The Local Branching as a Learning Strategy in the Evolutionary Algorithm: The Case of the Set-Union Knapsack Problem

Isma Dahmani and Meriem Ferroum
University of Sciences and Technology Houari Boumedienne, Algeria
Email: {dahmani.isma, meriem1.mf}@gmail.com

Mhand Hifi
University of Picardy Jules Verne, France
Email: hifi@u-picardie.fr

***Abstract*—In this paper, we introduce the local branching as a learning strategy for approximately solving the set-union knapsack problem; that is an NP-hard combination optimization problem. The designed method is based upon three features: (i) applying a swarm optimization for generating a set of current particles, (ii) using an iterative search for providing a series of diversified solutions linking some particles of the population and, (iii) injecting a local branching as a learning strategy for enhancing the global best solution: it can be viewed as a driving strategy employed for guiding particles towards the best position. The performance of the method is evaluated on benchmark instances of the literature, where its provided bounds are compared to those reached by the best methods available in the literature. New bounds have been discovered.**

***Index Terms*—evolutionary, knapsack, learning, branching**

I. Introduction

The Set-Union Knapsack Problem (PSUK) is a variant of the well-known Knapsack Problem (KP), a NP-hard combination optimization problem. An instance of PSUK is characterized by a set of elements and a set of items, where items are related to the elements. Each item is represented by its profit and weight, while an element should be activated whenever an item related to that element is selected. The KP family arises in plenty real-world applications, like cryptography studied by [1] Merkle and Hellman [1], logistics tackled in Perboli *et al.* [2], packing studied in Grange *et al.* [3], and Hifi and Yousef [4], and other problems like multimedia and telecommunications.

In this paper, we investigate the use of an efficient learning strategy for approximately solving PSUK. An instance of the problem is represented by two sets: a set U including elements such that $U = \{1, \dots, n\}$ and a set I of items such that $I = \{1, \dots, m\}$. Each item i, $i \in I$, represents a subset I_i of elements such that $|I_i| = m$ and a non-negative profit p_i while each element j is characterized by its nonnegative weight w_j. The goal of the problem is to search for a subset $I^\star$ of items $I^\star \subseteq I$, where the overall profit $\sum_{i \in T^\star} p_i$ is to maximize without exceeding the knapsack capacity b.

The formal description of PSUK can also be stated as follows. Let $x = (x_1, \dots, x_m) \in \{0,1\}^m$ be an m-dimension vector. $A_x = \{i \mid x_i \in x, x_i = 1,\ 1 \le i \le m\} \subseteq I$, then for an arbitrary $i \in I, x_i = 1$ if and only if $i \in A_x$. Therefore, PSUK is given as follows:

$$z(x) = \max \sum_{i=1}^{m} p_i x_i \quad (1)$$

$$\text{s.t.} \quad W(A_x) = \sum_{j \in \cup_{i \in A_x} U_i} w_j \le b \quad (2)$$

In recent years, it has been observed that the hybridization of procedures may be considered as a very promising research issue. Nevertheless, the difficulty which may be encountered when applying this type methods may be related to the runtime consumed. Herein, we propose an efficient and powerful alternative approach which is able to achieve solutions with high quality and overcomes to the heaviness of this type of methods. However, tailoring a proper hybridization of different methods can make use of advantages of each one and result in a powerful method. Herein, we opt in including a local branching operator, where almost of using a standard swarm optimization, we run the operator for highlighting each improved global best solution related to the current system; that is applied as a learning strategy for enriching the diversified solutions related to a series of local solutions. It is composed of the following features: (i) A random search of positions for generating the first population of particles, where a normalized configuration is applied using the sigmoid function; (ii) A repairing operator used for reaching a feasible solution, and enhanced with local operators; (iii) A local branching operator used as learning strategy for driving the search process. These steps are embedded into an iterative process for highlighting the solutions

Manuscript received November 11, 2021; revised April 19, 2022.

doi: 10.12720/jait.13.3.259-264

The remainder of the paper is organized as follows. The related work is exposed in Section II. Section III discusses the proposed method. Finally, a preliminary experimental study is considered in Section IV.

