nostalgia and we only act based on collective intelligence and inertia.
3- Zero the weight of collective intelligence and act only based on nostalgia and inertia.

We initialize it by the equation (3). The initial speed is zero according to equation(4).

$$x(0) = x_{\min} + rand(x_{\max} - x_{\min}) \quad (3)$$

$$v(0) = 0 \quad (4)$$

2.1.1 The basic problem of the first particles swarm algorithm

A fundamental problem with the proposed formula for the initial particles swarm algorithm is that it constantly increases and no program is presented to reduce it. If it is necessary, we need to slow down as quickly as possible, so that our particle doesn't

DOI: 10.24818/18423264/53.2.19.17

pass through the global optimum. To do this, there are a few solutions that we'll cover two of them.

1- Using the maximum cutting speed

In this method, in order to avoid over-speeding, a maximum speed is defined so even the speed exceeded, it can be cut.

$$v'(t+1) = \begin{cases} v(t+1) & v(t+1) < v_{\max} \\ v_{\max} & v(t+1) \geq v_{\max} \end{cases} \qquad (5)$$

2- Using the maximum speed of the hyperbolic tangent

In this method, a hyperbolic tangent function is used to limit the speed to a specific value. The difference between this mode and the cutting mode is that in this case, our function is differentiable in all directions, while in the cutting case the derivative is destroyed.

$$v'_{ij}(t+1) = \tanh\left(\frac{v_{ij}(t+1)}{v_{\max j}}\right) v_{\max_j}(t) \qquad (6)$$

In cutting point of cutting mode, the derivative disappears, but in the hyperbolic tangent mode, there is no problem.

2.2 Types of Swarm Particle Algorithms

2.2.1 Continuous Particle Swarm Algorithm

Particle swarm optimization algorithm is a population-based algorithm that is stimulated birds movement for finding foods and firstly starts with initial solutions and gradually try to improve final results. In the particle swarm optimization algorithm, particles are flowing in the search space. The particle movement during their process finds a new generation according to solutions which are found in previous steps. The outcome of the modeling of this behavior is a search procedure that directs population toward successful solutions. Particles learn from each other and according to previous findings, go to their best solutions. The basis of the work of the particle swarm optimization algorithm is based on the attitude that at any specific second, each particle sets its position according to the best position ever found to be taken place in it and the best place exists in the whole neighborhood. In the following, we will briefly explain the concept of collective intelligence.

2.2.2 Collective Intelligence

Collective intelligence is a methodical property that the agents collaborate, and the collective performance of all particles leads to a convergence at the optimal global

DOI: 10.24818/18423264/53.2.19.17

answer. The strength of these algorithms is the lack of their need for a global control. Each particle in these procedures has an independence that can move across the space of the answers and should cooperate with other particles (agents). Two popular intelligence procedures are optimization of ants' nest and particle mass optimization.

3. Methodology

3.1. Steps to perform particle swarm optimization

Sometimes there are differences in the way the algorithm is separated. In other words, the steps are separated in some cases, and in some cases, they combine two or more steps together into one step, but this does not cause any problems in the programming, because what matters is the execution of the program steps in the order that follows. For example, in some references, steps 4 and 5 are combined; therefore, the update phase of particle speed and the transfer of particles to new locations considered as a stage. This change will not cause a problem in the implementation of the algorithm. In the following, we will study 6 steps of PSO:

Step 1: Random generation of initial particles population

Random generation of the initial population is simply a random determination of the initial site of particles with uniform distribution in the solution space (search space). The random generation step of the initial population exists in almost all probabilistic optimization algorithms, but in this algorithm, in addition to the initial random location of particles, a value is also allocated for the initial speed of particles. The initially suggested range for particles speed can be extracted from the following equation (7).

$$\frac{X_{\min} - X_{\max}}{2} \leq V \leq \frac{X_{\max} - X_{\min}}{2} \qquad (7)$$

Step 2: Selecting the number of initial particles

We know that growing the number of initial particles reduces the number of repetitions needed to converge the algorithm, but sometimes it is observed that users of optimization algorithms assume that this reduction in the number of repetitions means reducing the program execution time to achieve convergence, while such an idea is completely wrong. However, a growth in the number of initial particles leads to a reduction in the number of repetitions, But the growth in the number of particles causes the procedure to spend more time in the particle evaluation stage, which increases the time of the evaluation so that the algorithm's execution time does not decrease until the convergence progresses despite a reduction in the number of repetitions. So growing the number of particles cannot be used to reduce the runtime of the algorithm.

