www.editada.org

International Journal of Combinatorial Optimization Problems and Informatics, 15(3), Sep-Dec 2024, 101-114. ISSN: 2007-1558.
https://doi.org/10.61467/2007.1558.2024.v15i3.398

A Novel Particle Swarm Optimization Algorithm to Solve the Internet Shopping Optimization Problem with Shipping Costs

Miguel Á. García-Morales, José A. Brambila-Hernández, Héctor J. Fraire-Huacuja, Laura Cruz-Reyes, Nelson Rangel-Valdez, Juan Frausto-Solís, Claudia G. Gómez-Santillán*

National Technology of Mexico / Madero City Technological Institute, Madero City, Tamaulipas, Mexico.
soporteclusterlanti@gmail.com (M.A.G-M); jabrambila@gmail.com (J.A.B-H); automatas2002@yahoo.com.mx (H.J.F-H); lauracruzreyes@itcm.edu.mx (L.C.-R.); nelson.rangel@itcm.edu.mx (N.R-V); nce: juan.frausto@gmail.com or juan.frausto@itcm.edu.mx (J.F.-S.); claudia.gomez@itcm.edu.mx (C.G.G-S)
* Correspondence: jabrambila@gmail.com

Abstract. In this paper, we approach the Internet Shopping Optimization Problem with Shipping Costs (IShOP), an NP-hard-relevant problem in the current e-commerce environment. To our knowledge, several solution metaheuristic algo-rithms have been reported in the literature. In this paper, we propose a novel Particle Swarm Optimization algorithm (PSO) to solve the problem. PSO in-corporates a neighborhood diversification technique (NDT), a local search (LS) technique, and an adaptive parameter tuning (APT) method. The proposed al-gorithm (NDTLSAA-PSO) includes two techniques at the end of each iteration to avoid premature convergence in the search process, balancing exploration and exploitation in selecting candidate solutions. A comparison of the perfor-mance of the proposed algorithm has been made against the performance of the best state-of-the-art memetic algorithm solution of the IShOP. A total of 90 in-stances classified into three groups of small, medium, and large sizes were re-solved. The Wilcoxon and Holm non-parametric tests were applied to validate the significance of the differences observed between these two algorithms. The results show that the proposed NDTLSAA-PSO algorithm is superior to the memetic algorithm. In addition, the proposed algorithm obtains the best results in all the in-stances evaluated in terms of quality.

Keywords: Particle Swarm Optimization, Diversification, Local Search, Internet Shopping Problem, Adaptive Parameter Tuning.

Article Info
Received Nov 27, 2023
Accepted May 20, 2024

1 Introduction

Electronic commerce in recent years has gained significant relevance among consumers due to the tremendous demand for transactions carried out after the health crisis caused by Covid-19 (Martínez et al, 2022). A particular case is Internet shopping because, to this contingency, customers of various online stores have preferred to make their purchases from the comfort of their homes to safeguard their health (Martínez et al, 2022). Due to this, the Internet Shopping Optimization Problem (IShOP) has become one of the most studied, using different solution methods such as heuristics, integer linear programming, metaheuristics, and others (Huacuja et al., 2021). In (Gen & Cheng, 2000) the problem was proposed, and they show that it is NP-hard.

Two metaheuristic solutions have been reported in the state-of-the-art. The first reported solution is a cellular Processing algorithm that simulates parallel processing through cells (López et al., 2015). The second is a memetic algorithm (Huacuja et al., 2021) corresponding to the best state-of-the-art IShOP algorithm with shipping costs. This metaheuristic algorithm obtains approximate solutions to the optimal solutions in considerable time and adapts to different problems.

Kennedy and Eberhart propose for the first time a particle swarm optimization (PSO) algorithm (Kennedy & Eberhart, 1995), (Eberhart & Kennedy, 1995); this metaheuristic algorithm has been used to solve problems in different areas (Gad, 2022).

Although the PSO algorithm has premature convergence problems, finding the globally optimal values could be more efficient. In order to avoid slow convergence problems and increase the performance of the PSO algorithm, the following main contributions are proposed in this research work:

- A novel particle swarm optimization algorithm to solve the IShOP problem with shipping costs.
- A neighborhood diversification technique (NDT) is avoid local stagnation in the search process of the best solutions.
- A local Search (LS) only affects the global solution obtained at the end iteration.

In addition, an adaptation of the bandit-based adaptive operator selection method, FRRMAB (Li et al., 2014), has been carried out, which contains a series of modifications that allow for an adaptive adjustment of the social learning parameters (c_1, c_2).

To determine the feasibility of the proposed algorithm, an evaluation of the instances used in (Huacuja et al., 2021) has been carried out. Furthermore, the results obtained during the experiments have been analyzed, and the conclusions are established. This evaluation and analysis have been carried out in order to validate the feasibility of the proposed PSO algorithm (NDTLSAA-PSO).

Formal definition of the problem

A customer needs to buy a set of n products N online, which he can buy in a set of m available stores M. The set N_i contains the products available in-store i, each product $j \in N_i$ has a cost of c_{ij} and a shipping cost d_i.

We add the shipping cost only if the customer buys one or more products from in store i. The problem is to minimize the total cost of buying all the products in N (cost of the products plus shipping).