II. Background

For an exact resolution, Goldschmidt *et al.* [5] designed a dynamic programming-based algorithm, where the initial problem is reduced to the densest *k*-subgraph problem. Experimentally, the authors pointed out the efficiency of the method, especially when solving small-sized instances.

Because exact algorithms are intractable, especially when tackling large-scale instances, several studies were concentrated on designing approximate heuristics. Indeed, Arulselvan [6] discussed a quick approximation algorithm, where a greedy rule was used for iteratively choosing the next item to be added to the partial solution.

He *et al.* [7] designed a binary artificial bee colony-algorithm combined with a mapping function. The algorithm employed a greedy search, where both feasible or unfeasible solutions are accepted in the current population. The experimental part showed its good behavior on a set of instances of the literature.

Ozsoydan *et al.* [8] proposed a hybrid binary swarm optimization, where two phases are considered. The first phase applies a standard swarm optimization to explore local neighborhoods, and the second phase applied the genetic algorithm to diversify the search space.

Wu and He [9] proposed a hybrid (double) Jaya algorithm, where the method was reenforced with intensification and diversification operators. It combines both the standard Jaya procedure and the differential evolution operator. Crossover and repairing operators were also added. The resulting method was evaluated on instances of the literature.

Wei and Hao [10] designed an iterated local search algorithm based on two phases: exploration phase and escaping phase. The first phase intensifies the search using variable neighborhood descend and tabu list. While the escaping phase tries to diversify the search. They evaluated their proposed method on instances of the literature.

Dahmani *et al.* [11] designed an iterative rounding strategy-based algorithm for tackling the same problem. It is based on three features: a partial solution built according to a fractional value related the solution of a given linear relaxation, a rounding procedure coupled with an improved operator using the critical element, and degrading and repairing operators added for a diversification. The efficiency of that method was evaluated on instances of the literature.

Dahmani *et al.* [12] investigated the use of a particle swarm optimization. A standard version was reinforced with a local search operator, where the notion of the critical item for a single knapsack was considered. Their method was evaluated on benchmark instances of the literature.

In this paper, we propose a local branching employed as learning strategy in the population-based algorithm, where three key features are considered:

1) An initial population is built using a randomized procedure; that is based upon Arulselvan's rule,
2) According to a new position of a given particle, local operators are added for enhancing its position; where that position is measured according to its value.
3) A local branching operator is applied for enhancing the position of the global best position (with better bound); that is employed for searching to good tuning and so, enhancing positions which must be used as a series of particles belonging to the current population.

These strategies are embedded into an iterated process till satisfying a stopping criterion. The proposed algorithm is then analyzed computationally on a set of benchmark instances of the literature, where its provided results are compared to the best bounds available in the literature.

III. A Learning Strategy for Highlighting Approximate Solutions of PSUK

A. A Basic Swarm Optimization

In earlier works, it has been remarked that the first bounds related to solutions may decrease quickly at the beginning of the resolution while a stagnation appears at the end of the resolution. These methods are often based-upon a single solution procedure for which any diversification may tend toward a solution not evident to use for optimizing the system. In order to enrich the search process, we propose a population-based method; that is based upon Particle Swarm Optimization (PSO). That method can be viewed as an evolutionary algorithm, which has been proposed first by Eberhart and Kennedy [13]. The used process considers a set of agents/particles described by the following three vectors (at a certain iteration $t \geq 1$):

- θ_i^t: the current position of particle i on the space.
- p_{Best}: the best position reached by the particle i.
- g_{Best}: the global best solution over all particles.
- v_i^t: the velocity applied for guiding the moves of the current particle *i* in the space.