DOI: 10.24818/18423264/53.2.19.17

There is another misconception that the number of particles can be reduced to reduce the runtime of the algorithm. But to get the algorithm to the optimal answer, the number of repetitions increases. (If we consider the convergence condition to be unchanged at the expense of the best member in several successive repetitions), which ultimately does not reduce the runtime of the program to achieve the optimal response. It should also be noted that the reduction in the number of particles may be trapped in the local minimum, and the algorithm can't reach to the minimum. If we consider the convergence condition as the number of repetitions, however, reducing the number of initial particles, the algorithm will be reduced, but the answer obtained will not be an optimal solution to the problem. Because the algorithm is incompletely implemented. In summary, the number of primary population is determined by the problem. In general, the number of initial particles is a compromise between the parameters involved in the problem. Experimentally selecting an initial particles population of 20 to 30 particles is a good choice, which works well for almost all testing issues. You can count the number of particles a bit more than you need to have a little margin of safety when facing local minimum.

Step 3: Evaluating the objective function (cost or expense calculation) of particles

At this step, we must evaluate each particle, which represents a solution to the problem under consideration. Depending on the issue under consideration, the evaluation method will be different. For example, if it is possible to define a mathematical function for a target, by inserting the input parameters (extracted from the particle position vector) in this mathematical function, the cost of this particle will easily be calculated. Note that each particle contains comprehensive data about the input parameters of the problem, which is extracted from this information and puts in the target function. Sometimes it is not possible to define a mathematical function for particles evaluation. This occurs when we link the algorithm to another software or use the algorithm for experimental data. In these cases, information about the input parameters of the software or the test should be extracted from the particle position vector and placed on the software linked to the algorithm or placed on the relevant test. By executing the software or doing the test, observing and measuring the results, the cost of each particle will be determined.

Step 4: Recording the best location for each particle ($P_{i,best}$) and the best location among all the particles ($P_{g,best}$)

At this step, according to the number of repetitions, two states can be verified:

If we are in the first repetition ($t=1$), consider the present position of each particle as the best location found for that particle.

DOI: 10.24818/18423264/53.2.19.17

$$t = 1 \begin{cases} P_{i,best} = X_i(t) \qquad i = 1,2,3,\ldots,d \\ \cos t(P_{i,best}) = \cos t(X_j(t)) \end{cases} \tag{8}$$

In other repetitions, we compare the amount of cost for particles in step 2 with the best cost for every single particle. If this cost is less than the best cost recorded for this particle, then the location and cost of this particle will replace the previous value. Otherwise, there will be no change in the place and cost recorded for this particle, i.e.:

$$t = 2 \begin{cases} if \ \cos t(X_i(t)) < \cos t(P_{i,best}) \\ \quad else\, not change \end{cases} \Rightarrow \begin{cases} \cos t(P_{i,best}) = \cos t(X_j(t)) \\ P_{i,best} = X_i(t) \end{cases} \qquad i = 1,2,\ldots,d \tag{9}$$

Step 5: Updating the speed vector of all the particles

$$V_i(t) = w \times V_i(t-1) + c_1 \times rand_1 \times (P_{i,best} - X_i(t-1)) + c_2 \times rand_2 \times (P_{g,best} - X_i(t-1)) \tag{10}$$

The coefficients w, c_1 and c_2 are determined experimentally according to the problem. But as a general rule, keep in mind that w must be less than one, because if it is larger than one, $V(t)$ is constantly increasing to the point where it is divergent. Note also, however in the theory, the coefficient w, can be negative, but never consider these coefficients to be negative in using of this practical algorithm, since the negativity of w causes oscillation in $V(t)$. Selecting a small value for this coefficient (w) also has problems. Often, in the particles swarm optimization algorithm, this coefficient is positive and ranges from 0.7 to 0.8. The c_1 and c_2 should also not be too large, since the selection of large values for these two factors causes a large deviation of the particles from its path. Often, in the particles swarm optimization algorithm, these coefficients are positive and range from 1.5 to 1.7. It should be noted that the above values are not necessarily the only possible choices for the coefficients w, c_1, c_2 but according to the problem under consideration, there may be better choices other than the above.