Formally, the problem consists of determining a partition of the products to be bought in the different stores $I = (I_1, \dots, I_m)$, so that $X_i \subseteq N_i$ and that all the products $U_{i=1}^{m} I_i = N$ and that the total cost is minimized (Huacuja et al., 2021), we calculate the objective value of the solution I with Eq. 1:

$$F(I) = \sum_{i=1}^{m} \sum_{j=1\&\&I(j)=i}^{n} \left(d_i + c_{ij}\right) \qquad (1)$$

If the list of products that the customer will buy consists of N products and there are M stores available, then a solution is represented in a vector I of length N, which contains for each product the store where it will be purchased. A more detailed description can be found in Błażewicz et al. (Błażewicz et al, 2010).

2 General structure of the proposed PSO algorithm (NDTLSAA-PSO)

A In this section, a description of the proposed algorithm (NDTLSAA-PSO) and its main components is made.

Neighborhood diversification technique (NDT)

This technique is applied to each of the particles in the population. Initially, the first particle is selected, and later it is done by generating a random number within the range of available stores [1, *M*]. Then, in the current particle, the first element is selected. Next, its value is modified according to the following two rules: if the position of the current particle is even, the total number of available stores *M* must be subtracted by the value that is produced by adding the best position of the current particle and the previously randomly generated r value, conversely, if the current particle position is odd, then *M* must be subtracted by the value that is produced by subtracting the best position from the current particle and the random r value. After the above, a repair process is applied to avoid violating the minimum and maximum position constraints. This process will continue until that all the elements of the particle are traversed. Subsequently, the total cost of the updated particle is calculated, and it is verified if there is an improvement concerning its best cost; if there is an improvement, the current particle is established as the best global particle. This process will end when all the population particles have been modified and evaluated. The process described above is shown in Algorithm 1.

Algorithm 1 Neighborhood diversification technique (NDT)

Inputs: Population of solutions

Outputs: Updated Population.

Variables:

$particles$: Population

$particles.size$: Population size

N: Number of products

M: Number of stores

Functions:

$random\_number\ [1, M]$: Generates a Random number in the range [1, M].

$ObjectiveFunction(particles_i)$: Calculate objective value of the particle.

$Max(particles_i.position[j], 1)$: Calculates the maximum value between the j position of a particle and 1.

$Min(particles_i.position[j], M)$: Calculates the minimum value between the position j of a particle and M.

```
1:  $\textbf{for}\ i = 1\ \textbf{to}\ Particles.size\ \textbf{do}$
2:  $r = random\_number\ [1, M]$
3:    $\textbf{for}\ j = 1\ \textbf{to}\ N\ \textbf{do}$
4:        $\textbf{if}\ i\ \%\ 2 == 0\ \textbf{then}$
5:            $particles_i.Position[j] = M - (particles_i.BestPosition[j] + r)$
6:            $particles_i.position[j] = Max(particles_i.position[j], 1)$
7:            $particles_i.position[j] = Min(particles_i.position[j], M)$
8:        $\textbf{else}$
9:            $particles_i.Position[j] = M - (particles_i.BestPosition[j] - r)$
10:           $particles_i.position[j] = Max(particles_i.position[j], 1)$
11:           $particles_i.position[j] = Min(particles_i.position[j], M)$
12:       $\textbf{endif}$
12:   $\textbf{endfor}$
13: $ObjectiveFunction(particles_i)$
14: $\textbf{if}\ particles_i.Cost < particles_i.bestCost\ \textbf{then}$
15:     $particles_i.bestPosition = particles_i.position$
16:     $particles_i.bestCost = particles_i.cost$
17:     $\textbf{if}\ particles_i.bestCost < GlobalBest.cost\ \textbf{then}$
18:         $GlobalBest = particles_i$
19:     $\textbf{endif}$
20: $\textbf{endif}$
21: $\textbf{endfor}$
22: $\textbf{return}\ Particles$
```

Local Search (LS)

The local search process is presented in Algorithm 2. It starts by selecting the best global particle from the entire population. Afterward, each element of the particle N_i starts a traversal, and a new position M_j is assigned to it. Once the change is applied, it is evaluated if it improves the total cost of the particle, this process is repeated until reviewing all the positions in *M*. Nor will it be updated with the best position found. Subsequently, the total cost of the updated particle will be calculated and compared with the best cost of the current particle. If the cost of the updated particle is better than the current one, the updated particle will now be the best global particle.

Algorithm 2 Local Search (LS)

Inputs: Best Global solution
Outputs: Best Global Solution Updated.

Variables:
N: Number of products
M: Number of stores
$Cost$: Objective value cost

Functions:
$ObjectiveFunction(GlobalBest)$: Calculate objective value of the Global Best

```
1.  $Cost = ObjectiveFunction(GlobalBest)$
2.  $\textbf{for}\ i = 1\ \textbf{to}\ N\ \textbf{do}$
3.      $\textbf{for}\ j = 1\ \textbf{to}\ M\ \textbf{do}$
4.          $GlobalBest.Position[i] = j$
5.          $\textbf{if}\ ObjectiveFunction(GlobalBest) < Cost\ \textbf{then}$
6.              $Cost = ObjectiveFunction(GlobalBest)$
7.              $j' = j$
8.          $\textbf{endif}$
9.      $\textbf{enfor}$
10. $GlobalBest.Position[i] = j'$
11. $\textbf{endfor}$
12. $Objectivefunction(GlobalBest)$
13. $\textbf{if}\ (GlobalBest.Cost < GlobalBest.bestCost)\ \textbf{then}$
14.     $GlobalBest.bestPosition = GlobalBest.Position$
15.     $GlobalBest.bestCost = GlobalBest.Cost$
16. $\textbf{endif}$
17. $\textbf{return}\ GlobalBest$
```

Applying the NDT and LS techniques (Fig. 1) will allow the algorithm not to remain stagnant and quickly converge to an optimal or close to an optimal solution.