Further, information shared between particles are updated for providing new positions:

$$v_i^t = \omega \times v_i^{t-1} + c_1 \times v \times (p_{Best} - \theta_i^{t-1}) + c_2 \times \nu \times (g_{Best} - \theta_i^{t-1}) \quad (3)$$

and

$$\theta_i^t = \theta_i^{t-1} + v_i^t \quad (4)$$

where Eq. (3) acts on the velocity and Eq. (4) acts on each particle's position.

One can observe that Eq. (3) is composed of three parts:

(i) $\omega \times v_i^{t-1}$: represents the *i*-th particle's velocity at iteration $(t-1)$, where ω denotes the inertia weight that tries to control the magnitude of the "old velocity" (usually ω belongs to the interval [0.4, 0.9]).

(ii) $(p_{Best} - \theta_i^{t-1})$: denotes a natural tendency of a particle *i* to return to its best position.

(iii) $(g_{Best} - \theta_i^{t-1})$: is the tendency of a particle to follow the best position.

Both parameters c_1 and c_2, represent the cognitive and social factors, respectively, where often $c_1 + c_2$ is less than or equal to 4, and both parameters v and ν are

randomly generated in the interval [0, 1]; they are used to determine the degree of influence of P_{Best} and g_{Best}, respectively.

B. Adaptation of the Swarm Optimization

A P_{SUK}'s feasible solution is such that all selected items and elements satisfies all constraints of the problem. Thus, each solution/configuration, namely $\vec{S}$, can be stated as follows: a vector of dimension $m + n + 1$ such that $S = (z, h)$, where $h = \big((x_1, x_2, \dots, x_n), (y_1, y_2, \dots, y_m)\big)$ which denotes the current positions of all decision variables of a solution / configuration $\vec{S}$ with its objective value z.

1) A standard population-based method

Because we are looking to feasible solutions, the proposed method uses a standard normalization of the configuration, where a Sigmoid function is used for splitting each decision variable into either 1 or 0. The normalized configuration, the repair operator is called for removing unfeasibility, and augmenting the quality of the solution for making it complete. The repair operator works as follows:

1. Let S be the current configuration.
2. Remove items violating the knapsack capacity (and their related elements) for making the solution feasible, and let $S_{(par)}$ be the provided partial solution.

Let $P_{SUK}^{(red)}$ be the new reduced subproblem containing items and elements removed from *S* and items fixed to zero in *S* with its associated elements (when all items related to these elements are free or fixed to 0).

3. Let $S_{(red)}$ be the (sub)optimal solution reached when solving $P_{SUK}^{(red)}$ (cf. the procedure below).
4. Combine both $S_{(par)}$ and $S_{(red)}$ to achieve a new complete solution.

When providing the reduced problem $P_{SUK}^{(red)}$, the following greedy procedure is applied in step 3:

1. Set $\mathrm{P_{rest}} = \mathrm{P}_{(SUK)}^{(red)}$.
2. For P_{rest}, rank all items in decreasing order of their profit per weight, i.e., $\frac{p_1}{w_1} \geq \frac{p_2}{w_2} \geq \cdots$, and set $\bar{b} = b$, $\bar{I} = I$, and $S = \emptyset$.
3. $\forall i \in \bar{I}$, if $w_i^x > \bar{b}$, remove these items from $\bar{I}$.
4. If $\bar{I} \neq \emptyset$, then *stop* with the best solution *S*.

Let $\rho = argmax_{i \in \bar{I}}\{\frac{p_i}{w_i^x}\}$:

a) Set $S = S \cup \{\rho\}$, $\bar{I} = \bar{I} \setminus \{\rho\}$ and $\bar{b} = \bar{b} - w_i^x$.
b) Evaluate all *potential weights* of all items of $\bar{I}$.
c) If $\bar{I} \neq$, rank all items of $\bar{I}$ in decreasing order of their profit per weight.