Step 6: Convergence test

The convergence test in this algorithm is similar to other optimization algorithms. There are various methods for examining the algorithm. For example, it is possible to determine the exact number of repetitions from the beginning, and at each step, it is checked whether the number of repetitions reached the specified value? If the number of repetitions is smaller than the initial value, then you must go back to step 2. Otherwise, the algorithm ends. Another method that is often used in the convergence test of the algorithm is that if, in continued repetitions, for example, 15 or 20 repetitions, no change in the cost of the best particle is created, then the algorithm

DOI: 10.24818/18423264/53.2.19.17

ends, otherwise you need to return to step 2. The circular diagram (flowchart) of particles swarm optimization algorithm is shown in Figure1.

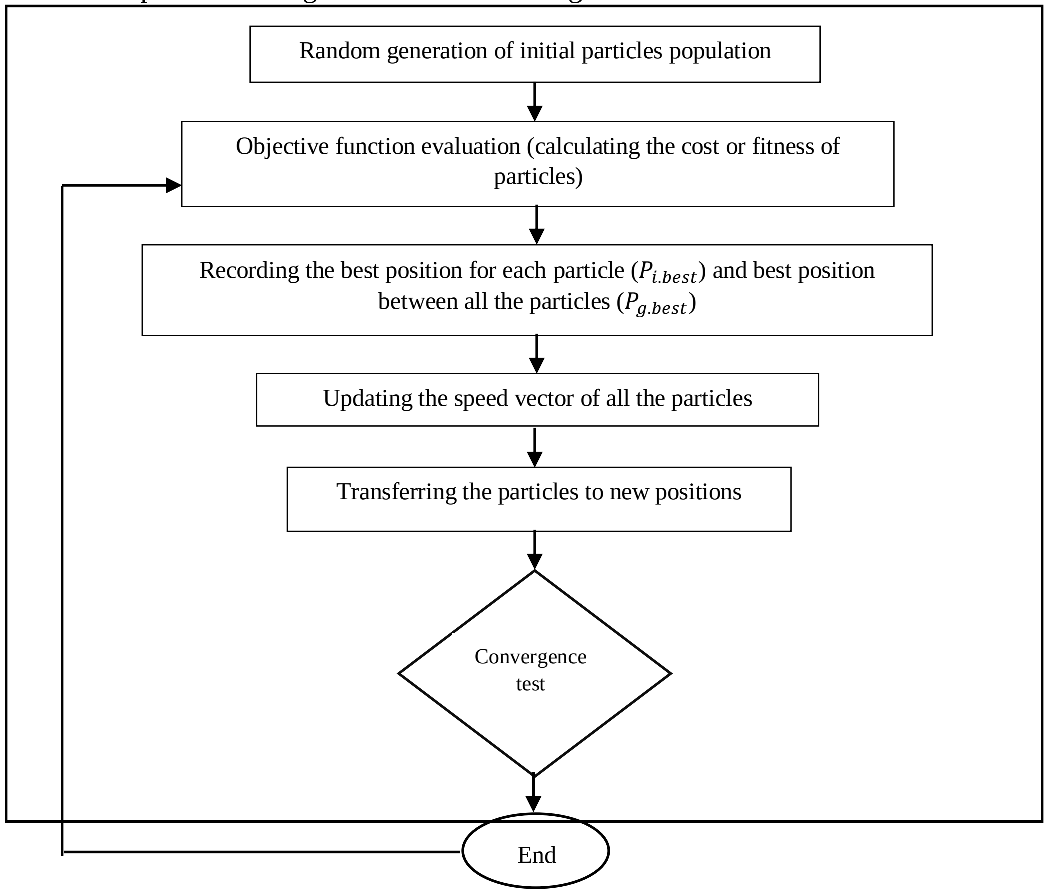
Random generation of initial particles population
Objective function evaluation (calculating the cost or fitness of particles)
Recording the best position for each particle ($P_{i.best}$) and best position between all the particles ($P_{g.best}$)
Updating the speed vector of all the particles
Transferring the particles to new positions
Convergence test
End