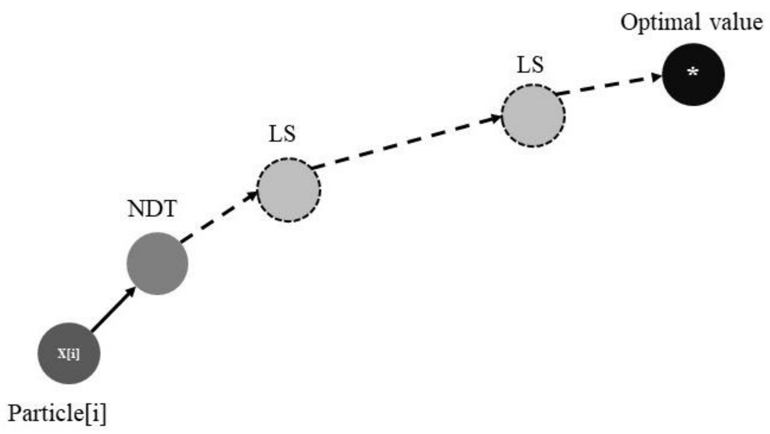

Figure 1 Application of the NDT and LS mechanisms to a particle.

Adaptive parameter tuning (APT)

In order to perform a correct online adjustment of the control parameters, a set of predefined parameter values is usually initialized. Subsequently, these values are adjusted with a specific objective established at the convenience of each user during the search process.

Taking into account that mentioned above, in this research work, we have worked on taking an adaptive operator selection (AOS) method and modifying it so that it is capable of performing an adaptive tuning of parameters. The AOS used is Fitness-Rate-Rank-Based Multi-armed Bandit Adaptive (FRRMAB), proposed in 2014 by Ke Li Ke Li (Li et al., 2014), based on multi-armed bandits. Initially, the AOS is used to select genetic operators; however, through a modification, it has been possible to successfully perform an adaptive parameter adjustment.

Assignment of credits

According to Fialho (Fialho et al., 2010), using the direct value of the improvement in aptitude without being processed is not recommended. Because using the raw value directly causes a deterioration in the robustness of the algorithm. The FRRMAB method avoids this problem using fitness improvement rates (FIR). The formula for calculating these rates is shown in Eq. 2.

$$FIR_{i,t} = \frac{pf_{i,t} - cf_{i,t}}{pf_{i,t}} \quad (2)$$

Where $pf_{i,t}$ is the fitness value of the parent, and $cf_{i,t}$ is the fitness value of the children.

A first-in-first-out (FIFO) structure called a sliding window having a fixed size W is used to store the FIR values of actions that have been recently used. This structure allows us to keep the most recent actions and eliminate the oldest ones as the sliding window fills up. Each element of the sliding window contains the index of the action and its FIR value, which can be seen in Fig. 2.

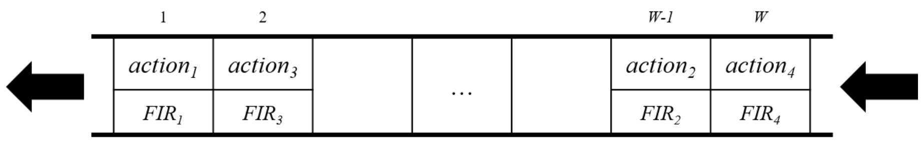

Figure 2 Sliding FIFO window structure based on (Li et al, 2014)**.**

The sliding window ensures that the FIR information stored is for the current search situation.
The reward for each action $(Reward_i)$ is calculated by adding up all the FIR values assigned to each action in the sliding window, then sorted in descending order, using the rank $(Rank_i)$ of each action i. Finally, to allow the best actions to be selected, a decay factor $D \in [0,1]$ is used to transform the $Reward_i$ using Eq. 3:

$$Decay_i = D^{Rank_i} \times Reward_i \quad (3)$$

Then, credit is assigned to action i using Eq. 4.

$$FRR_{i,t} = \frac{Decay_i}{\sum_{j=1}^{K} Decay_j} \quad (4)$$

The lower the D decay value, the greater the influence for the best action. Algorithm 3 shows the credit allocation process.

Algorithm 3 Assignment of credits

```
1: Initialize each reward $Reward_i = 0$
2: Initialize $n_i = 0$;
3: for $i \leftarrow 1$ to SlidingWindow.length do
4:     $action = \text{SlidingWindow.GetIndexaction}(i)$
5:     $FIR = \text{SlidingWindow.GetFIR}(i)$
6:     $Reward_{action} = Reward_{action} + FIR$
7:     $n_{action}++$
8: endfor
9: Rank $Reward_i$ in descending order and set $Rank_i$ to be the rank value of action $i$
10: for action ← to $K$ do
```

```
11:     $Decay_{action} = D^{Rank_{action}} \times Reward_{action}$
12: $\textbf{endfor}$
13:     $DecaySum = \sum_{action=1}^{K} Decay_{action}$
14: $\textbf{for}\ action \leftarrow \textbf{to}\ K\ \textbf{do}$
15:     $FRR_{action} = Decay_{action}/DecaySum$
16: $\textbf{endfor}$
```

Bandit-based action selection

The bandit-based action selection scheme selects an action based on credits using the FRR values as a quality indicator, as shown in Algorithm 4.

