A Hybrid Evolutionary Algorithm for the Sequencing *m*-Vector Bin Packing Problem

Méziane Aïder, Amina N. Benahmed, and Isma Dahmani
University of Sciences and Technology Houari Boumediene, Algeria
Email: m-aider@usthb.dz, {aminabenahmd, dahmani.isma}@gmail.com

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

***Abstract*—In this paper, the product sequencing decisions in multiple-piece-flow assembly lines problem is approximately solved with a hybrid evolutionary algorithm. The product sequencing decisions in multiple-piece-flow assembly lines, known as the sequencing *m*-vector bin packing problem, occurs in manufacturing organization and because of its NP-hardness it is however computationally challenging. The designed method combines a population approach and both first fit bin packing procedure coupled with a repairing operator: the population approach tries to maintain the diversity of a series of populations reached throughout an iterative procedure while the added operators try to highlight the quality of the solutions throughout the search process. The performance of the proposed method is evaluated on a set of benchmark instances taken from the literature. The results provided by the method are compared to those reached by recent published methods and to those reached by the state-of-the-art Cplex solver. The preliminary experimental part showed that the designed method outperforms the other ones by discovering new bounds for most of considered instances.**

***Index Terms*—evolutionary, intensification, optimization, packing, sequencing**

I. Introduction

The packing problem can be viewed as one of the old problems of combinatorial optimization problems, which belongs to the well-known Cutting and Packing (CP) family (Wascher *et al.* [1]). Such a problem appears in several real-world applications, like logistics (Perboli *et al.* [2]), manufacturing, production process, automated planning and other problems related to the decision making. The manufacture of customized products at low cost (Battaïa *et al.* [3]) contains several problems linking scheduling and packing problems (Aïder *et al.* [4]): the multiple-piece-flow assembly lines problem is one of that problems in which a group of workpieces are launched in sets on the paced assembly line to be processed at sequentially arranged stations. All parts of a set enter and leave each station simultaneously.

Manuscript received December 11, 2021; revised May 22, 2022.

In this paper, we focus on approximately solving the problem of the *m*-Vector Bin Packing with Sequencing (*m*-VBPS). An instance of the problem can be defined as follows: at each station and for a given period, a set of workpieces is available; that is a fixed cycle time. After fixed cycle time, the workpieces are transferred to the neighbor (next) station such that another set of workpieces is performed at the former station. This set of possible launch times are denoted $1, \dots, \beta$, where β represents the number of potential launch times. Further, each workpiece j is characterized with its due date d_j that is the latest launch time allowing the workpiece to be finished in time, and its release time r_j, denoting the earliest launch time when the product order is fixed and the required material supply is secured. In this case, the set of dimensions is $1, \dots, m$ and the size of the workpiece j along dimension $i \in 1, \dots, m$ is equal to time v_{ji}. Along each dimension, the total size of the workpieces launched together in a set is restricted. The objective of the problem is to find the solution x such that:

- Each workpiece is performed once and not earlier than its release time, i.e., $x_j \geq r_j$.
- The total size related to the workpieces along each dimension is less than or equal to the cycle time, i.e., $\sum_{\{j \mid x_j = b\}} v_{ji} \leq \text{cycle time}, \forall b \in \{1, \dots, \beta\}$ and $\forall i \in \{1, \dots, m\}$.
- The function $z(x) = \sum_{\forall j} w_j x_j$ should be minimized.

Due to the complexity of the majority of the problems belonging to CP family, there are several available papers tackling some part of these problems in the literature. Several problems have been tackled with exact and approximate solution procedures: for these problems it is often important to be able to get optimal solutions or to achieve solutions with high-qualities.

