International Journal of Mathematics Trends and Technology
ISSN: 2231-5373/ https://doi.org/10.14445/22315373/IJMTT-V70I9P104

Volume 70 Issue 9, 26-34, September 2024
© 2024 Seventh Sense Research Group®

Original Article

Comparison of Heuristics for Packing While Travelling Problem with Dropping Rate

Rani Kumari[1], Kamal Srivastava[2]

[1,2]*Department of Mathematics, Dayalbagh Educational Institute, Uttar Pradesh, India.*

[1]*Corresponding Author : rani.kumari95@gmail.com*

Received: 27 July 2024 Revised: 29 August 2024 Accepted: 16 September 2024 Published: 30 September 2024

Abstract *- Many real-life optimization problems consist of multiple well known problems which are interconnected with each other. A recent example of such a problem is the Travelling Thief Problem (TTP) or Packing While Travelling Problem (PWTP), which combines the classical Travelling Salesman problem and the Knapsack problem. In this paper, heuristics for a variant of PWTP, namely the PWTP with dropping rate (PWTP_DR), are examined. In PWTP_DR, a vehicle with a fixed-capacity container visits several cities exactly once. These cities contain various items that have specific profits and weights assigned to them. The vehicle has to pick up the items from the cities while meeting the container capacity constraint. In the process, the vehicle's speed decreases as the weight of the container increases, while the profit associated with an item drops by a factor called dropping rate, which depends on the duration of the article in the container. The problem is to maximize the total profit while minimizing the tour time. So far, this problem has not received much attention, with the only exception of the well-known NSGAII based metaheuristic (NSGAII_PWTP_DR) that utilizes a construction heuristic to create a population of initial solutions, which then evolves through an iterative process. In this work, three heuristics tailored to the problem are designed, and their various combinations are investigated with an aim to achieve a well-diversified population. Comprehensive experiments supported by statistical tests provide the best combination of heuristics to generate a population of solutions as it yields results that are significantly better than those obtained by the previously proposed NSGAII_PWTP_DR.*

Keywords *- Bi-objective optimization problem, NSGAII, Heuristic, Travelling thief problem, Dropping rate, Packing While Trravelling Problem.*

1. Introduction

Goods with short shelf life, such as fruits, vegetables, flower cuttings, and medicines, are time and/or temperature-sensitive items. They require fast and safe delivery to maintain quality and effectiveness. Delivering items with a short shelf life by road poses a significant challenge due to the increasing traffic congestion. As the travel time increases, the profit associated with such items decreases; hence, it is crucial to minimize the travel time, which in turn requires good planning of the tour. Further, as more items are picked, the profit increases. Nevertheless, the velocity of the vehicle comes down. Consequently, this leads to a rise in the duration of travel, resulting in a decrease in the overall value of the items over time. Motivated by such complex problems in real life, a hybridization of standard optimization problems, namely the Travelling Thief Problem (TTP), was introduced by Bonyadi et al. [1] in 2013. This problem is a combination of classical optimization problems - Knapsack Problem (KP) [2] and Travelling Salesman Problem (TSP) [3], in which objective functions of both the problems are connected by velocity, rent or dropping rate. Two variants of TTP, namely- TTP1 (Single objective optimization problem) and TTP2 (Bi-Objective Optimization Problem (BOOP)), are proposed in [1]. We refer to TTP2 as PWTP with dropping rate (PWTP_DR) as this nomenclature signifies the motivation behind the problem and the dropping rate as the interconnecting component of the two subproblems. In fact, Polyakovskiy et al. [4] used the name Packing while Travelling Problem (PWTP) for the TTP with the fixed tour. After the formulation of PWTP and PWTP_DR, Polyakovskiy et al. [5] came up with a test suit consisting of 9720 benchmark instances for PWTP and Bi-objective PWTP [6] which is an important contribution for the researchers working on this problem and its subsequent variants. The single objective PWTP, i.e. TTP1, is dealt with by many researchers using socially inspired algorithm [7], memetic algorithm [8], branch and bound [9], fitness landscape analysis using hill climbing [10], Efficient hybrid-local search [11], Hyper heuristic approach [12], and a study of the influence of subproblem on the quality of TTP [13]. Rodriguez et al. [12] used a sequence-based hyper-heuristic approach for TTP1, which utilizes a simulated annealing algorithm and some local search. They also generated several sets of TTP1 instances. Junfeng et al. [13] studied the influence of subproblem solutions on the quality of PWTP solutions. Blank et al. [6] proposed a Bi-objective variant of TTP1 (Bi_PWTP) by removing the rent of the container. They also implemented the well-known metaheuristic NSGAII [6] for Bi_PWTP. This version of PWTP received a lot of attention from other researchers too.