Algorithm 4 Bandit-based action selection

```
1: $\textbf{if}$ There are actions that have not been selected $\textbf{then}$
2:     $action_t$ = randomly select a security from the action pool
3: $\textbf{else}$
4:     $action_t = \underset{i=\{1,\dots,K\}}{\operatorname{argmax}} \left( FRR_{i,t} + C \times \sqrt{\frac{2\times \ln(\sum_{j=1}^{K} n_{j,t})}{n_{i,t}}} \right)$
5: $\textbf{endif}$
```

Actions pool

Originally, this mechanism was a pool of variation operators, which has been modified to carry out six actions for adjusting two parameters c_1 and c_2, of the PSO algorithm. Each of the actions is presented below:

1) Action 1
   $c_1 = c_1 + 0.0001$;
   $c_2 = c_2 + 0.0001$;
2) Action 2
   $c_1 = c_1 - 0.0001$;
   $c_2 = c_2 - 0.0001$;
3) Action 3
   $c_1 = c_1 + 0.0001$;
4) Action 4
   $c_2 = c_2 + 0.0001$;
5) Action 5
   $c_1 = c_1 - 0.0001$;
6) Action 6
   $c_2 = c_2 - 0.0001$;

Proposed particle swarm optimization algorithm (NDTLSAA-PSO)

This algorithm also uses the neighborhood diversification technique to avoid local stagnation. In steps 1 to 3, the input parameters are initialized, a random population of particles is generated, and the overall best particle is obtained. Step 4 obtains the minimum ($VelMin$) and maximum ($VelMax$) velocity with which each particle will move. In step 7, an action number is obtained, and in step 8, an action is executed to adjust the parameters c_1 and c_2. From step 10 to step 15, an update of the position and velocity of the particle is made, and it is verified that they are within the established ranges; if they are out of range, a repair method is applied. Step 17 calculates the objective value of the particle, and in step 18, the improvement of the cost of the current particle is obtained concerning its best cost. From steps 19 to 26, the value obtained in the previous step is assigned to an improvement vector, and the best position of the current particle is compared with the global particle; if it has a better target value, the current particle will replace the global particle. In step 27, the obtained values of action number ($indiceAccion$) and the improvement of the cost of the current particle ($improvement[i]$) are added to the sliding window. In step 28, the rewards in the sliding window are updated, and in step 29, they are ranked in descending order. In step 31, the neighborhood diversification technique is applied to the entire population and compared to the best global particle; if any new particles generated with the diversification technique have a better target value, the global best particle will be replaced. Step 32 applies the local Search to the best global solution. Finally, step 33 updates the inertial weight (w). The process continues until the maximum number of iterations ($MaxIt$) is reached. The algorithm returns the best global solution obtained during all processes in step 35. The above process was represented in Algorithm 5.

Algorithm 5 Proposed particle swarm optimization algorithm(NDTLSAA-PSO)

Inputs: Population.
Outputs: Global Best Solution.

Variables:
$particles$: Population
$particles.size$: Population size
N: Number of products
M: Number of stores
$MaxIt$: Maximum number of iterations
w: Inertial weight
$wdamp$: Inertial weight damping radius
c_1: Personal learning quotient
c_2: Global learning quotient
$GlobalBest$: Global Best Solution

Functions:
$random\_number\ [0,M]$: Generates a Random number in the range [1, M].
$ObjectiveFunction(particles_i)$: Calculate the objective value of a particle
$Max(particles_i.position_j, 1)$: Calculates the maximum value between the j position of a particle and 1.
$Min(particles_i.position_j, M)$: Calculates the minimum value between the j position of a particle and M.
$NDT(particles)$: Applies the neighborhood diversification technique to all particles.
$LS(GlobalBest)$: Apply local search to the best global solution.
$FRRMAB()$: Gets an index of an action to perform.
$executeAction(actionIndex)$: Execute an action according to an index.
$SlidingWindow\ (actionIndex, improvement[i])$: Sliding window that stores the index of action to be performed and the improvement in the cost of the particle.
$UpgradeRewards(SlidingWindow)$: Update the sliding window rewards.

```
1: Initialize parameters $MaxIt, Particles.size, w, wdamp, c_1, c_2, N, M$.
2: $particles = GeneratePopulation(Particles.size)$
3: $GlobalBest = getBestGlobal(particles)$
4: $velMax = 0.2 * M,\ velMin = -velMax$
5: $\textbf{for}\ it = 1\ \textbf{to}\ MaxIt\ \textbf{do}$
6:   $\textbf{for}\ i = 1\ \textbf{to}\ nPop\ \textbf{do}$
7:     $actionIndex = FRRMAB()$
8:     $executeAction(actionIndex)$
9:     $\textbf{for}\ j = 1\ \textbf{to}\ d\ \textbf{do}$
10:      $particles_i = w \times particles_i.velocity_j + c1 \times random\_number\ [0,1] \times (particles_i.bestPosition_j - particles_i.position_j) + c2 \times random\_number[0,1] \times (GlobalBest.position_j - particles_i.position_j);$
11:      $particles_i.velocity_j = Max(particles_i.velocity_j, velMin);$
12:      $particles_i.velocity_j = Min(particles_i.velocity_j, velMax);$
13:      $particles_i.position_j = particles_i.posicion + particles_i.velocity_j;$
14:      $particles_i.position_j = Max(particles_i.position_j, 1);$
```

```
15:       $particles_i.position_j = Min(particles_i.position_j, M)$;
16:     $\textbf{endfor}$
17:     $Objectivefunction(particles_i)$
18:     $fitnessimprovement = (particles_i.bestCost - particles_i.cost) / particles\_i.bestCost$
19:     $\textbf{if}$ $particles_i.cost < particles_i.bestCost$ $\textbf{then}$
20:       $improvement[i] = fitnessimprovement$
21:       $particles_i.bestPosition = particless_i.position$
22:       $particles_i.bestCost = particles_i.cost$
23:       $\textbf{if}$ $particles_i.bestCost < GlobalBest.cost$ $\textbf{then}$
24:         $GlobalBest = particles_i$
25:       $\textbf{endif}$
26:     $\textbf{endif}$
27:     Add to $SlidingWindow$ $(actionIndex, improvement[i])$
28:     $Upgrade\_Rewards(SlidingWindow)$
29:     Rank Rewards
30:   $\textbf{endfor}$
31:  $NDT(particles)$
32:  $LS(GlobalBest)$
33:  $w = w * wdamp$
34: $\textbf{endfor}$
35: $return$ $GlobalBest$
```