In recent years, it has been observed that the hybridization of solution procedures is a very promising research issue. Nevertheless, the difficulty that can be encountered with these methods may be related to the runtime consumed, which in some cases can exponentially grow, especially when solving large-sized problem instances. Herein, we propose an efficient and powerful

doi: 10.12720/jait.13.4.306-311

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. In this work, we opt in including a drop and rebuild operator into a population-based method for highlighting the quality of the solutions of *m*-VBPS; that is applied as a learning strategy for enriching the diversified solutions a series of local best solutions and a global best solution considered as a driven solution at a certain iteration of the method. The designed method is composed of the following features:

- A random search of positions for generating the first population of solutions, where a normalized strategy is applied for determining a sequence linking items and stations.
- A perturbation phase using a flexible swapping between positions of some workpieces.

The steps mentioned above are embedded into an iterative process for highlighting solutions belonging to the current population.

The rest of the paper is organized as follows. Section II-A presents a correct formal description of the problem studied. Section III describes the designed evolutionary algorithm to approximately solve the *m*-VBPS. More precisely, a basic representation of a solution is given in Section III-A, the main principle of particles swarm optimization is summarized in Section III-B, and the intensification operators are discussed in Section III-C. Finally, a preliminary experimental study is considered in Section IV, where the behavior of the method is analyzed on a set of benchmark instances extracted from the literature, and its achieved bounds are compared to the best bounds available in the literature and to those provided by the state-of-the-art Cplex solver.

II. Solving the m-Vector Bin Packing and Sequencing Problem

We first describe a correct formal description of the *m*-vector bin packing and sequencing problem. Next, we discuss the basic steps of the Particle Swarm Optimization (PSO) as an Evolutionary Algorithm (EA). Finally, we present a Hybrid of EA (noted HEA) for the problem studied and how highlighting the quality of the solutions by incorporating two move operators.

A. The Model

The product sequencing decisions in multiple-piece-flow assembly lines (Otto and Li [5]), can be viewed as the *m*-vector bin packing and sequencing problem (*m*-VBPS) such that a set *J* of workpieces are placed on sets on the assembly line to be processed sequentially according to the positioned stations. Each station *i* is characterized by its cycle time *c* such that a set of workpieces is available during this time. In this case, each workpiece $j \in J$ corresponds to a specific customer order, it has a specific due date d_j and a specific release time r_j which is the earliest possible launch time when the specification of the product order is fixed and the required material supply is secured. The set of dimensions (namely stations) is denoted by $i \in I = \{1, \dots, m\}$ and the size of each workpiece $j \in J$ along the dimension $i \in I$ is represented by v_{ji}. Let $B = \{1, \dots, \beta\}$ be the set of possible launch times, where β denotes the number of potential launch times. The cost parameters w_{jb}, where $w_{jb} = \max\{b - d_j, 0\}$, is introduced if workpiece *j* is performed at time $b \in B$.

Let x_{jb} equal to 1 if job (workpiece or item) j is assigned to bin (station or knapsack) b, 0 otherwise. Thus, a formal description of a mixed-integer programming related to m-VBPS (noted P_{m-VBPS}) can be stated as follows:

$$min\ z(x) = \sum_{j \in J} \sum_{b=r_j, b>0}^{\beta} w_{jb} x_{jb} \quad (1)$$

$$Subject\ to \sum_{b=r_j, b>0}^{\beta} x_{jb} = 1, \forall j \in J \quad (2)$$

$$\sum_{j \in J, r_j \leq b} v_{ji} x_{jb} \leq c, \forall i \in I, \forall b \in B \quad (3)$$

$$x_{jb} \in \{0,1\}, \forall j \in J, \forall b \in B \quad (4)$$

where the objective function (1) should be minimized. Constraints (2) ensure that each job j has to be assigned to one bin, and not earlier than its release time r_j. Inequalities (3) indicate that the capacity constraints, where each bin along each dimension must be satisfied. Finally, constraints (4) are related to the domain of all decision variables.

III. A Hybrid Algorithm for P_{m-VBPS}

This section is divided into three parts. The first part (Section III-A) describes the standard population-based method. The last part (Section III-B) describes the main steps of the proposed hybrid evolutionary algorithm.