Figure1. Circulating diagram of particle swarm optimization algorithm

The pseudo code of the particle swarm optimization algorithm is as follow.

Algorithm 1. Pseudo code

```
1:  For each particle
2:      Initialize particle
3:  End For
4:  Do
5:  For each particle
6:      Calculate fitness value of the particle fp
7:      /*updating particle's best fitness value so far) */
8:      If fp is better than pBest
9:      set current value as the new pBest
10: End For
11:  /*updating population's best fitness value so far) */
12: Set gBest to the best fitness value of all particles
13: For each particle
14:     Calculate particle velocity according equation
15:     Update particle position according equation
16: End For While maximum iterations OR minimum error criteria is not attained
```

But in the case of the stopping condition, the following ways are available.

a. Number of determined repetitions.
b. Reaching a threshold merit.
c. A number of repetitions that do not change the merit (for example, if after 10 repetitions the merit was constant and not better).
d. The last one is based on the aggregate density around the optimal point. In this way, if the 80 percent of the particles be in a distance less than 20 percent of most distance of the best solution, the algorithm should be stopped.
e. In the last methodR_{norm}can be obtained in accordance with equation (11). As stated, $R_{norm}$is a value between 0 and 1, and also F is the greatest distance between two particles in the current state.

$$R_{norm} = (\frac{R_{\max}}{F}) \qquad (11)$$

3.2 Mathematical and suggested algorithm

In this segment, using the data obtained from the problem; we codified the algorithms affecting particles swarm and reported it in the form of a table.

In the research that Fischetti et al. did in 2015 to solve the packaging problem caused by mixed-integer linear programming (Fischetti et. al., 2015), they provided the data in the table1 and determined the experimental variables.

DOI: 10.24818/18423264/53.2.19.17

Table 1. Performance of a black box MILP solver on the instances from (Hoskins et. al., 2014). Single run for each instance. Times in CPU seconds (time limit of 3600 s) (Fischetti et. al., 2015)

Instance	Time (s)	Primal bound	Final gap
Black58	5.4	58	0.0%
Red58	5.1	160	0.0%
Green58	1.0	0	0.0%
Blue58	6.4	0	0.0%
BlackBlue10	352.7	10	0.0%
BlackBlue58	3600.0	583	90.2%
AllColor10	3600.0	407	98.8%
AllColor58	3600.0	6981	99.4%

The sample variables of this test are: Black58, Red58, Green58, Blue58, BlackBlue10, BlackBlue58, AllColor10, AllColor58, and the initial time and limitations of each variable, are also calculated in Table1. However, Finally, Fischetti et al. (2015) compared the variablesBlackBlue10, BlackBlue58, Allcolor10, and Allcolor58 in terms of the value of the target function and the best time by using four heuristic methods like fast heuristic, random, most similar, and most dissimilar heuristic as shown in the Table2.

Table2.Average performance (out of 100 runs) of our heuristics (Fischetti et. al., 2015)

Instance	Heuristic	Time (s)		time best (s)		Pint	Opt
BlackBlue10	fast heu	1.08		1.08		0.34	100
	Random	1.44		1.44		0.27	100
	most dissimilar	1.26		1.26		0.25	100
	most similar	1.25		1.25		0.29	100
BlackBlue58	fast heu	4.61		4.61		9.88	100
	Random	6.40		6.40		10.11	100
	most dissimilar	2.76		2.76		9.13	100