5. Repeat steps (3) and (4).

Thus, the second partial solution reached is added to the first partial one for forming the complete solution for PSUK; that is a new position of the current particle.

2) A learning strategy

Often learning strategy is employed for enhancing parameters of a given model, where forward and backtracking procedures are used. Instead of using these processes, we introduce a deep exploration aiming at improving the position of the feed according to g_{Best} position. The goal is to drive all particles, by their local positions, to this position using a jumping principle that can sometimes be more ambitious. Obviously, this exploration can be done by simple local searches, but it has often been observed that such procedures can easily get trapped in local optima. In order to better explore the search subspaces, we propose a learning strategy by applying a series of local branching.

a) Local branching (LB): It is a specialized optimization strategy considered as an alternative to exact methods for solving complex problems. LB has been first proposed by Fischetti *et al.* [14], where its principle is to mimic an optimal resolution of mixed integer programming problems (P_{MIP}). Such a technic has been successfully used for tackling several optimization problems (Hifi *et al.* [15], and Boukhari *et al.* [16]). Given $\bar{x}$ as a reference solution for P_{MIP}, B the set of its binary variables, and $k \in N^{\star}$ be the k_{opt} neighborhood related to $\bar{x}$ which corresponds to the set of solutions related to P_{MIP} with the following additional branching:

$$\Delta(x, \bar{x}) \coloneqq \sum_{j \in S} \big(1 - x_j\big) + \sum_{j \in \mathrm{B} \setminus S} x_j \leq k \qquad (5)$$

such that $S = \{j \in B \mid \bar{x}_j = 1\}$ and the term of the right-side of Ineq (5) is the number of binary variables switching their values either from 1 to 0, and vice versa. The additional local constraint may be rewritten as follows:

$$\Delta(x, \bar{x}) \coloneqq \sum_{j \in S} \big(1 - x_j\big) \leq k' \qquad (6)$$

such that $k' = \frac{k}{2}$, where adding the above constraint to the current problem induces a branching criterion within an enumerative scheme for the current P_{MIP}. Therefore, the feasible space related to the current branching node can be divided according to the following two adding constraints:

$$\Delta(x, \bar{x}) \leq k \qquad \text{or} \qquad \Delta(x, \bar{x}) \geq k + 1 \qquad (7)$$

where *k* is often determined experimentally. According to the value assigned to *k*, the search process may be iterated by altering between normal and local branches.

b) Highlighting solutions with LB: Of course, calling a specialized black-box solver, like Cplex, for solving the complete problem can induce a very expensive resolution, even if the instance's size is relatively small. In order to avoid a slow convergence, we propose to build a reduced problem by splitting the components of the current solution s into two parts. We do it as follows:

- Fix α%, α ∈]0, 100[×|s|, of the variables of s belonging to the knapsack (we note the resulting fixed solution as x_α with its set of indices ξ, and $z_1(x_\alpha)$ its value).
- Gather the $(|s| - \alpha)$ unfixed items with the $(n + \alpha - |s|)$ (fixed to 0 in s) to form a reduced subproblem, noted P_{red}. Let m' be the number of resulting items:

$$\mathrm{z}_2(\bar{s}) = \max \sum_{i=1}^{m'} p_i \, \bar{s}_i \qquad (8)$$

$$\text{s.t.} \quad \mathrm{W}(A_s) = \sum_{j \in \cup_{i \in A_s} \bar{U}_i} w_j \leq \bar{b} \qquad (9)$$

where $\bar{b} = \mathrm{b} - \sum_{i \in \xi} w_j$. Solve P_{red} with LB and let $z_2(\bar{s})$ be the value related to the complementary solution $\bar{s}$.

- Set $\underline{s} = s \cup \bar{s}$, and $\underline{z} = z_1(x_\alpha) + z_2(\bar{s})$.

c) An overview of the proposed method: Algo 1 describes the main steps of the population-based algorithm using the local branching as learning strategy.