Next, Fig. 3 it is shown in a general way how the various mechanisms presented in the previous algorithm interact.

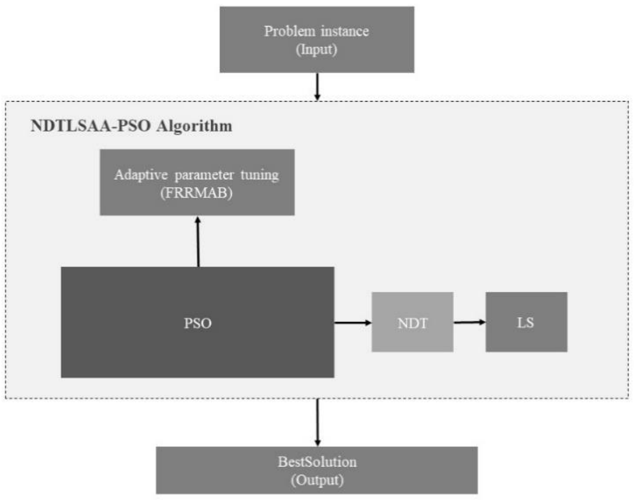

Figure 3 Interaction between algorithm mechanisms.

In addition, the flowchart is also presented in Fig. 4.

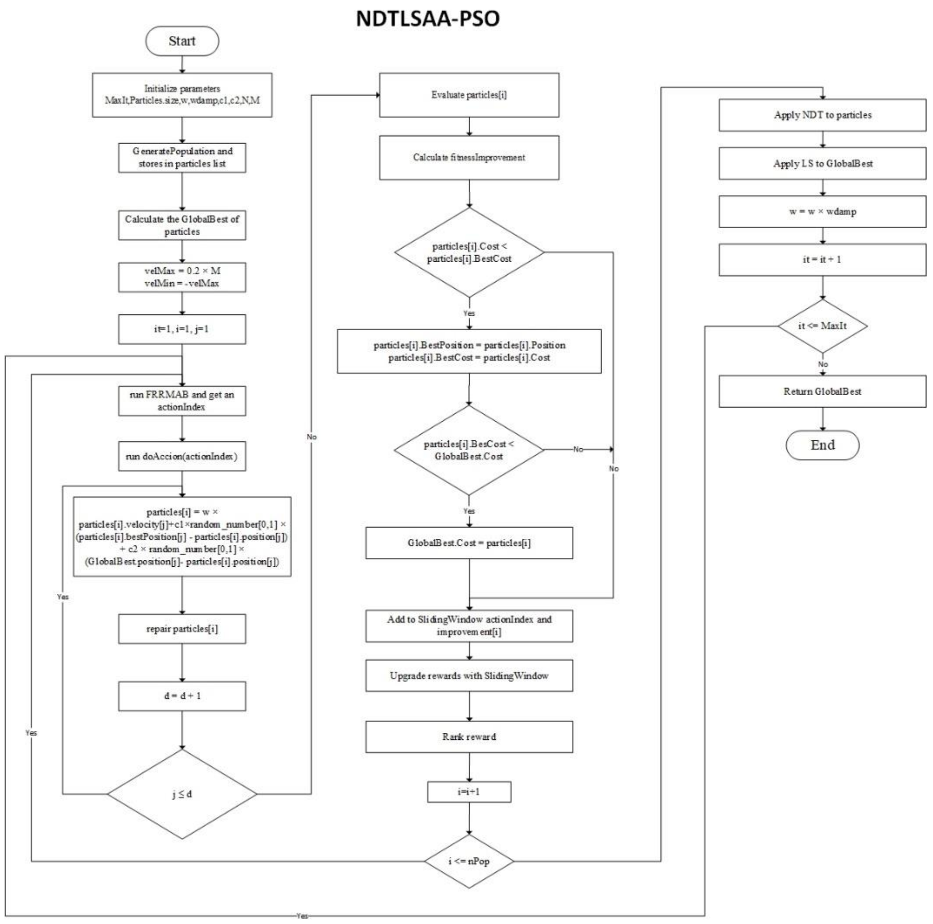
NDTLSAA-PSO
Start
Initialize parameters MaxIt,Particles.size,w,wdamp,c1,c2,N,M
GeneratePopulation and stores in particles list
Calculate the GlobalBest of particles
velMax = 0.2 × M velMin = -velMax
it=1, i=1, j=1
run FRRMAB and get an actionIndex
run doAccion(actionIndex)
particles[i] = w × particles[i].velocity[j]+c1×random_number[0,1] × (particles[i].bestPosition[j] - particles[i].position[j]) + c2 × random_number[0,1] × (GlobalBest.position[j]- particles[i].position[j])
repair particles[i]
d = d + 1
j ≤ d
Yes
Yes
No
Evaluate particles[i]
Calculate fitnessImprovement
particles[i].Cost < particles[i].BestCost
Yes
particles[i].BestPosition = particles[i].Position particles[i].BestCost = particles[i].Cost
particles[i].BesCost < GlobalBest.Cost
No
No
Yes
GlobalBest.Cost = particles[i]
Add to SlidingWindow actionIndex and improvement[i]
Upgrade rewards with SlidingWindow
Rank reward
i=i+1
i <= nPop
Yes
Apply NDT to particles
Apply LS to GlobalBest
w = w × wdamp
it = it + 1
it <= MaxIt
No
Return GlobalBest
End