This is an open access article under the CC BY-NC-ND license (http://creativecommons.org/licenses/by-nc-nd/4.0/)

Good quality solutions of Bi_PWTP were obtained by using dynamic programming [14], Non-Dominated Tournament Genetic Algorithm (NTGA) [15], Variable neighbourhood Search [16], Non-Dominated Sorting Biased Random-Key Genetic Algorithm (NDS-BRKGA) [17] and weighted sum method [18]. NDS-BRKGA [17] proved to be best for smaller instances, whereas the weighted sum method [18] performs better for larger ones, even though it relies heavily on the weights assigned to each objective function.The original bi-objective variant TTP2 or PWTP_DR has not received much attention from the researchers despite its real-life applications. The only effort made towards solving this problem is a recent work by Kumari and Srivastava [19] in which NSGAII is adapted and implemented for this problem. NSGA II, being a population-based metaheuristic, requires a diverse population of solutions.

Thus, in [19], three construction heuristics are proposed and tested, and the best one is used to create the initial population. This leads to a set of solutions having the same tour but different packing plans. Besides, it does not consider the impact of taking multiple heuristics together in the population. The incorporation of multiple heuristics simultaneously has the potential to yield improved results by allowing for the exploration of solution space using multiple tours, which may contribute to getting better solutions. The lack of comparative analysis on the significance of using a combination of heuristics to generate an initial population of solutions for PWTP_DR motivates us to investigate whether different combinations of heuristics could improve the quality of solutions in NSGAII. The remaining part of the paper is organised as follows: Section 2 presents the relevant preliminaries and definitions pertaining to a BOOP. A detailed description and mathematical model of PWTP_DR is given in Section 3. The three construction heuristics are explained in Section 4. Section 5 is dedicated to the experiments and discussion of the results. The concluding remarks summarize the paper.

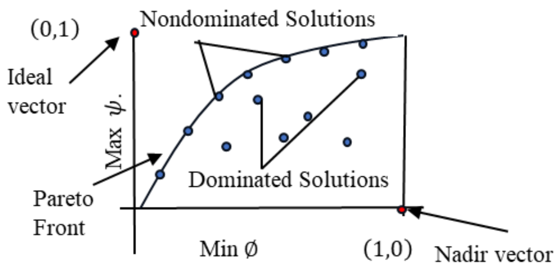
2. Preliminaries and Notations

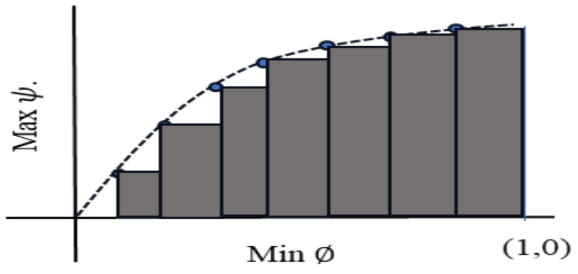
Given that PWTP_DR represents a bi-objective optimization problem, it involves two objective functions, ϕ and ψ, with the goal of minimizing $\emptyset$ while maximizing ψ. Let $\mathcal{B} \subseteq \{V\text{: } V \text{ is a solution vector with } q \text{ coordinates (decision variables)}\}$. A solution $V^{(i)} \in \mathcal{B}$ is said to be a nondominated solution (NDS) in $\mathcal{B}$ if there does not exist $V \in \mathcal{B}$ such that $\emptyset(V) > \emptyset(V^{(i)})$ and $\psi(V) < \psi(V^{(i)})$. The Pareto front (*PF*) of $\mathcal{B}$ is the set of all NDS, as shown in Figure 1(a). In multi-objective optimization problems, comparison between the *PF*s obtained for a problem by two distinct runs of the same metaheuristic or by different techniques is usually compared using a metric, namely- the hypervolume (*HV*) of the pareto front [2]. *HV* measures the area covered by the *PF* with respect to Nadir point, which is the vector containing the worst values of both the objective functions, i.e. $(\max \emptyset, \min \psi)$, in the case of the min-max type of objective functions. Besides, a vector consisting of the best value of both the objective functions is known as the Ideal point or Perfect point. Since PWTP_DR is a min-max type problem, the Ideal point is computed as $(\min \emptyset, \max \psi)$.