Algorithm 1 − Local branching as a learning strategy

Input. A population $\mathcal{P}$ of solutions.
Output. A best particle with objective value g_{Best} .

1: **Initialization.**
2: Generating the initial population with positions $\vec{\mathrm{h}}^{\mathrm{i}}, \mathrm{i} \in \mathcal{P}$.
3: Normalize all positions and make them feasible.
4: $\forall i \in \mathcal{P}$, set $p^i_{Best} = \overrightarrow{X^i}$,, and $g_{Best} = \mathrm{argmax}_{i \in \mathcal{P}}\{\mathrm{z}(\mathrm{p}^i_{Best})\}$
5: Generate $\overrightarrow{V^i}$ in $[-r_i, r_i]$ for each (x_i) and from $[-r_j, r_j]$ for (y_j).
6: **Iterative**
7: **while** (the stopping conditions are not performed) **do**
8: **for** (each particle $i \in \mathcal{P}$) **do**
9: Compute its objective value $\mathrm{z}(\vec{\mathrm{h}}^{\mathrm{i}})$.
10: **if (** $\mathrm{z}(\vec{h^i}) > \mathrm{z}(\overrightarrow{p^i_{Best}})$ **) then** Set $p^i_{Best} = \vec{h^i}$,
12: **end if**
13: **end for**
14: **if** $(z(g_{Best}) < max_{i \in \mathcal{P}}\{z(p^i_{Best})\})$ **then**
15: Update g_{Best} and call LB operator for enhancing g_{Best}.
16: **end if**
17: **for** (each particle $i \in \mathcal{P}$) **do**
18: Update the particle's velocity by using Eq. (3),
19 Update the particle's position by using Eq. (4).
20 **end for**
21: **end while**
22: **Return** the best positiong_{Best} whose value is $z(g_{Best})$.

IV. Experimental Part

In this section, the behavior of the proposed Learning Strategy-Based Population Algorithm (LS-BPA) is analyzed on benchmark instances of the literature. The used instances are composed of two groups: the first group contains medium instances and the second one includes large-scale instances. The number of items (elements) varies from 100 to 300 for the first group, and from 600 to 1000 for the second group (an Intel Pentium Core i5 with 2.4GHz was used).

TABLE I. Behavior of LB-PSO: Varying the Size *K* of the Neighborhood

#inst	k=4	k=5	k=6	k=7	k=8
n<m	11343.4	11428	11364.8	11360.1	11398.9
n=m	11797.0	12066.0	12052.0	12034.5	12047.2
n>m	11988.8	11989.6	12012.9	11930.8	12018.1
Av.	11709.7	11827.9	11809.9	11775.1	11821.4

A. Parameter Settings

LB-PSO needs to tune (i) the fixation parameter α used for providing a partial solution, and (ii) the size k of the neighborhood needed by the local branching. We also considered the stopping criteria as a maximum runtime limit fixed to 1 hour (even augmenting that limit, bounds were not improved), and the size of population to 20 (other values were also tested for which the bounds may be improved at the expense of runtime).

Table I reports the average bounds achieved by LB-PSO on instances of Group 1, where k varies in $\{4, \ldots, 8\}$. Column 1 refers to the three subsets of instances, columns from 2 to 10 show the average objective values (ten instances by subset) and the last line (Average) tallies the global average value overall the thirty instances of the three subsets. From Table I, we observe that the best global average value is reached for $k = 5$ (column 5, last line), which is a break-point overall other values. We then fix that value for the rest of the study.

Table II displays the average bounds achieved by LB-PSO, where α belongs to {20%, 70%} by a step of 10%. Column 1 refers to the subsets, columns from 2 to 8 report the average objective value of each subset, and the last line tallies the global average value overall subsets. From Table II, we observe that the break-point is reached for α = 60% (it reaches a global average objective value of 11978.6) even if for some values of α the algorithm provides a close value. Hence, α is fixed to 60%.