Figure 4 NDTLSAA-PSO algorithm flowchart.

3 Computational experiments

Table 1 shows the size of the standard instances used in the computational experiments, *n* represents the number of products, and *m* is the number of stores. For each algorithm and instance, 30 independent runs were performed (Huacuja et al., 2021).

Table 1 Standard instance size.

Small instances	Medium instances	Large instances
$3n20m$	$5n240m$	$50n400m$
$4n20m$	$5n400m$	$100n240m$
$5n20m$	$50n240m$	$100n400m$

Table 2 presents the values assigned to the variables and parameters used in the computational experiments for each algorithm analyzed.

Table 2 Variables and parameters.

variables/parameters	MAIShOP	NDTLS-PSO	NDTLSAA-PSO
$population.size$	100	100	100
pc $(crossover\ percentage)$	0.6	--	--
pm $(mutation\ percentage)$	0.01	--	--
er	0.05	--	--
$wdamp$	--	0.99	0.99
$MaxIt$	100	100	100
w	--	1	1
c_1, c_2	--	1.5, 2	1.5, 2*

*start values

4 Results

Table 3 shows the results obtained from the computational experiments performed. The first column identifies the name and configuration of the instances used; the second and third columns show the averages of the best 30 objective values found and the time said solution was determined the MAIShOP reference algorithm. The fourth and fifth columns have the same contents as the previous columns but, in this case, for the proposed NDTLS-PSO algorithm without parameter adjustment. On the other hand, the fourth and fifth columns indicate the p-value obtained from applying the Wilcoxon non-parametric test with a significance level of 5% for each instance. The shaded cells represent the lowest total cost and time. The symbol ↑ indicates a significant difference in favor of the MAIShOP reference algorithm. On the other hand, the symbol ↓ indicates a significant difference in favor of the proposed NDTLS-PSO algorithm, and the symbol "—" indicates no significant difference in favor of either algorithm.

Table 3 Results were obtained with the MAIShOP and NDTLS-PSO algorithms.

Instances	MAIShOP	MAIShOP	NDTLS-PSO	NDTLS-PSO	p-value (OV)	p-*value* (Time)
	OV	Time	OV	Time		
$3n20m$	63.06↓	0.0006↓	62.76	0	0.00512	0.00014
$4n20m$	80.03↓	0.0013↓	78.73	0	0.007686	0.00001
$5n20m$	104.26↓	0.0018↓	102.19	0	0.00262	0.00001
$5n240m$	80.07↓	0.1222↓	75.79	0	0.00044	0.00001
$5n400m$	72.74↓	0.6987↓	69.93	0.0018	0.00424	0.00001
$50n240m$	497.93—	30.1426↓	494.96	0.0422	0.17702	0.00001
$50n400m$	438.74—	33.1063↓	437.75	0.0436	0.14156	0.00001
$100n240m$	993.70↓	50.4910↓	888.99	0.0741	0.00001	0.00001
$100n400m$	801.51↓	176.111↓	728.68	0.0601	0.00001	0.00001
Total average	348.00	32.297	326.642	0.0246		

The shaded cell in gray represents the lowest objective value (OV) or the shortest time when the best solution is found (Time), as the case may be.

Regarding the objective value, the Wilcoxon test results in the NDTLS-PSO algorithm are better in 7 out of 9 instances. In the case of the time used by each algorithm, the NDTLS-PSO algorithm is better in all the evaluated instances.

Table 4 presents a summary of the experiments carried out with each of the three algorithms; the first column shows the names of the instances used. From columns two to seven, the averages of the total cost and the time obtained in the 30 experiments are shown, and the ranking assigned by the non-parametric Friedman test is presented as a sub-index. In this test, a significance level of 5% was applied for each instance. Columns eight and nine show the p-value for the best found objective value and the time that solution was determined, respectively. The shaded cells represent the lowest total cost and time. The symbol ↑ indicates a significant difference in favor of the MAIShOP reference algorithm. On the other hand, the symbol ↓ indicates a significant difference in favor of the proposed algorithm, and the symbol "–" indicates no significant difference in favor of any algorithm.