A. A Basic Population-Based Method

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 [6]. 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 the particle *i* on the space.
- P_{Best}: the best position reached by the particle *i*. Thus, the best position allows us to compute the lower bound related to the length of the container.
- $\vec{v}_i^t$: 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:

$$v_i^t = w \times v_i^{t-1} + c_1 \times v \times (p_{Best} - \Theta_i^{t-1}) + c_2 \times v \times (g_{Best} - \Theta_i^{t-1}) \quad (5)$$

and

$$\Theta_i^t = \Theta_i^{t-1} + \mathrm{v}_i^t \quad (6)$$

where Eq. (5) acts on the velocity and Eq. (6) acts on each particle's position.

One can observe that Eq. (5) is composed of three parts:

1) $\omega \times v_i^{t-1}$: represents the I 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]).
2) $(p_{Best} - \Theta_i^{t-1})$: denotes a natural tendency of a particle to return to its best position.
3) $(g_{Best} - \Theta_i^{t-1})$: 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 and v 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

1) Configuration: A $\mathrm{P}_{m-\mathrm{VBPS}}$'s feasible solution is such that all items (workpieces) are assigned once to a knapsack (bin or station). Thus, each solution, namely $\vec{S}$, can be stated as follows: a vector of dimension $m \times (n+1)$ such that $S = (z, x)$, such that:

$$x = \left((x_{11}, x_{21}, \dots, x_{m1}), \dots, (x_{1\beta}, x_{2\beta}, \dots, x_{m\beta})\right)$$

which denotes the current positions of all decision variables of a solution $\vec{S}$ with objective value *z*.

2) A starting population: Herein, a staring population is generated by using the so-called *first fit bin packing procedure*, where a random selection is introduced. Such a choice is used for generating different solutions. Thus, the following steps are used for generating that population of solutions.

Set Iter $= 0$ and Pop $= \emptyset$ (related to the population size with a Max_Size_Pop)
Repeat
Set Iter $=$ Iter $+ 1$;
Set $\bar{J} = J$, and $\bar{\bar{J}} = \emptyset$;
Do
If $\bar{J} \neq \emptyset$, then let $i \in J$, be a random item.
i) Search for an open bin *b* and try to affect the current item.
ii) If *i* is not assigned to *b*, move *i* from $\bar{J}$ to $\bar{\bar{J}}$.
While $\bar{J} \neq \emptyset$.
If $\bar{\bar{J}} \neq \emptyset$, then set $\bar{J} = \bar{\bar{J}}$, open a new bin *b*, and repeat step ii).
Update Pop with the new solution.
Until (Iter $=$ Max_Size_Pop);

Of course, only different solutions are accepted in the population, where the update (the last line) is slightly modified for receiving only these solutions.

C. A Standard Population-Based Method

Because we are looking to feasible solutions, the proposed method uses a standard normalization of the configuration at hand, where the Sigmoid function is considered for splitting all decisions variables into either 1 or 0. According to the used normalized solution, 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 for making the solution feasible, let $S_{(par)}$ be the provided partial feasible solution, and $\bar{I}$ be the set of the removed items.
   Let $P_{m-VBPS}^{(red)}$ be the new reduced subproblem containing all items removed from *S* and the items fixed to zero in *S*.
3) Let $S_{(red)}$ be the (sub)optimal solution reached when solving the reduced problem $P_{m-VBPS}^{(red)}$ (as described below).
4) Combine both $S_{(par)}$ and $S_{(red)}$ to provide a new complete solution.

Procedure Complete (**PC**): When providing the reduced problem $P_{m-VBPS}^{(red)}$, the following greedy knapsack procedure is applied in step 3.

1) Set $P_{rest} = P_{m-VBPS}^{(red)}$.
2) For P_{rest} and for all available bins, rank all items (those of $\bar{I}$) in non-decreasing order of their cost per size.
3) For an item $i \in \bar{I}$, search for the bin $b \in B$, such that $v_{ib} \leq \bar{c}$.
   If *b* exists, then assign *i* to *b*, open a new bin otherwise; remove *i* from $\bar{I}$, increment the cardinality of *B*, and assign *i* to the new bin.
4) If $\bar{I} = \emptyset$, then **stop** with the best solution *S*.
5) Repeat steps iii) and iv).