It is obvious that all the values of the objective functions need to be normalized between 0 and 1 using the Perfect point and Nadir point before computing the *HV*. Nadir point and the Ideal point are also normalized to (0, 1) and (1, 0) respectively. The area of the shaded region shown in Figure 1(b) is the hypervolume of the pareto front. Nadir point, shown in Figure 1 (a), is (0, 1), which corresponds to the normalized lowest and highest values (worst values) of $\emptyset$ and ψ respectively. Hence, if normalized $PF = \{(x_i, y_i)\}_{i=1,2,\ldots,n}$ where (x_i, y_i)'s are arranged in ascending order of $x_i's$, then

$$HV = \sum_{i=1}^{n-1} (x_i - x_{i-1})(y_0 - y_1) \tag{1}$$

Where (x_0, y_0) is the Nadir Point





(a) Nondominated solution, dominated solution and pareto front (normalized) **(b) Hypervolume obtained by Min, Max pair of objective functions**

Fig. 1 Nondominated solutions, dominated solutions, pareto front, Ideal and Nadir vectors of Normalized pareto front and hypervolume of min-max type bi-objective optimization problem

3. Packing While Travelling Problem with Dropping Rate

In PWTP_DR a vehicle having a container with capacity (C) has to visit a set of cities $N = \{1, 2, \ldots, n\}$ exactly once and return back to the initial city. The distance between cities is given by the distance matrix $DM = \left[d_{ij}\right]_{i,j=1,2,..,n}$ where d_{ij} is the distance between city i and city j. The set $M = \{1, 2, \ldots, m\}$ contains items available at these n cities. The weight and initial profit of i^{th} item is given by w_i and $p_i, i = 1, 2, \ldots, m$ respectively. The total weight of items picked from the city x_i, is denoted by wt_{x_i}. In this problem, items are to be chosen within their container capacity C during the tour, which is a sequence of cities $X = \{x_1, x_2, \ldots, x_n\}$. Also, whenever the weight in the container increases, the velocity of the vehicle in the city x_k reduces to

$$v_{x_k} = v_{max} - \frac{\sum_{j=1}^{k} wt_{x_j}}{C}(v_{max} - v_{min}) \tag{2}$$

where maximum and minimum velocities of the vehicle are given by v_{max} and v_{min} respectively. The time needed to travel between two consecutive cities is

$$t_{x_i,x_{i+1}} = \frac{d_{x_i,(x_{i+1})}}{v_{x_i}} \tag{3}$$

If T_i represents the duration for which i^{th} item is kept in the container, the final reduced profit (p_i^F) of item i depends on T_i, a constant α, dropping rate Dr and is calculated by

$$p_i^F = p_i \times Dr^{\left\lfloor \frac{T_i}{\alpha} \right\rfloor} \tag{4}$$

Let the packing plan be denoted by $Y = (y_1, y_2, \ldots, y_m)$, where y_i denotes the city from where i^{th} item is picked. If i^{th} item is not picked, then $y_i = 0$. The objective is to find a solution $S = (X, Y)$ which

$$minimizes\ f(X,Y) = \sum_{i=1}^{n-1}\left(t_{x_i,x_{i+1}}\right) + \left(t_{x_n,x_1}\right) \tag{5}$$

$$maximizes\ g(X,Y) = \sum_{i=1}^{m}\left(p_i\chi_i \times Dr^{\left\lfloor \frac{T_i}{\alpha} \right\rfloor}\right) \tag{6}$$

$$subject\ to\ the\ constraint \quad \sum_{i=1}^{m} w_i\chi_i \leq C \tag{7}$$

where $\chi_i = \begin{cases} 1\ if\ y_i \neq 0 \\ 0\ if\ y_i = 0 \end{cases}$

It is worth noting that for a given problem instance, the first city is always fixed, and items from the first city can be picked only at the beginning of the tour. They cannot be picked up while returning. Further, it is permitted to collect more than one item from a city, while the same item cannot be picked from multiple cities.

