All Sciences Proceedings
http://as-proceeding.com/

2nd International Conference on Contemporary Academic Research
November 4-5, 2023 : Konya, Turkey

© 2023 Published by All Sciences Proceedings

ICCAR
https://as-proceeding.com/index.php/iccar

Population-Based Vortex Search Optimization Algorithm for Solving Bin Packing Problem

Tahir SAĞ

Department of Computer Engineering, Selçuk University, Türkiye

tahirsag@selcuk.edu.tr

***Abstract*—** The Population-based Vortex Search Optimization Algorithm abbreviated as PVS is applied to the bin packing problem in this study. PVS is a recently developed metaheuristic converting the original vortex search algorithm inspired by the rotational motion of liquids from a single solution-based to a population-based optimizer. Metaheuristics that rely on a single solution focus on exploring the vicinity of that individual solution, whereas population-based metaheuristics carry out their search by generating multiple candidate solutions at various locations within the search space. In this context, PVS is a remarkably successful algorithm that combines the advantages of two types of techniques. Here, PVS is tested by running on the bin packing problem which is a classic optimization challenge with broad industrial relevance, involving the efficient allocation of items of varying sizes into a limited number of containers to minimize wasted space. Moreover, the findings of PVS are compared with the firefly algorithm, and invasive weed optimization algorithm. Experimental results prove that PVS is a successful tool with the potential to solve industrial optimization problems.

Keywords — PVS, Bin Packing Problem, Metaheuristic Optimization.

I. INTRODUCTION

Many real-world optimization problems fall under the category of NP-hard problems, making them challenging to solve efficiently. Due to the impracticality of exact algorithms, researchers prioritize metaheuristic algorithms for tackling complex computational tasks. Metaheuristics, often inspired by nature and biological systems, efficiently guide the search for global optima. They achieve this by iteratively updating candidate solution positions, originally randomly generated in the search space, based on information from previous iterations. Thus, minimizing the reliance on the initial starting point enhances the algorithm's robustness. Metaheuristics can be categorized into two groups: single-solution-based algorithms, exemplified by Simulated Annealing, focus on local search, and operate more exploitatively but risk getting stuck at local optima. Population-based algorithms, like Genetic Algorithms and Particle Swarm Optimization, maintain multiple solutions across different search space locations, promoting better exploration and diversity preservation. Population-based techniques are more prevalent in the literature than single-solution-based ones, primarily due to their superior exploration capabilities, reducing the likelihood of local optima traps [1].

This study focuses on a population-based version of the Vortex Search (VS) algorithm [2] called PVS, which was proposed by the author of this study in 2022 to improve the discovery ability of the single-solution-based VS algorithm without disrupting its original working structure. The PVS algorithm comprises two primary phases, with an equal division of the vortex size allocated to each phase.

During the initial phase, candidate solutions are generated randomly by incorporating the calculated radius value and applying a Gaussian distribution centered around the focal point, mirroring the methodology of the original VS algorithm. In the subsequent phase, the location update mechanism, which incorporates the probabilistic selection technique from the ABC algorithm, is integrated into the VS algorithm, thereby generating the second half of the population. Furthermore, to enhance the disturbance within the population, the polynomial mutation operator is employed.

The Bin Packing Problem is a well-known combinatorial optimization challenge in the field of computer science. It belongs to the class of NP-hard problems, indicating that it is computationally difficult to find optimal solutions in polynomial time for large instances. The fundamental objective of the Bin Packing Problem is to efficiently pack a set of items of varying sizes into a minimum number of identical or capacity-constrained bins with the goal of minimizing wasted space. Each item has a specific size, and the bins have a fixed capacity. The challenge lies in determining how to assign items to bins in such a way that the number of bins used is minimized.

In the context of this research, we have applied the population-based Vortex Search algorithm to address the Bin Packing Problem. To assess its performance, we conducted experiments employing the Firefly Algorithm (FA) [3] and the Invasive Weed Optimization Algorithm (IWO) [4]. The outcomes of our investigations demonstrate that PVS exhibits significant promise as a highly effective and alternative tool in comparison to other established metaheuristic algorithms.