TABLE II. Behavior of LB-PSO: Variation the Fixation Parameter for LB

#inst	20%	30%	40%	50%	60%	70%
n<m	11575.1	11542.6	11575.1	11537.3	11575.1	11537.3
n=m	12141.3	12141.3	12114.9	12131.2	12141.3	12141.3
n>m	12180.4	12163.7	12167.6	12210.8	12219.4	12186.9
Av.	11965.6	11949.2	11952.5	11959.8	**11978.6**	11955.2

B. LB-PSO versus more Recent Methods: Group 1

We compared LB-PSO' results to those achieved by more recent methods of in the literature: DHJaya(1,2) [9], and IRSP and HSPO-BA ([11], and [12]). Table III reports the results provided by BABC [9], DHJaya(1,2), I2PLS [10], HSPO-BA, and LB-PSO on instances of Group 1.

The first column displays the instance's information, and other columns report the best objective values (Best) achieved by all considered algorithms. From Table III, we observe that LB-PSO outperforms BABC, since it is able to provide 21 better bounds and matches 9 bounds, while BABC matches 9 bounds and it fails for the rest of the instances. LB-PSO performs better than DHJaya(1,2). In this case, it reaches 22 better bounds over the 30 instances tested. In term f percentage, PSO-B achieves 70% of the best bounds and it matches the rest of the bounds. LB-PSO has a good behavior when compared to I2PLS. Indeed, it matches 29 bounds as I2PLS and reaches a new (lower) bound for the instance S-I-5. Finally, LB-PSO remains competitive when comparing its provided results to those achieved by the best method of the literature HPSO-BA.

Indeed, one the one hand, the learning strategy is capable to enhance solutions, where a new (lower) bound (instance S-I-5) is discovered, and it matches 29 bounds. On the other hand, LB-PSO's global average bound is equal to 11991.7 while HPSO-BA provides an average value of 11973.60.

TABLE III. BEHAVIOR OF LB-PSO ON 30 INSTANCES

#Inst.	BABC Best	DHJaya(1,2) Best	I2PLS Best	HSPO-BA Best	LB-PSO Best
S-I-1	12045	12045	12045	12045	12045
S-I-2	12369	12369	12369	12369	12369
S-I-3	13647	13696	13696	13696	13696
S-I-4	10926	11298	11298	11298	11298
S-I-5	11374	11568	11568	11568	**11894**
S-I-6	10822	11714	11802	11802	11802
S-I-7	10110	10483	10600	10600	10600
S-I-8	9659	10302	10506	10506	10506
S-I-9	10835	11036	11321	11321	11321
S-I-10	9380	10104	10220	10220	10220
S-II-1	14044	14044	14044	14044	14044
S-II-2	13508	13508	13508	13508	13508
S-II-3	12350	12522	12522	12522	12522
S-II-4	11929	12317	12317	12317	12317
S-II-5	12304	12736	12817	12817	12817
S-II-6	10857	11425	11585	11585	11585
S-II-7	10869	11569	11665	11665	11665
S-II-8	10048	10927	11325	11325	11325
S-II-9	10755	10943	11249	11249	11249
S-II-10	9601	10214	10381	10381	10381
S-III-1	13283	13283	13283	13283	13283
S-III-2	12479	12479	12479	12479	12479
S-III-3	13402	13521	13521	13521	13521
S-III-4	14215	14215	14215	14215	14215
S-III-5	10572	11385	11563	11563	11563
S-III-6	12245	12402	12607	12607	12607
S-III-7	11021	11484	11484	11484	11484
S-III-8	9649	10710	11209	11209	11209
S-III-9	10927	11722	11771	11771	11771
S-III-1	9306	10194	10238	10238	10238
Average	11484.4	11873.8	11973.6	11973.6	**11984.5**
"="	7	13	29	29	**30**
"↓"	23	17	1	1	0

C. Behavior of LS-BPA on Instances of Group 2

LB-PSO was also tested on 18 large-scale instances belonging to Group 2. Table IV reports the results achieved by I2PLS, IRSP, HSPO-BA and LB-PSO. Column 1 displays the instance's label, column 2 report I2PLS' results (Best), and columns from 2 to 5 tally those achieved by I2PLS, IRSP, HSPSO-BA and the proposed LB-PSO.