4. Heuristics for Packing While Travelling Problem with Dropping Rate

PWTP_DR is an NP-hard, bi-objective optimization problem. The application of exact algorithms for solving such problems is often impractical. Furthermore, in real-world optimization scenarios, exact results are rarely required. In such cases, heuristic algorithms that find approximate or near-optimal solutions but an acceptable time play an indispensable role. This paper aims to determine the optimal heuristic combination for generating the initial population of solutions used in the NSGAII metaheuristic for PWTP_DR. To commence, we outline the construction heuristics presented in [19].

4.1. Construction Heuristic1 (Cons_1)

Tour for the TSP part of PWTP_DR is generated by the Chained Lin Kernighan heuristic [3]. Once the tour is fixed, adding even a single item to the container results in a new solution with different profit and tour time. The objective is to maximize the sum of the final profit ($\sum_{i=1}^{m} p_i^F$), while satisfying inequality (7), which serves as a motivation for the next phase of the heuristic. Consequently, the efficiency of each item is determined by dividing its profit by its weight. The items are then arranged in descending order based on their efficiencies. Placing the first item from this sorted list into the container offers a feasible solution for PWTP_DR. However, the selection of the city from which this item is taken is done randomly, keeping in view that an item may be available in multiple cities. Similarly, including the next item from the list in the container leads to another solution. Continuing this process with the remaining items from the list until inequality (7) is satisfied results in subsequent feasible solutions.

4.2. Construction Heuristic2 (Cons_2)

The first step is to find the tour, which is generated as in *Cons*_1. same way. For the items picking the part of the solution, the associated factor with minimum distance ($AF\_MinD = \frac{profit}{weight \times MDTT}$) for each item is computed where *MDTT* refers to the minimum distance to be travelled by the vehicle while it is carrying this item in the container. Items are selected in descending order of AF_MinD.

In other words, it is the total distance travelled by the vehicle from the last city of availability of this item to the city positioned at the end of the tour. Items are then sorted in the descending order of their *AF_MinD* values. The first item in the list is picked and placed in the container. This provides one feasible solution to the problem. Now next item from the sorted list is picked and added to the container, resulting in another feasible solution. This process is repeated for the remaining items in the list till inequality (7) is satisfied, resulting in a set of solutions which are all feasible. It is worth noting that the total number of feasible solutions thus created can differ for each instance. In other words, this quantity is specific to both the instance and the heuristic employed.

4.3. **Construction Heuristic 3 *(Cons*_3)**

In *Cons*_3, the cities are first chosen on the basis of minimum distance criteria, and the final tour sequence is obtained by reversing the sequence. The next city to be visited after the current city x_i is chosen using the minimum distance criteria. Let the sequence of cities thus obtained is $x_1, x_2, \ldots, x_n$. As the vehicle traverses the cities, the weight in the container increases. Thus, it is desirable that towards the end of the trip, the vehicle travels shorter distances. Hence, the tour for the solution is taken in the reverse order, i.e., $x_1, x_n, x_{n-1}, \ldots, x_3, x_2, x_1 = X$ (say). Item selection procedure in *Cons*_3 after generating the tour is the same as discussed in *Cons*_2.

5. Experiments and Results

Implementation of the heuristics for PWTP_DR is done in C++, Intel core i7 laptop running at 1.80 GHz, 8GB RAM. The test suite of 100 instances (listed in Table 1) used in this work for the experiments is available at https://cs.adelaide.edu.au/~optlog/research/combinatorial.php. As already mentioned in Section 2 for a BOOP, the comparison of heuristics is done using the values of *HV* of the pareto fronts (*PF*) obtained by each run of the heuristics. Thus, corresponding to each instance, *HV* is computed using (3), which requires Ideal and Nadir Vectors for the instance. For obtaining these values, the tour for each instance is obtained using the LK heuristic [3] (the algorithm code used is available on https://github.com/lingz/LK-Heuristic), which provides the distance of the tour (*Dis*).