The rest of this paper is organized as follows. Population-Based Vortex Search Optimization Algorithm is briefly explained in section 2. The Bin Packing Problem is introduced in section 3. Experimental results of the PVS and compared algorithms are given in section 4. Finally, concluding remarks and possible future works are provided in section 5.

II. Population-Based Vortex Search Optimization Algorithm

The Vortex Search algorithm is a single-solution-based metaheuristic, which is firstly introduced in 2015 [2]. VS is derived to solve numeric optimization problems from the vortex pattern formed by the swirling motion of stirred fluids. Algorithms based on a single solution, like VS, work quickly, but this feature can cause the search process to get stuck in a local optimum. Population-Based Vortex Search Optimization Algorithm (PVS) was proposed to introduce a population-based structure into the VS algorithm with the position update operator and polynomial mutation operator, while preserving the original working strategy of VS [5]. For this purpose, the candidate solutions (CS) are divided into two equal parts to be developed in two phases.

PVS starts by setting control parameters: population size (psize), vortex size (vsize=psize/2), termination condition, and probability of mutation (η_m). The parameters μ_0 and r_0 are also calculated in the same way as in VS. Then, CS (psize) is initially randomly generated in the first iteration, with only half (vsize) generated in subsequent iterations. The generation process, utilizing a Gaussian distribution as in the original VS, divides the population into two halves. One half is replicated through a best-center-oriented exploitation process, while the other undergoes an update via a population-based approach with selection pressure in mind. After this step, the central point is updated by replacing it with the best-found solution. After that, candidate solutions are matched using the proportional selection method. The selection probability (prob) for each candidate solution is calculated by Eq (1). Thus, each solution in the second half of the CS (from vsize+1 to psize) is updated with a solution determined by taking the prob values into account.

$$prob_i = csum_i / csum_{psize} \tag{1}$$

$$csum_i = \sum_{j=1}^{i} normp_i$$

$$normp_i = p_i / \sum_{i=1}^{psize} p_i$$

$$p_i = 0.9 \times \left(max\{\vec{f}\} - f_i\right) + 0.1$$

where p_i corresponds to the rescaled fitness value of the i-th solution, where the objective function values have been converted from minimization to maximization. f_i represents the fitness value associated with the i-th solution, while $max\{\vec{f}\}$ signifies the highest fitness value within the current population. $normp$ denotes the vector of probability values derived from normalizing the p values, ensuring they fall within the [0,1] range. On the other hand, $csum$ is the cumulative sum vector comprised of the normalized $normp$ values.

Within the latter half of the population, denoted as $i = \{vsize + 1, vsize + 2, \dots, psize\}$, a random neighboring solution is selected for each solution CS_i, using the probability vector derived from the entire population. A fresh solution, denoted as CS_{new}, is created by modifying the value of a randomly selected dimension according to Eq (2). Then, values exceeding the limit on the obtained dimension value are pushed to the limit values.

$$\begin{aligned} CS_{new} &= CS_{current} \\ &\quad then \\ CS_{new}^{i} &= CS_{current}^{i} \\ &\quad + \left(CS_{current}^{i} - CS_{neigbour}^{i}\right) \\ &\quad \times (r - 0.5) \times 2 \end{aligned} \qquad (2)$$

where r represents a random number within the range of [0,1]. The fitness value of CS_{new} is calculated and then compared with the current solution $CS_{current}$. If CS_{new} outperforms $CS_{current}$, it supersedes the latter as the new solution. In cases where CS_{new} doesn't surpass $CS_{current}$, we generate a mutant solution, CS_{mutant} using the polynomial mutation method as described in Eq (3) referenced in [6].