DOI: 10.24818/18423264/53.2.19.17

	most similar	5.62		5.62		15.42	100
AllColor10	fast heu	71.82		71.82		3.81	100
	Random	600.29		304.06		18.63	36
	most dissimilar	704.33		241.12		19.69	27
	most similar	626.15		302.40		20.54	26
AllColor58	fast heu	900.00		332.43		328.20	0
	Random	874.87		329.59		562.95	2
	most dissimilar	893.48		323.93		545.47	1
	most similar	859.86		287.50		404.29	1

We now consider the method of particles swarm optimization for this study. In this segment, we need to implement the execution stepsof the particles swarm optimization algorithm for this research. In the first step, which is called the randomized generation of the initial particles population, we must code the initialization to create the population. Algorithm 2 indicates initialization.

Algorithm 2. Initialization

```
function [InSol]  = initialization (B , S , nPop , K)
InSol = zeros (B , S , nPop) ;
for =1: nPop
k = K ;
I = zeros (B , S) ;
SUM = zeros (B ,1) ;
for I =1:  B
    for j =1: S
        I( I , j) = randi ( [ 0 , k(i) ] ) ;
        k(i) = k(i) - I( i , j) ;
    end
end
for i = 1: B
    SUM (i) = sum ( I( i , : )) ;
end
```

DOI: 10.24818/18423264/53.2.19.17

```
for i=1: B
     if  SUM (i) < K(i)
          t = K(i) - SUM (i) ;
          t1= find(I(i,:) = = min ( I( i , : ))) ;
          I ( i , t1(1) ) = t + I ( i , t1 (1) ) ;
     end
  end
 InSol (: , : , z) = I( : , : ) ;
 end
 end
```

In the next step, we need to allocate some value for the initial velocity of particles, which we have specified in Algorithm 3.

Algorithm 3. Speed

```
function V=speed(c1, c2, pbest, gbest, InSol)
V=(c1* rand *(pbest - InSol.fitness) + c2 * rand * (gbest - InSol.fitness));
end
```

In the step 3, we must evaluate and record each particle representing a solution to the problem under consideration, which we have identified in Algorithm 4.

Algorithm 4. Recording

```
 function [pbest, gbest]=recording(InSol,nPop)
pbest=zeros(1, 1, nPop);
 for z=1:nPop
       pbest (z)=InSol.fitness (:, :, z);
end
       gbest =min(InSol.fitness);
end
```

In the step 4, we need to determine the best location for each particle and the best location among all the population, which we have specified in Algorithm 5.

Algorithm 5. Position

```
 function [P]=position(alfa, beta, InSol, r, S, nPop)
O=zeros(1, S, nPop);
U=zeros(1, S, nPop);
```

DOI: 10.24818/18423264/53.2.19.17

```
SUM=zeros(1, S, nPop);
for j=1: S
    SUM=sum(InSol.value);
end
for z =1: nPop
for j=1: S
    if SUM( :, j, z)==r(j)
        continue
    else if SUM(:, j, z) > r(j)
        O(:, j, z)=SUM(:, j, z) - r(j);
    else
        U(:, j, z)=r(j) - SUM(:, j, z);
    end
 end
 end
 t1= alfa * U;
 t2= beta * O;
 P =sum( t1+t2);
 end
```

In step 5, we need to update the velocity vector of all particles, which we did in Algorithm 6.

Algorithm 6. Updating

```
function bestsol = updating(InSol, nPop, bestsol)
for z =1: nPop
if InSol . Fitness <bestsol .fitness
     bestsol(: , : , z) = InSol(: , : , z);
  end
end
```

DOI: 10.24818/18423264/53.2.19.17

In the final step, we implement the convergence test and the general scheme of particles swarm optimization algorithm shown in Algorithm 7.