Nadir tour time is calculated by the distance of the tour with minimum velocity. $\left(\frac{Dis}{v_{min}}\right)$ and Ideal tour time is 0. Ideal profit is taken as the sum of profits of all items whereas Nadir profit is 0. These values are taken from [19]. For the purpose of investigating the best-case scenario for employing the construction heuristics to generate an initial set of solutions in a metaheuristic, for each test instance, the *HV* of the set of solutions obtained by each of the three construction heuristics is noted. These values are listed in the respective columns of Table 1.

Column *Cons*_1_2 contains the *HV* computed for the NDS of the solutions provided by both *Cons*_1 and *Cons*_2 taken together. Similarly, the *HV* values are reported for the pairs *Cons*_2, *Cons*_3 (*Cons*_2_3) and *Cons*_1, *Cons*_3 (*Cons*_1_3). Experiments also include the case when solutions are generated by all the three heuristics indicated by *Cons*_1_2_3. A significant observation from these results is that the combination heuristics *Cons*_2_3 and *Cons*_1_2_3 yield identical *HV* values across all test instances; hence, they are reported in the single column under the heading '*Cons*_2_3 / *Cons*_1_2_3'. This indicates that the solutions produced by *Cons*_1 are either dominated by those generated by *Cons*_2 and *Cons*_3 or are identical to them. The best value of each row is highlighted by showing it in bold. The mean of *HV* overall test instances corresponding to each heuristic is in the last row of Table 1. Another important observation is that the highest *HV* values are obtained by the combination *Cons*_2_3 in all instances, with the rest attaining these peaks in only a few instances. Consequently, the best average value of *HV* is achieved by the combination of *Cons*_2 and *Cons*_3. Therefore, there is no justification for utilizing *Cons*_1 in population generation, as incorporating additional heuristics necessitates increased computational time.

Table 1. Hypervolume (*HV*) obtained by heuristics.