$$\begin{aligned} CS_{mutant} &= CS_{current} \\ &\quad + \delta_q \times (upper - lower) \end{aligned} \qquad (3)$$

$$\delta_q = \begin{cases} [2r + (1-2r)(1-\delta_1)^{\eta_{\mathrm{m}}+1}]^{\frac{1}{\eta_{\mathrm{m}}+1}} - 1, \\ 1 - [2(1-r) + 2(r-0.5)(1-\delta_2)^{\eta_{\mathrm{m}}+1}]^{\frac{1}{\eta_{\mathrm{m}}+}} \end{cases}$$

$$\delta_1 = \frac{CS_{current} - lower}{upper - lower}$$

$$\delta_2 = \frac{upper - CS_{current}}{upper - lower}$$

Polynomial mutation provides a perturbation effect by distorting a solution. Subsequently, a selection process is carried out in a greedy manner between $CS_{current}$ and CS_{mutant}. Upon completing this step, the center point (μ) is updated by replacing it with the newly discovered best solution. After the completion of one generation, the size of the radius to be utilized in the subsequent generation is reduced. This ensures that in the first phase, a number of solutions within the reduced radius, denoted as $vsize$, are reproduced. In the second phase, stochastic information continues to be utilized with the solutions constituting the second half of the population. The PVS algorithm remains operational until it reaches the maximum number of function evaluations. For a more detailed description of the algorithm, readers can refer to the original article referenced in [5].

III. Bin Packing Problem

The Bin Packing Problem (BPP) is a classic optimization problem in computer science and mathematics. BPP is encountered as a classical computational challenge. In its most canonical formulation, a collection of bins, each endowed with finite capacity, and a set of items characterized by known weights, are presented as the input. The primary objective inherent in the BPP is the allocation of these items to the bins, sans division, in a manner such that the summation of item weights within any given bin does not surpass the bin's specified capacity, all while striving to employ the minimum number of bins attainable. The BPP, due to its NP-hard nature, signifies a computationally intricate conundrum, denoting that the quest for an optimal solution to this problem entails a formidable computational burden, particularly when confronted with sizable instances. Numerous heuristic methods and approximation algorithms have been devised to expedite the derivation of satisfactory solutions, though they may not guarantee optimality [7-9]. The fundamental one-dimensional BBP is mathematically defined by Eq (4), Eq (5), and Eq (6).

$$Minimum \sum_{b \in Bins} y_b \quad (4)$$

$$Subject\ to \sum_{b \in Bins} x_{pb} = 1 \quad (5)$$

$$Minimum \sum_{p\, \in\, Products} size_p x_{pb} \le cap_b \quad (6)$$

$y_b \in \{0,1\} \quad \wedge \quad x_{pb} \in \{0,1\}, \quad p \in Products,$

where $size_p$ represents the size of the pth elements in the Products set, and Bins = {1, ..., NP} denotes the set of candidate bins based on each one having a capacity shown as cap_b. The constraint defined in Eq (2) guarantees that each product is assigned to exactly one bin. y_b and x_{pb} are control variables that can only take values of 0 and 1. $if\ x_{pb} = 1$, it means that product p is assigned to bin b. Similarly, if $y_b = 1$, it indicates that box b is used.

IV. Experimental Results

In this study, the PVS algorithm was tailored to address the bin packing problem, and the algorithm's outcomes in solving this problem were juxtaposed with those of two established algorithms from the literature: the Firefly Algorithm (FA) and the Invasive Weed Optimization Algorithm (IWO). The bin packing problem instance employed in this study was sourced from the website www.yarpiz.com [10]. Additionally, the source code of the PVS algorithm can be accessed from the website www.mathworks.com [11].

This specific problem entails the task of determining the optimal allocation of 30 differently sized products into containers, each with a maximum capacity of 30, with the objective of minimizing the overall cost. The cost function used in the solutions incorporates penalty terms for instances where the container capacity is exceeded.

All algorithms were executed under uniform conditions on a computer equipped with an Intel(R) Core(TM) i5-4210U CPU@1.70GHz 2.40GHz processor and 8 GB of RAM. Each algorithm underwent ten repetitions with a maximum limit of 1000 function evaluations (MaxFEs). Subsequently, the mean elapsed times and the mean values, along with their respective standard deviations, were computed based on the obtained results. The results are presented in Table 1. Figure 1, Figure 2, and Figure 3 respectively depict a visual representation of one of the best results obtained by the PVS, FA, and IWO algorithms over 10 runs for the bin packing problem.

Table-1. The statical results of the algorithms.