From Table IV, we can observe what follows:

- LB-PSO performs better than all tested methods for the large-scale instances, since it is able to provide the 18 best bounds.
- LB-PSO achieves a global average bound of 13675.2, which remain better than that provided by IRSP (13642.7). In this case the Gap realized is equal to 32.5, while that Gap becomes more significant when comparing it to HSPO-BA (107.9), and I2PLS (498.3).
- Over all the 18 best bounds matched by LB-PSO, it is capable to discover 11 new solutions, which represent a percentage of 61% of the bounds.

TABLE IV. BEHAVIOR OF I2PLS, IRSP, HSPO-BA AND LB-PSO ON THE 18 LARGE - SCALE INSTANCES

#Inst #Inst.	I2PLS Best	IRSP Best	HSPO-BA Best	LB-PSO Best
S-IV-1	12774	13114	**13170**	**13170**
S-IV-2	7565	7962	7647	**7995**
S-IV-3	15464	15917	15760	**15923**
S-IV-4	9365	9423	9423	**9507**
S-IV-5	19597	**20221**	**20221**	**20221**
S-IV-6	11698	11730	11698	**11732**
S-IV-7	13841	14113	**14211**	**14211**
S-IV-8	7142	7759	7759	**7772**
S-IV-9	17175	17250	17232	**17289**
S-IV-10	8939	9385	9230	**9388**
S-IV-11	21748	22049	21601	**22204**
S-IV-12	11322	11696	11696	**11704**
S-IV-13	13903	14282	**14296**	**14296**
S-IV-14	7647	**8125**	**8125**	**8125**
S-IV-15	16912	17767	17541	**17775**
S-IV-16	8935	9856	**9917**	**9917**
S-IV-17	21600	22590	22355	**22595**
S-IV-18	11557	**12330**	**12330**	**12330**
Average	13176.9	13642.7	13567.3	**13675.2**
"="	0	3	7	**18**
"↓"	18	15	11	0

V. CONCLUSION

In this paper, the set-union knapsack problem was studied, where a cooperative population-based algorithm was proposed for approximately solve it. The resulting method combined three features: an iterative swarm optimization process, local search operators, and a learning strategy that is based upon local branching operator. Finally, the performance of the resulting method was evaluated on a set of benchmark instances of the literature, which contains instances varying from medium to large-scale ones. The experimental part showed that the method is very competitive, where it is able to outperform more recent of the literature and is able to discover new lower bounds for several instances.

CONFLICT OF INTEREST

The authors declare that they have no conflict of interest.

AUTHOR CONTRIBUTION

Mhand Hifi proposed the main idea related to this work. He designed the local branching as a learning strategy for highlighting the evolutionary algorithm. Meriem Ferroum coded the first version of the algorithm which is based upon the above idea. Meriem Ferroum and Isma Dahmani tested the final version of the code on a set of benchmark instances of the literature. Mhand Hifi investigated the sensitivity analysis, extended the experimental part with a statistical analysis, and provided the final version of the paper.

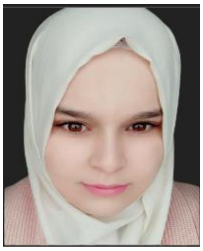
REFERENCES

[1] R. Merkle and M. Hellman, "Hiding information and signatures in trapdoor knapsacks," *IEEE Transactions on Information Theory*, vol. 24, no. 5, pp. 525-530, 1978.