	Cons_1	**Cons_2**	**Cons_3**	**Cons_1_2**	**Cons_1_3**	**Cons_2_3 / Cons_1_2_3**	**NSGAII_PWTP_DR**
10_3_2_25.txt	0.095558	0.122754	**0.156563**	0.122754	**0.156563**	**0.156563**	**0.156563**
10_3_2_50.txt	0.177782	**0.300793**	0.287729	**0.300793**	0.287729	**0.300793**	0.287729
10_3_2_75.txt	0.036294	**0.267623**	0.265721	**0.267623**	0.265721	**0.267623**	0.265721
10_3_3_50.txt	**0.360747**	**0.360747**	0.357551	**0.360747**	**0.360747**	**0.360747**	0.350633
10_3_3_75.txt	0.424959	0.426623	0.496007	0.426623	0.496106	**0.49771**	0.492106
10_3_4_50.txt	0.255851	0.391196	**0.394256**	0.391196	**0.394256**	**0.394256**	0.386705
10_5_1_50.txt	0.352114	0.371346	0.392162	0.371346	0.392531	**0.39327**	0.392162
10_5_1_75.txt	0.12158	0.517304	**0.523981**	0.517304	**0.523981**	**0.523981**	**0.524423**
10_5_2_50.txt	0.1602	0.236914	0.284675	0.236914	0.284739	**0.285103**	0.284675
10_5_2_75.txt	0.130439	0.294334	**0.629687**	0.294334	**0.629687**	**0.629687**	**0.630316**
10_5_3_25.txt	0.190386	0.210879	**0.221768**	0.210879	**0.221768**	**0.221768**	**0.221768**
10_5_3_50.txt	0.118766	**0.358015**	0.35327	0.358015	0.353591	**0.358015**	0.35327
10_10_1_25.txt	0.166429	0.250109	0.265015	0.250109	0.265141	**0.268651**	0.265015
10_10_4_75.txt	0.354925	0.399244	0.531199	0.399244	0.535585	**0.539912**	0.531345
10_10_5_25.txt	0.119683	0.14036	0.139217	0.14036	0.142556	**0.143473**	0.139217
10_10_5_50.txt	0.092857	0.210228	**0.321668**	0.210228	**0.321668**	**0.321668**	**0.321677**
10_10_5_75.txt	0.203493	0.329206	0.332346	0.329206	0.332468	**0.332839**	0.332508
10_10_6_25.txt	0.098743	0.173222	**0.272959**	0.173222	**0.272959**	**0.272959**	**0.272979**
10_15_2_50.txt	0.217051	0.31251	**0.315127**	0.31251	**0.315127**	**0.315127**	**0.315354**
10_15_2_75.txt	0.099499	0.278531	0.418339	0.278531	0.418363	**0.418451**	0.418361
10_15_3_25.txt	0.083197	0.160469	**0.233549**	0.160469	**0.233549**	**0.233549**	**0.234274**
10_15_3_50.txt	0.147742	0.225177	0.31435	0.225177	0.314543	**0.315111**	0.314487
10_15_3_75.txt	0.289047	0.606359	**0.613393**	0.606359	**0.613393**	**0.613393**	0.613329
10_15_4_25.txt	0.092311	0.269568	**0.298877**	0.269568	**0.298877**	**0.298877**	**0.299011**
10_15_4_50.txt	0.140151	0.295491	**0.374401**	0.295491	**0.374401**	**0.374401**	**0.374482**
20_5_1_25.txt	0.13762	0.231292	0.230877	**0.231292**	0.230893	**0.231292**	0.230877
20_5_1_50.txt	0.253846	0.326069	0.429221	0.326069	0.429789	**0.431041**	0.429221
20_5_1_75.txt	0.156379	0.410918	0.482369	0.410918	0.484147	**0.487408**	0.482612
20_5_2_25.txt	0.174567	0.113042	0.113748	0.113042	**0.177479**	**0.113748**	**0.113748**
20_5_2_50.txt	0.044747	0.19364	0.235923	0.19364	0.236715	**0.242715**	0.235923
20_10_2_25.txt	0.060449	0.174183	0.207745	0.174183	0.208322	**0.20869**	0.217448
20_10_2_50.txt	0.090825	0.347258	0.391484	0.347258	0.391801	**0.394934**	**0.391712**
20_10_2_75.txt	0.228254	0.603021	**0.709654**	0.603021	**0.709654**	**0.709654**	0.709994
20_10_3_25.txt	0.038801	0.232944	0.245218	0.232944	0.245708	**0.252111**	0.24528
20_10_3_50.txt	0.138731	0.368473	0.348825	0.368473	0.349509	**0.368473**	0.349236
20_10_3_75.txt	0.076516	0.318409	**0.357214**	0.318409	**0.357214**	**0.357214**	**0.360602**
20_10_4_25.txt	0.083174	0.226395	**0.241156**	0.226395	**0.241156**	**0.241156**	**0.241176**
20_20_1_75.txt	0.322979	0.551638	0.654114	0.551638	0.654543	**0.655996**	0.654226
20_20_2_25.txt	0.09353	0.207141	0.23963	0.207141	0.239643	**0.240067**	**0.242285**
20_20_2_50.txt	0.062989	0.306526	0.322696	0.306526	0.322881	**0.324928**	0.322757
20_20_2_75.txt	0.237011	0.613393	0.596888	0.613393	0.603534	**0.620404**	0.601581