Table 4 Results were obtained with the MAIShOP and NDTLSAA-PSO algorithms.

Instances	MAIShOP	MAIShOP	NDTLSAA-PSO	NDTLSAA-PSO	p-value (OV)	p-value (Time)
	OV	Time	OV	Time		
$3n20m$	63.06↓	0.0006↓	62.76	0	0.00512	0.00001
$4n20m$	80.03↓	0.0013↓	78.73	0	0.007686	0.00001
$5n20m$	104.26↓	0.0018↓	102.19	0	0.00262	0.00001
$5n240m$	80.07↓	0.1222↓	75.77	0	0.00044	0.00001
$5n400m$	72.74↓	0.6987↓	69.84	0.009	0.00208	0.00001
$50n240m$	497.93↓	30.1426↓	495.55	0.040	0.00001	0.00001
$50n400m$	438.74–	33.1063↓	432.34	0.052	0.07508	0.00001
$100n240m$	993.70↓	50.4910↓	881.66	0.0765	0.00001	0.00001
$100n400m$	801.51↓	176.111↓	727.48	0.069	0.00001	0.00001
Total average	348.00	32.297	325.15	0.0274		

The shaded cell in gray represents the lowest objective value (OV) or the shortest time when the best solution is found (Time), as the case may be.

In terms of the objective value, the Friedman test obtains that the proposed algorithm is better in 8 out of 9 instances. In the case of the time used by each algorithm, the proposed algorithm is better in all the evaluated instances.

Table 5 presents a summary of the experiments carried out with each of the three algorithms; the first column shows the names of the instances used. From columns 2 to 7, the averages of the total cost and the time obtained in the 30 experiments are shown, and the ranking assigned by the non-parametric Friedman test is presented as a sub-index. In this test, a significance level of 5% was applied to the values of the objective value and time. Columns eight and nine show the p-value for the best found objective value and the time that solution was determined, respectively. The shaded cells represent the lowest total cost and time. The symbol ↑ indicates a significant difference in favor of the MAIShOP reference algorithm. On the other hand, the symbol ↓ indicates a significant difference in favor of the proposed algorithm, and the symbol "–" indicates no significant difference in favor of any algorithm.

Table 5 Results were obtained with the MAIShOP, NDTLS-PSO and NDTLSAA-PSO algorithms.

Instances	MAIShOP		NDTLS-PSO		NDTLSAA-PSO		p-value	p-value
	OV	Time	OV	Time	OV	Time		
$3n20m$	63.06_{70} –	$0.0006_{57.5}$ ↓	62.76_{55}	$0_{37.5}$	62.76_{55}	0_{85}	0.08208	0.00012
$4n20m$	80.03_{69} –	$0.0013_{62.5}$ ↓	$78.73_{55.5}$	0_{32}	$78.73_{55.5}$	$0_{85.5}$	0.13199	0.00001

$5n20m$	104.2 6$_{73}$ ↓	0.0018$_{6}$ 7 ↓	102.19$_{5}$ 3.5	0$_{33}$	102.19$_{5}$ 3.5	0$_{80}$	0.014 63	0.000 01
$5n240m$	80.07 76 ↓	0.1222$_{8}$ 5 ↓	75.79$_{52}$	0$_{30}$	75.77$_{51.}$ 5	0$_{65}$	0.001 65	0.000 01
$5n400m$	72.74 71 ↓	0.6987$_{9}$ 0 ↓	69.93$_{53.}$ 5	0.0018$_{3}$ 1.5	60.84$_{55.}$ 5	0000 9$_{45.5}$	0.020 07	0.000 01
$50n240m$	497.9 3$_{88}$ ↓	30.142 6$_{90}$ ↓	494.96$_{4}$ 5.5	0.0422$_{3}$ 4	495.55$_{4}$ 7	0.040 48.5	0.000 01	0.000 01
$50n400m$	438.7 4$_{76}$ ↓	33.106 3$_{90}$ ↓	437.75$_{5}$ 2	0.0436$_{3}$ 5	432.34$_{5}$ 1.5	0.052 50.5	0.001 65	0.000 01
$100n240m$	993.7 0$_{90}$ ↓	50.491 0$_{90}$ ↓	888.99$_{4}$ 5	0.0741$_{3}$ 6	881.66$_{4}$ 4.5	0.076 5$_{45}$	0.000 01	0.000 01
$100n400m$	801.5 1$_{86}$ ↓	176.11 1$_{90}$ ↓	728.68$_{4}$ 8	0.0601$_{3}$ 8	727.48$_{4}$ 6	0.069 48.5	0.000 01	0.000 01
Total average	348	32.297	326.64 2	0.0246	325.15	0.027 4		

The shaded cell in gray represents the lowest objective value (OV) or the shortest time when the best solution is found (Time), as the case may be.

In terms of the objective value, the Friedman test obtains that the proposed algorithm is better in 7 out of 9 instances. In the case of the time used by each algorithm, the proposed algorithm is better in all the evaluated instances.
After performing the Wilcoxon and Friedman non-parametric tests, the Holm Pos Hoc test was carried out, resulting in regarding the value of the total cost. This test shows the following results in Table 6 and Table 7.