	PVS	FA	IWO
best	7	7	7
worst	8	8	8.7500
mean	7.3571	7.3000	7.9393
std	0.4774	0.4830	0.5448
mean elapsed time	2.5030	17.007	5.5469

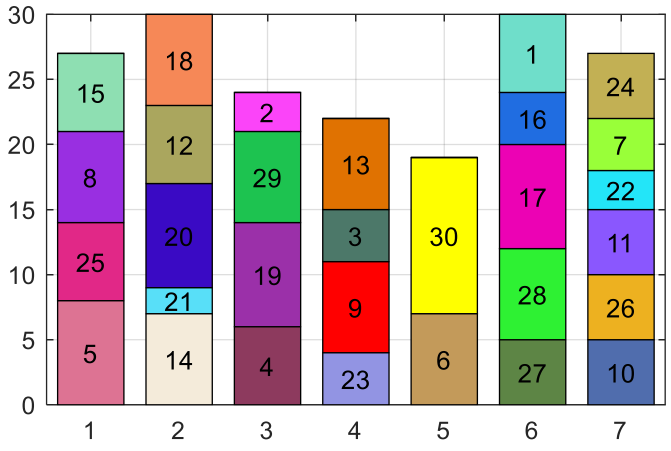

Fig. 1. Placement of the bins for the solution found by PVS

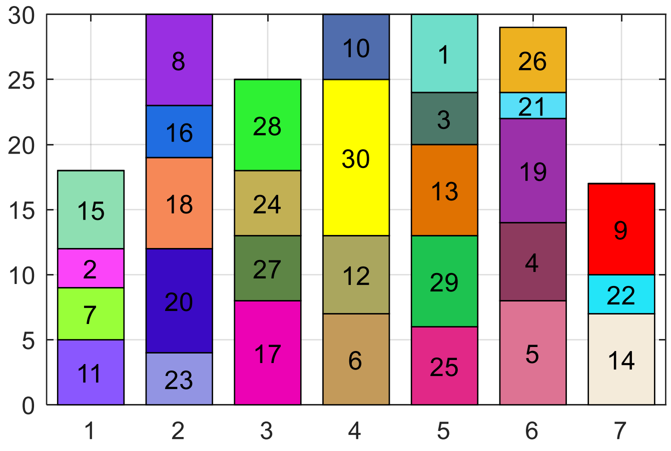

Fig. 2. Placement of the bins for the solution found by FA

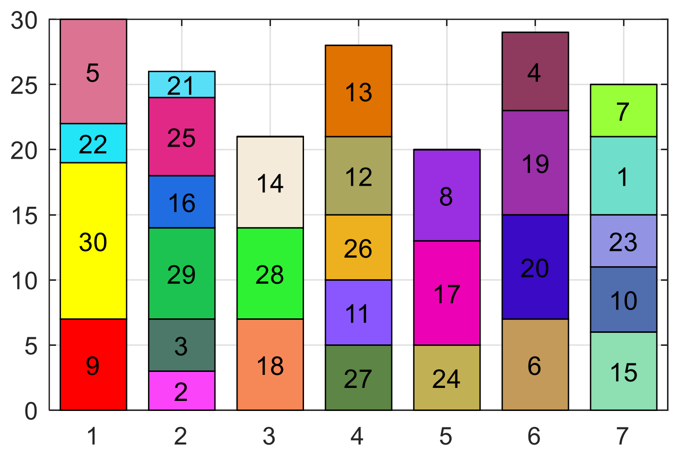

Fig. 3. Placement of the bins for the solution found by IWO

As illustrated in the figures, all algorithms converged to a cost value of 7 after 2000 MaxFEs in their respective solutions. This consistency is further corroborated by the concurrence of these best values in Table-1. The notable divergence among the algorithmic solutions emerges in the arrangement of products within the bins. Specifically, in 6 out of 10 runs, PVS achieved the optimal result, whereas FA and IWO achieved the same outcome in 7 and 2 runs, respectively. In terms of computational efficiency, PVS exhibits a significantly shorter runtime compared to the other two metaheuristic algorithms. On average, PVS completed its computations in a mere 2.5 seconds, whereas FA required 17 seconds, and IWO completed the process in 5.5 seconds. This situation, while arising as a manifestation of PVS's fast single-solution-based working characteristic, also enables it to achieve the same best values as other metaheuristics due to its advanced exploration ability when operating in a population-based manner.