Thus, the second partial solution reached by the above procedure is added to the first partial solution for forming the complete solution for $\mathrm{P}_{m-\mathrm{VBPS}}$; that is a new position of the current particle.

Highlighting the search process: Often the 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 massive exploration aiming at improving the position of the feed according to the $Local_{Best}$ positions. The goal is to drive all particles in an aggressive manner while reducing the size of the population. Indeed, instead of employing a large size for the population, we just fixe its size to 10, and to update a half of the population for diversifying the search process. The population is renewed every 200 iterations. In order to better explore the search subspaces, we also propose to add two operators.

a) First move: It is a specialized strategy considered as an alternative to the 2-opt operator (Croes [7], and Aïder *et al.* [8]). It works as follows:

1) Let *i* be a random item generated from the set *J*.
2) Move *i* from the current bin, namely b_1, to another bin $b_2 \in B \setminus \{b_1\}$.
3) Correct the constraints related to the violated bins (i.e., make a random removing of items), and call Procedure **PC** to enrich a complete solution.
4) Exit with the best solution $\hat{S}$.

b) Second move: It can be viewed as a swapping operator where two different items belonging to two different bins are swapped (Dahmani *et al.* [9]). In this case a more general version may be provided using two loops linking a series of couples of items. Because such a processes is much consuming, we then apply the following random two-exchange operator, which works as follows:

1) Let (i_1, i_2), $i_1 \neq i_2$, be a couple of items randomly generated from J, and b_1 and b_2 be their bins, respectively.
2) Assign i_1 to b_2, and i_2 to b_1.
3) Correct the constraints related to the violated bins (i.e., make a random removing of items), and call Procedure **PC** to enrich a complete solution.
4) Exit with the best solution $\hat{S}$.

IV. Preliminary Experimental Part

In this section, a preliminary experimental part is given to assess the performance of the Hybrid Evolutionary Algorithm (HEA) by comparing its achieved results to the best solutions available in the literature (all methods were coded in C++ and performed on a computer with an Intel Pentium Core i5 with 2.10 GHz). A subset of instances used as benchmarks represents the instances extracted from [3], where twenty instances noted from VBP-n100-d5-10-30-LDNR01 to VBP-n100-d5-10-30-LD-NR20. In this case, the number of items is fixed to 100, the number of stations is equal to 10, and the capacity of each bin is set equal to 30. We then compare the results achieved by the proposed method HEA to those provided by the best method available in the literature: Iterative Variable Neighborhood Heuristic (IVNH) designed in Otto and Li [3], and the state-of-the-art Cplex solver.

The computational study was conducted by fixing the population size to 10, the half of the population was renewed after 200 iterations, and the final runtime was fixed to ten minutes (as used in Otto and Li [3]). Because the Cplex solver is a specialized exact method, we then fixed its runtime to 1 hour. As used in Otto and Li [3], the average absolute performance is considered which denotes the average difference in the absolute bound of the algorithm's solution and the best bound reached. Because IVNHs' solutions are not available, we then used a comparative study between HEA and Cplex, where the provide bounds are compared and the average absolute performance of each of them is also analyzed.

Table I reports the results achieved by both the Cplex solver (column 2), HEA (column 3), the average absolute performance of both HEA (column 4), and Cplex (column 5). From Table I, we observe that:

- For the Cplex solver, its average absolute performance varies from 0 (VBP-n100-d5-10-30-LD-NR07) to 11 (VBP-n100-d5-10-30-D-NR16) and its global average absolute performance is equal to 4.3.
- For HEA, its average absolute performance is equal to 0 for overall instances, which also achieves a global average absolute performance of 0. It means that HEA dominates the state-of-the-art Cplex solver in all occasions.