Algorithm 7. PSO algorithm

```
clc;
clear;
%% PSO algorithm
%% problem parameters
alfa =1;
beta =1;
r =[10, 10, 10];
K =[8; 8; 8];
S =3;
B =3;
%% algorithm parameters
nPop =100;
c1=2;
c2 =2;
MNI =2000;

%% algorithm main loop
bestsol.value = zeros(B, S);
bestsol.fitness =1000 * ones(1, 1, nPop);
bestsol.V =zeros (1, 1, nPop);

bestsolstage =zeros (MNI, 1);
for m =1: MNI

% initialize random solution
InSol.value = zeros(B, S);
InSol.fitness =zeros (1, 1, nPop);
InSol.V = zeros(1, 1, nPop);
InSol.value = initialization(B, S, nPop, K);

% calculating particles positions(costs)
InSol.fitness = pisition(alfa, beta, InSol, r, S, nPop);
```

DOI: 10.24818/18423264/53.2.19.17

```
% recording current and best position for population
[pbest , gbest]= recording(InSol, nPop);

% calculating current speed and position for every particles
InSol.V = speed (c1 ,c2 ,pbest, gbest, InSol);
InSol.fitness = InSol.fitness + InSol.V;

%updating best solution which is found so far
bestsol = updating (InSol, nPop, bestsol);

%showing algorithm convergence
MIN = min(bestsol.fitness);
bestsolstage (m)=MIN;
End
plot(bestsolstage)
```

In Table 3, the outcomes of the particles swarm optimization algorithm are calculated in terms of the value of the target function and runtime in the reference article.

Table3.Results of Particle Swarm Optimization Algorithms

Instance	Heuristic	Time (s)	Time best (s)	Opt
BlackBlue10	PSO	1.68	1.68	100
BlackBlue58	PSO	5.22	5.22	100
Allcolor10	PSO	405.33	405.33	20
Allcolor58	PSO	844	844	1

4. Result and discussion

The simulated behavior of birds ‘population, which is known as particle swarm optimization (PSO), was proposed in (Eberhart and Shi, 2001). In opposite of other population-based evolutionary algorithms, Particle Swarm Optimization researches the search space according to 2 key “leaders”: p_{best} andg_{best}, which are the prior best postures achieved by the individual particle and the swarm so far, individually. As the Particle Swarm Optimization researches algorithm is simplified in conception, prosperous to utilize and computationally cheap, it attracts the attention of many investigators in recent years. PSO has now been effectively utilized to a varied range

DOI: 10.24818/18423264/53.2.19.17

of application areas (Agrawal and Bawane, 2015; Gonzalez et. al., 2015;Ishaque et. al., 2012;Sa-ngawongand Ngamroo, 2015; Wang and Yang, 2013) engineering design(Jin and Rahmat-Samii, 2010; Kachroudi et. al., 2012; Yeung et. al., 2012) neural networks (Bevrani et. al., 2012; Georgeand Panda, 2012; Silva et. al., 2010; Valdez et. al., 2014; Xiong et. al., 2015) and so on.

However, Particle Swarm Optimization researches has a quick rate of convergence and worldwide search ability when searching in the unimodal problem space, it frequently gets stuck and snare in local minima when applied to more complex problems such as multimodal or shifted and rotated functions (Chen, 2012).In Table4, the results of the particles swarm algorithm are compared with the algorithms examined in the reference article in terms of the value of the target function and runtime.

Table4.Results of the algorithm of particle swarm with other methods

Instance	Heuristic	Time (s)	Time best (t)	Opt
BlackBlue10	Fast-heuristic	1.08	1.08	100
	Random	1.44	1.44	100
	Most-dissimilar	1.26	1.26	100
	Most-similar	1.25	1.25	100
	PSO	1.68	1.68	100
BlackBlue58	Fast-heuristic	4.61	4.61	100
	Random	6.4	6.4	100
	Most-dissimilar	2.76	2.76	100
	Most-similar	5.62	5.62	100
	PSO	5.22	5.22	100
Allcolor10	Fast-heuristic	71.82	71.82	100
	Random	600.29	600.29	36
	Most-dissimilar	704.33	704.33	27
	Most-similar	626.15	626.15	26
	PSO	405.33	405.33	20

DOI: 10.24818/18423264/53.2.19.17

Allcolor58	Fast-heuristic	900	900	0
	Random	874.87	874.87	2
	Most-dissimilar	893.48	893.48	1
	Most-similar	859.86	859.86	1
	PSO	844	844	1

Regarding the comparisons made in Table 4, it was observed that the method of particles swarm optimization algorithm obtained better results than other four heuristic methods in some variables. For example, in the Allcolor58 section, using particles swarm optimization, the optimal time was 844 seconds, which is better than the other four methods. Alternatively, in the Allcolor10 section, using particles swarm optimization, the optimum time was 405.33, which is better than the other four methods, and also by using particles swarm optimization method, obtained optimum 20 and was recognized as the best answer. So it can be concluded that the method of particles swarm optimization is one of the most suitable and practicable methods for minimization.