Table 6 Holm test applied to the cost of the objective value of each algorithm.

i	Column	z	$p-value$	Calculated Alpha	Significant?
1	MAIShOP vs NDTLSAA-PSO	3.29983165	9.67E-04	0.025	If there is significance
2	MAIShOP vs NDTLS-PSO	3.06412939	0.00218304	0.05	If there is significance

Table 7 Holm test applied to the time of each algorithm.

i	Column	z	$p-value$	Calculated Alpha	Significant?
1	NDTLS-PSO vs NDTLSAA-PSO	-0.23570226	1.186336284	0.025	There is no significance
2	NDTLS-PSO vs MAIShOP	-3.299831646	1.999032572	0.05	There is no significance

5 Conclusions and future work

In this work, we approach the Internet Shopping Optimization Problem with Shipping Costs (IShOP), a relevant problem in the current e-commerce environment. In this paper, we propose a novel Particle Swarm Optimization algorithm (PSO) to solve the problem. A memetic algorithm is the best state-of-the-art to solve the problem. A comparison of the performance of the proposed algorithm has been made, compared with the performance of the memetic algorithm. According to the results of computational experiments, the proposed algorithm (NDTLSAA-PSO) has a clear advantage over the memetic algorithm. The incorporation of the technique of diversification in the neighborhood (NDT), the local Search (LS), and the adaptive adjustment of parameters (FRRMAB) allow us to obtain better results in the evaluation of the three sets of instances. The results show that the proposed algorithm (NDTLSAA-PSO) outperforms the memetic algorithm in 8 of 9 evaluated instances concerning the objective value. Moreover, in terms of time, the proposed algorithm exceeds the memetic algorithm in all instances evaluated. Furthermore, the non-parametric tests carried out showed that there are significant differences in favor of the proposed algorithm.

Finally, as future works, the adaptive adjustment of all the control parameters could be carried out, adding more diversity to the pool of actions and adding the actions according to the parameter one wishes to adjust adaptively.

Declaration of competing interest: The authors declare that they have no known competing financial interest or personal relationship that could have appeared to influence the work reported in this paper.

Data availability: Data will be made available on request.

Acknowledgments: Authors thank TecNM for supporting the projects through research networks such as the Electromobility Network and the Network of Applied Computing Technologies. Miguel Ángel García Morales and José Alfredo Brambila Hernández thank CONACYT for the support 731279 and 760308 respectively. Héctor Fraire thanks the economical support given by National Technological of Mexico to the projects “Optimización Mono-Objetivo y Multi-Objetivo de Problemas Complejos de Empacado” and 10362.21-P.

Contribution of the authors: Conceptualization: H.J.F.H.; Statistical Support: L.C.R; Methodology: N.R.V., C.G.G.S; Research: M.A.G.M.; Software: J.A.B.H., M.A.G.M.; Formal analysis: H.J.F.H., J.F.S; Writing, proofreading, and editing: J.A.B.H, M.A.G.M., N.R.V., J.F.S. All authors read and agree to the publication of the paper.

References

Martínez Valdez, R. I., Catache Mendoza, M. del C., Pedroza Cantú, G., & Huerta Cerda, Z. M. (2022). El impacto del COVID-19 en la incidencia de compras en línea de los millennials. *Revista Ingeniería Y Gestión Industrial, 1*(1). https://doi.org/10.29105/revig1.1-6

Huacuja, H. J. F., Morales, M. Á. G., Locés, M. C. L., Santillán, C. G. G., Reyes, L. C., & Rodríguez, M. L. M. (2021). Optimization of the internet shopping problem with shipping costs. In *Fuzzy Logic Hybrid Extensions of Neural and Optimization Algorithms: Theory and Applications* (pp. 249-255). Springer, Cham.

López Locés, M. C., Rege, K., Pecero, J. E., Bouvry, P., & Huacuja, H. J. F. (2015). Trajectory metaheuristics for the internet shopping optimization problem. In *Design of Intelligent Systems Based on Fuzzy Logic, Neural Networks and Nature-Inspired Optimization* (pp. 527-536). Springer, Cham.

Kennedy, J., & Eberhart, R. (1995). Particle swarm optimization. In *Proceedings of ICNN'95-international conference on neural networks* (Vol. 4, pp. 1942-1948). IEEE.

Eberhart, R., & Kennedy, J. (n.d.). A new optimizer using particle swarm theory. *MHS'95. Proceedings of the Sixth International Symposium on Micro Machine and Human Science*.

Gad, A. G. (2022). Particle swarm optimization algorithm and its applications: A systematic review. *Archives of Computational Methods in Engineering, 29*(5), 2531–2561. Springer Science and Business Media LLC.

Li, K., Fialho, A., Kwong, S., & Zhang, Q. (2014). Adaptive operator selection with bandits for a multiobjective evolutionary algorithm based on decomposition. *IEEE Transactions on Evolutionary Computation, 18*(1), 114–130. https://doi.org/10.1109/TEVC.2013.2240307

Błażewicz, J., Kovalyov, M. Y., Musiał, J., Urbański, A. P., & Wojciechowski, A. (2010). Internet shopping optimization problem. *International Journal of Applied Mathematics and Computer Science, 20*(2), 385-390.

Fialho, A., da Costa, L., Schoenauer, M., & Sebag, M. (2010). Analyzing bandit-based adaptive operator selection mechanisms. *Annals of Mathematics and Artificial Intelligence, 60*(1), 25-64. https://doi.org/10.1007/s10472-010-9184-4

Gen, M., & Cheng, R. (2000). *Genetic algorithms and engineering optimization*. John Wiley & Sons, Inc.