TABLE I. Behavior of HEA versus the Cplex Solver

#Inst	Cplex	PSO	PSO / Cplex	Cplex / PSO
VBP-n100-d5-10-30-LD-NR01	4	**1**	0	3
VBP-n100-d5-10-30-LD-NR02	16	**12**	0	4
VBP-n100-d5-10-30-LD-NR03	11	**8**	0	3
VBP-n100-d5-10-30-LD-NR04	17	**11**	0	6
VBP-n100-d5-10-30-LD-NR05	8	**5**	0	7
VBP-n100-d5-10-30-LD-NR06	30	**23**	0	7
VBP-n100-d5-10-30-LD-NR07	**19**	**19**	0	0
VBP-n100-d5-10-30-LD-NR08	18	**13**	0	5
VBP-n100-d5-10-30-LD-NR09	12	**9**	0	3
VBP-n100-d5-10-30-LD-NR10	9	**5**	0	4
VBP-n100-d5-10-30-LD-NR11	16	**15**	0	1
VBP-n100-d5-10-30-LD-NR12	16	**7**	0	9
VBP-n100-d5-10-30-LD-NR13	14	**11**	0	3
VBP-n100-d5-10-30-LD-NR14	12	**8**	0	4
VBP-n100-d5-10-30-LD-NR15	12	**8**	0	4
VBP-n100-d5-10-30-LD-NR16	19	**8**	0	4
VBP-n100-d5-10-30-LD-NR17	26	**24**	0	2
VBP-n100-d5-10-30-LD-NR18	10	**6**	0	4
VBP-n100-d5-10-30-LD-NR19	12	**9**	0	3
VBP-n100-d5-10-30-LD-NR20	20	**13**	0	7
Average			0	4.3

Fig. 1 illustrated the variation of the quality of the solutions over the twenty instances tested by both Cplex solver and HEA

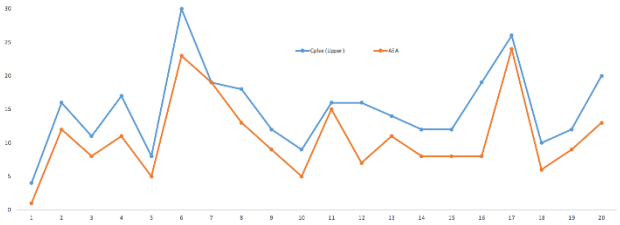

Figure 1. Variation of the bounds achieved by both Cplex and HEA.

TABLE II. P-Values for Sign Test and Wilcoxon Rank Test on all Tested Instances with the Significance Level α = 0:05

	Sign test	N^+	N^-	Wilcoxon test
All instances	1	0	20	1

From Table II we observe that the p-value for the Cplex vs HEA is greater than the significance level $\alpha = 0.05$ for both sign and Wilcoxon test. It indicates that HEA performs better than the Cplex solver, by rejecting the hypothesis H_0 (H_0: Bound(Cplex) - Bound(HEA) < 0) in all the experiments (as illustrated in Fig. 2).

Table III reports the average absolute performance of both IVNH and HEA on the twenty instances. From that table, one can observe that HEA remains very competitive, where it matches all the best solutions while HEA matches 17 out of the 20 tested instances.

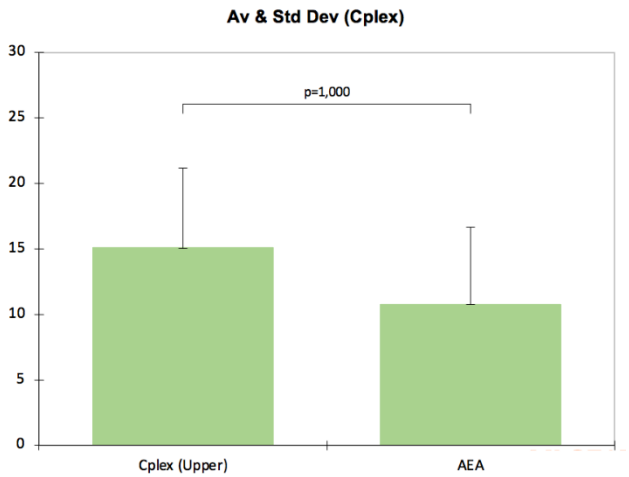

Figure 2. Illustration of p-values related to two compared algorithms: HEA versus Cplex.