5. Conclusion

In this paper, the method of particles swarm optimization algorithm was compared with Fast Heuristic, Random, Most-dissimilar and Most-similar methods to solve the packaging problem caused by inventory allocations. As shown in Table 4, in the Allcolor58 section, the best optimum time was obtained using particle swarm method and its time was estimated to be 844 seconds, which was found to be significantly lower than other methods, and the optimal value was obtained using the Fast Heuristic Method and its value was estimated to be 0. In Allcolor10, the best optimum time was obtained using particles swarm method, and its time was estimated to be 405.33 seconds, which was significantly reduced compared to other methods, and the optimal value was obtained by using the particles swarm method and its value was estimated to be 20.

Considering the findings of this paper and also because the particles swarm optimization algorithm is one of the most successful optimization algorithms due to its simplicity and efficiency, the work of this research can be extended from other aspects. Among other things, research and comparison of other parameters, such as the inertial coefficient, acceleration coefficients, etc., can be obtained for the best possible solution for the desired problem.

It can be concluded that the method of optimizing particles swarm is a useful method for minimization problems.

DOI: 10.24818/18423264/53.2.19.17

REFERENCES

[1] **Fischetti,M., Monaci, M.,Salvagnin, D. (2015),** ***Mixed-integer Linear Programming Heuristics for the Prepack Optimization Problem;*** *Discrete Optimization*, Pages 195-205;
[2] **Martello, S., Toth, P., Knapsack (1990),** ***Problems: Algorithms and Computer, Implementations;*** *John Wiley & Sons, Chichester*;
[3] **Lodi, A., Martello, S., Monaci, M. (2002),** ***Two-dimensional Packing Problems: A Survey;*** *European;J. Oper. Res*, 14, 241–252;
[4] **Iori, M., Salazar-Gonzales, J., Vigo, D. (2007),** ***An Exact Approach for the Vehicle Routing Problem with Two-Dimensional Loading Constraints;*** *Transp. Sci.* 41, 253–264;
[5] **Monaci, M. (2004),** ***Algorithms for Packing and Scheduling Problems;****4OR*,85–87;
[6] **Correia, I., Gouveia, L., Da Gama, F.S. (2008),** ***Solving the Variable Size Bin-Packing Problem with Discretized Formulations;****Comput. Oper. Res*, 35, 2103–2113;
[7] **Chen, W.N., Zhang, J., Lin, Y., Chen, N., Zhan, Z.H., Chung,H.S.H., and et al. (2013),** ***Particle Swarm Optimization with an Aging Leader and Challengers;*** *IEEE Transactions on Evolutionary Computation*, 17, 241–258;
[8] **Banks, A., Vincent, J., Anyakoha, C. (2007),** ***A Review of Particle Swarm Optimization. Parti: Background and Development;*** *Nat Compute*. 6, 467–84;
[9] **Banks, A., Vincent, J., Anyakoha, C. (2008), A** ***Review of Particle Swarm Optimization. Part ii: Hybridization, Combinatorial, Multi Criteria and Constrained Optimization, and Indicative Applications;*** *Nat Compute*, 7,109–24;
[10] **Hoskins, M., Masson, R., Melanon, G., Mendoza, J., Meyer, C., Rousseau, L.M. (2014),** ***The Prepack Optimization Problem,*** *in: H. Simonis (Ed.), Integration of AI and OR Techniques in Constraint Programming****;*** *Lecture Notes in Computer Science*, vol. 8451,pp. 136–143;
[11] **Eberhart, R., Shi, Y. (2001),** ***Particle Swarm Optimization: Developments, Applications and Resources;*** *Proceedings of IEEE Congress on Evolutionary Computation*, pp. 81–86;
[12] **Agrawal, R.K., Bawane, N.G. (2015),** ***Multi Objective PSO Based Adaption of Neural Network Topology for Pixel Classification in Satellite Imagery;*** *Appl. Soft Compute*. J. 28, 217–225;
[13] **Gonzalez, M., López, A., Jurado, F. (2015),** ***Corrigendum to Optimization of Distributed Generation Systems Using a New Discrete PSO and OPF;*** *Electric Power Systems Research*, Vol. 121, Page 379;
[14] **Ishaque, K., Salam, Z., Amjad, M., Mekhilef, S. (2012),** ***An Improved Particle Swarm Optimization (PSO)-Based MPPT for PV with Reduced Steady-State Oscillation;*** *IEEE Trans. Power Electron*, 27, 3627–3638;
[15] **Sa-Ngawong, N., Ngamroo, I. (2015),** ***Intelligent Photovoltaic Farms for Robust Frequency Stabilization in Multi-Area Interconnected Power System Based on PSO-Based Optimalsugeno Fuzzy Logic Control;*** *Renew. Energy*, 74, 555–567;