V. Conclusions

This study has introduced and applied the Population-Based Vortex Search Optimization Algorithm to the challenging Bin Packing Problem. BPP is a classic NP-hard optimization problem with significant industrial relevance, involving the efficient allocation of items into bins to minimize wasted space. PVS, a population-based metaheuristic algorithm, has been demonstrated to be a promising tool for solving this problem efficiently. The experimental results presented in this study have shown that PVS performs competitively with established metaheuristic algorithms such as the Firefly Algorithm and the Invasive Weed Optimization Algorithm. PVS achieved optimal solutions in a majority of runs, showcasing its effectiveness in tackling BPP. Moreover, PVS exhibited superior computational efficiency, completing its computations in significantly less time compared to FA and IWO.

The findings of this research highlight the potential of PVS as an alternative and efficient tool for solving industrial optimization problems like the BPP. As future work, further investigations can explore the application of PVS to other combinatorial optimization challenges and the fine-tuning of its parameters to enhance its performance in various domains. In summary, the Population-Based Vortex Search Optimization Algorithm, as presented in this study, offers a promising approach to addressing complex optimization problems and contributes to the growing body of research in the field of metaheuristic algorithms.

REFERENCES

[1] X.-S. Yang, *Engineering optimization: an introduction with metaheuristic applications*. John Wiley & Sons, 2010.

[2] B. Doğan and T. Ölmez, "A new metaheuristic for numerical function optimization: Vortex Search algorithm," *Information Sciences*, vol. 293, pp. 125-145, 2015/02/01/ 2015, doi: https://doi.org/10.1016/j.ins.2014.08.053.

[3] X.-S. Yang, *Nature-inspired metaheuristic algorithms*. Luniver press, 2010.

[4] A. R. Mehrabian and C. Lucas, "A novel numerical optimization algorithm inspired from weed colonization," *Ecological Informatics*, vol. 1, no. 4, pp. 355-366, 2006/12/01/ 2006, doi: https://doi.org/10.1016/j.ecoinf.2006.07.003.

[5] T. Sağ, "PVS: a new population-based vortex search algorithm with boosted exploration capability using polynomial mutation," *Neural Computing and Applications*, vol. 34, no. 20, pp. 18211-18287, 2022/10/01 2022, doi: 10.1007/s00521-022-07671-x.

[6] K. Deb, *Multi-objective optimization using evolutionary algorithms*. John Wiley & Sons, 2001.

[7] M. Casazza and A. Ceselli, "Mathematical programming algorithms for bin packing problems with item fragmentation," *Computers & Operations Research*, vol. 46, pp. 1-11, 2014/06/01/ 2014, doi: https://doi.org/10.1016/j.cor.2013.12.008.

[8] B. Byholm and I. Porres, "Fast algorithms for fragmentable items bin packing," *Journal of Heuristics*, vol. 24, no. 5, pp. 697-723, 2018/10/01 2018, doi: 10.1007/s10732-018-9375-z.

[9] H. I. Christensen, A. Khan, S. Pokutta, and P. Tetali, "Approximation and online algorithms for

multidimensional bin packing: A survey," *Computer Science Review*, vol. 24, pp. 63-79, 2017/05/01/ 2017, doi: https://doi.org/10.1016/j.cosrev.2016.12.001.

[10] M. K. Heris. "Bin Packing Problem using GA, PSO, FA, and IWO." Yarpiz. https://yarpiz.com/363/ypap105-bin-packing-problem (accessed September 18, 2023.

[11] T. Sag. "The MATLAB Source Code of Population-based Vortex Search Algorithm (PVS)." https://www.mathworks.com/matlabcentral/fileexchange/127938-pvs-population-based-vortex-search-algorithm?s_tid=srchtitle_site_search_1_pvs (accessed September 18, 2023.