In this case, the HEA's global average absolute performance is equal to 0 whereas IVNH is equal to 0.1; it means that HEA provides some better bounds (we recall that runtime limit of the Cplex solver was fixed to 1 hour, where our used machine is slightly more competitive and so, the Cplex *solver* may provide better bounds than those used in Otto and Li [3].

TABLE III. BEHAVIOR OF HEA VERSUS THE CPLEX SOLVER

Cplex-Best	IVNH-Best	AAP-Cplex	AAP-IVNH
3	17	3	0.1
Cplex-Best	HEA-Best	AAP-Cplex	AAP-HEA
0	**20**	**4.3**	**0**

V. CONCLUSION

In this paper, we investigated the use of an evolutionary algorithm for solving a special case of the packing problem: the *m*-vector bin packing problem with sequencing. The proposed algorithm is based upon the particle swarm optimization augmented with the complete procedure which serves to repair each solution at hand. The starting archive set containing a set of feasible solutions was built by tailoring a first fit bin packing procedure. Second, each particle is normalized and corrected using a repairing operator which is also enhanced by applying two intensifying operators. Finally, the behavior of the method was analyzed on a set of benchmark instances and its provided results were compared to those reached by a more recent published method of the literature, and to those achieved by the state-of-the-art Cplex solver. Encouraging results have been obtained.

CONFLICT OF INTEREST

The authors declare that they have no conflict of interest.

AUTHOR CONTRIBUTIONS

Mhand Hifi proposed the main idea related to this work. He designed the first hybrid evolutionary method. Amina N. Benahmed (PhD student supervised by Pr Mhand Hifi) and Isma Dahmani coded the first version of the algorithm which is based upon the above idea. Mhand Hifi 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. Pr Méziane Aïder is the second supervisor of the PhD student.

REFERENCES

[1] G. Wascher, H. Haussner, and H. Schumann, "An improved typology of cutting and packing problems," *European Journal of Operational Research*, vol. 183, pp. 1109-1130, 2007.
[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] O. Battaïa, A. Otto, F. Sgarbossa, and E. Pesch, "Future trends in management and operation of assembly systems: From customized assembly systems to cyber-physical systems," *Omega*, vol. 78, pp. 1-5, 2018.
[4] M. Aïder, F. Z. Baatout, and M. Hifi, "A reactive search-based algorithm for scheduling multiprocessor tasks on two dedicated processors," in *Proc. 15th Conference on Computer Science and Information Systems*, 2020, pp. 257-261.
[5] A. Otto and X. Li, "Product sequencing in multiple-piece-flow assembly lines," *Omega*, vol. 91, p. 102055, 2020.
[6] 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.
[7] G. A. Croes, "A method for solving traveling-salesman problems," *Operations Research*, vol. 6, pp. 791-812, 1958.
[8] M. Aïder, F. Z. Baatout, and M. Hifi, "A look-ahead strategy-based method for scheduling multiprocessor tasks on two dedicated processors," *Computers & Industrial Engineering*, vol. 158, no 107388, 2021.
[9] 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, no. 113224, 2020.

Méziane Aider is a full professor of Operations Research at USTHB (University of Sciences and Technology Houari Boumediene), Algeria. He is the head of the Laboratory of Operations Research and Mathematics of Decision (LaROMaD). His research work focuses on single and multi-objective combinatorial optimization, graph theory, and didactics of mathematics. He is a regular reviewer for more than twenty international journals.

Amina N. Benahmed is a PhD student at the USTHB (University of Sciences and Technology Houari Boumediene), Algeria. She received her BS in 2014 at the USTHB, and got her MS in 2016 from the same University.

Isma Dahmani is an associate professor of Operations Research at USTHB (University of Sciences and Technology Houari Boumediene), Algeria. Her area of interest is the resolution of combinatorial optimization problems using approximate, hybrid and exact methods.

Mhand Hifi is a full professor of Computer Science and Operations Research at UPJV (University of Picardy 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.