DOI: 10.24818/18423264/53.2.19.17

[16] **Wang, J., Yang, F., (2013),** ***Optimal Capacity Allocation of Standalone Wind/Solar/Battery Hybrid Power System Based on Improved Particle Swarm Optimization Algorithm;*** IET Renew. Power Generate,7, 443–448;
[17] **Jin, N., Rahmat-Samii, Y. (2010),** ***Hybrid Real-Binary Particle Swarm Optimization (HPSO) In Engineering Electromagnetics;*** *IEEE Trans. Ant. Prop*. 58, 3786–3794;
[18] **Kachroudi, S., Grossard, M., Abroug, N. (2012),** ***Predictive Driving Guidance of Full Electric Vehicles Using Particle Swarm Optimization;*** *IEEE Trans. Vehicle Technol.*, 61, 3909–3919;
[19] **Yeung, S.H., Chan, W.S., Ng, K.T., Man, K.F. (2012),** ***Computational Optimization Algorithms for Antennas and RF/Microwave Circuit Designs: An Overview;*** *IEEE Trans. Ind. Inf*, 8, 216–227;
[20] **Bevrani, H., Habibi, F., Babahajyani, P., Watanabe, M., Mitani, Y. (2012),** ***Intelligent Frequency Control in an AC Microgrid: Online PSO-Based Fuzzy Tuning Approach;*** *IEEE Trans. Smart Grid*, 3, 1935–1944;
[21] **George, N.V., Panda, G. (2012),** ***A Particle-Swarm-Optimization-Based Decentralized Nonlinear Active Noise Control System;*** *IEEE Trans. Instrum. Meas*, 61, 3378–3386;
[22] **Silva, P.H., Cruz,R.M.S., Assuncao, A.G.D. (2010),** ***Blending PSO and ANN for Optimal Design of FSS Filters with Koch Island Patch Elements;*** *IEEE Trans. Magn.*, 46, 3010–3013;
[23] **Valdez, F., Melin, P., Castillo, O. (2014),** ***Modular Neural Networks Architecture Optimization with a new Nature Inspired Method Using a Fuzzy Combination of Particles Warm Optimization and Genetic Algorithms;*** *Inf. Sci*, 270, 143–153;
[24] **Xiong, T., Bao, Y., Hu, Z., Chiong, R. (2015),** ***Forecasting Interval Time Series Using a Fully Complex-Valued RBF Neural Network with DPSO and PSO Algorithms;*** *Inf. Sci*, 305, 77–92;
[25] **Chen, P. (2012),** ***Two-Level Hierarchical Approach to Unit Commitment Using Expertsystem and Elite PSO;*** *IEEE Trans.PowerSystem*, 27, 780–789.

DOI: 10.24818/18423264/53.2.19.17