[2] G. Perboli, L. Gobbato, and F. Perfetti, "Packing problems in transportation and supply chain: New problems and trends," *Procedia-Social and Behavioral Sciences*, vol. 111, pp. 672-681. 2014.

[3] A. Grange, I. Kacem, and S. Martin, "Algorithms for the bin packing problem with overlapping items," *Computers and Industrial Engineering*, vol. 115, pp. 331-341, 2019.
[4] M Hifi and L. Yousef, "A local search-based method for sphere packing problems," *European Journal of Operational Researc*h, vol. 274, no. 2, pp. 482-500, 2019.
[5] O. Goldschmidt, D. Nehme, and G. Yu, "Note: On the set-union knapsack problem," *Naval Research Logistics*, vol. 41, no. 6, pp. 833-842, 1994.
[6] A. Arulselvan, "A note on the set union knapsack problem," *Discrete Applied Mathematics*, vol. 169, pp. 214-218, 2014.
[7] Y. He, H. Xie, T. L. Wong, and X. Wang, "A novel binary artificial bee colony algorithm for the set-union knapsack problem," *Future Generation Computer Systems*, vol. 78, pp. 77-86, 2018.
[8] F. Ozsoydan and A. Baykasoglu, "A swarm intelligence-based algorithm for the set-union knapsack problem," *Future Generation Computer Systems*, vol. 93, pp. 560-569, 2019.
[9] C. Wu and Y. He, "Solving the set-union knapsack problem by a novel hybrid Jaya algorithm," *Soft Computing*, vol. 24, no. 3, pp. 1883-1902, 2020.
[10] Z. Wei and J. Hao, "Iterated two-phase local search for the set-union knapsack problem," *Future Generation Computer Systems*, vol. 101, pp. 1005-1017, 2019.
[11] I. Dahmani, M. Ferroum, and M. Hifi, "An iterative rounding strategy-based algorithm for the set-union knapsack problem," *Soft Computing*, vol. 25, pp. 13617-13639, 2021.
[12] I. Dahmani, M. Hifi, T. Saadi, and L. Yousef, "A swarm optimization-based search algorithm for the quadratic knapsack problem with conflict graphs," *Expert Systems with Applications*, vol. 148, p. 113224, 2020.
[13] R. Eberhart and J. Kennedy, "A new optimizer using particle swarm theory," in *Proc. the Sixth International Symposium on Micro Machine and Human Science*, 1995, pp. 39-43.
[14] M. Fischetti and A. Lodi, "Local branching," *Mathematical Programming*, vol. 98, no. 1, pp. 23-47, 2003.
[15] M. Hifi, S. Negre, and M. O. A. Mounir, "Local branching- based algorithm for the disjunctively constrained knapsack problem," in *Proc. International Conference on Computers Industrial Engineering*, 2009, pp. 279-284.
[16] S. Boukhari, I. Dahmani, and M. Hifi, "Local branching strategy-based method for the knapsack problem with setup," in *Proc. 4th International Conference on Artificial Intelligence, Soft Computing and Applications*, 2020, pp. 65-75.

Mhand Hifi is a full professor of Computer Science and Operations Research at UPJV (Université de Picardie Jules Verne), France. He is the head of the laboratory EPROAD of the UPJV. He is area editor and academic editor for several international journals: CAIE, Advances in OR, AJIBM, etc. His research interest is NP Hard combinatorial optimization (sequential and parallel optimization) applied to logistic and OR problems.



Meriem Ferroum is a PhD student at the USTHB (Université des Sciences et de la Technologie Houari-Boumedienne), Algeria. She received her BS in 2012 at the USTHB, and got her MS in 2017 from the same University.

Isma Dahmani is an associate professor of Operations Research at USTHB (Université des Sciences et de la Technologie Houari-Boumedienne), Algeria. Her area of interest is the resolution of combinatorial optimization problems using approximate, hybrid and exact methods.