20_20_3_25.txt	0.061715	**0.236274**	0.234283	**0.236274**	0.234305	**0.236274**	0.234406
20_20_3_50.txt	0.222748	**0.443646**	0.430934	**0.443646**	0.434333	**0.443646**	**0.443801**
20_25_1_25.txt	0.055155	0.134404	0.194718	0.134404	0.195188	**0.196248**	0.194821
20_25_1_50.txt	0.18073	0.364055	0.361712	0.364055	0.363181	**0.368298**	**0.381227**
20_25_2_25.txt	0.049422	0.268709	0.269502	0.268709	**0.269502**	**0.269502**	**0.269657**
20_25_2_50.txt	0.147354	0.404766	**0.4075**	0.404766	**0.4075**	**0.4075**	**0.407548**
20_25_2_75.txt	0.075402	0.319235	**0.418303**	0.319235	**0.418303**	**0.418303**	**0.418309**
20_25_1_75.txt	0.13501	0.3157	0.335892	0.3157	0.335959	**0.336881**	0.335991
20_25_3_25.txt	0.053339	0.246232	0.260944	0.246232	0.261137	**0.262937**	0.261007
50_15_1_25.txt	0.122553	0.248431	0.249871	0.248431	0.250758	**0.253826**	0.249912
50_15_1_50.txt	0.041883	0.413807	0.409267	0.413807	0.40989	**0.425423**	0.409339
50_15_1_75.txt	0.122908	0.369256	0.434581	0.369256	0.436172	**0.441324**	0.434631
50_15_2_25.txt	0.033961	**0.223408**	0.216363	**0.223408**	0.216733	**0.223408**	0.219333
50_15_2_50.txt	0.043205	0.364625	0.382943	0.364625	0.383695	**0.389443**	0.38307
50_15_2_75.txt	0.093516	0.407224	0.517674	0.407224	0.517729	**0.518372**	**0.532821**
50_25_8_50.txt	0.054865	0.300225	**0.314872**	0.300225	**0.314872**	**0.314872**	**0.314919**
50_25_8_75.txt	0.157273	0.553167	0.615084	0.553167	0.616053	**0.62323**	0.615104
50_25_9_25.txt	0.014195	0.146624	0.181992	0.146624	0.181994	**0.182301**	**0.21346**
50_25_9_50.txt	0.100835	0.425374	0.446436	0.425374	0.448479	**0.459665**	0.4466
50_25_9_75.txt	0.110345	0.554235	0.572333	0.554235	0.572358	**0.572977**	**0.57568**
50_25_10_25.txt	0.087516	0.263294	0.273513	0.263294	0.274889	**0.280236**	0.273581
50_50_1_25.txt	0.045002	0.218102	0.262683	0.218102	0.262696	**0.26277**	**0.263377**
50_50_1_50.txt	0.11384	**0.462158**	0.393074	**0.462158**	0.395405	**0.462158**	0.393165
50_50_1_75.txt	0.15256	0.542078	0.625482	0.542078	0.625487	**0.625851**	0.625772
50_50_2_25.txt	0.039201	0.18082	0.236153	0.18082	0.236329	**0.237342**	**0.249788**
50_50_2_50.txt	0.079669	0.432493	0.453516	0.432493	0.453637	**0.455125**	0.453552
50_50_2_75.txt	0.083961	0.520995	0.632598	0.520995	0.63328	**0.638795**	0.632647
50_75_1_25.txt	0.022438	0.171533	0.232156	0.171533	0.232157	**0.232243**	**0.23252**
50_75_8_50.txt	0.123408	**0.393265**	0.388738	**0.393265**	0.389265	**0.393265**	0.390982
50_75_9_50.txt	0.127251	0.468877	0.479131	0.468877	0.480449	**0.486948**	0.479331
50_75_9_75.txt	0.092509	0.538556	0.530985	0.538556	0.532009	**0.539932**	0.531002
50_75_10_25.txt	0.04134	**0.17413**	0.160275	**0.17413**	0.160776	**0.17413**	**0.174394**
50_75_10_50.txt	0.06563	**0.384094**	0.375332	**0.384094**	0.376458	**0.384094**	0.376007
50_75_10_75.txt	0.091932	0.648469	0.638459	0.648469	0.639568	**0.649275**	0.638478
100_3_9_50.txt	0.058633	0.462953	0.458478	0.462953	0.458478	**0.464005**	0.455942
100_3_9_75.txt	0.368213	**0.603445**	0.600899	**0.603445**	0.601016	**0.603445**	0.598878
100_5_10_25.txt	0.00405	0.032368	0.050306	0.032368	0.050342	**0.050752**	**0.101217**
100_5_10_50.txt	0.022983	0.225337	0.33251	0.225337	0.332512	**0.333508**	0.33251
100_25_9_75.txt	0.032987	0.495753	0.508332	0.495753	0.508615	**0.517292**	0.508332
100_25_10_25.txt	0.009198	**0.211757**	0.209848	**0.211757**	0.209913	**0.211757**	**0.219493**
100_25_10_50.txt	0.171252	0.4627	0.473659	0.4627	0.473693	**0.474702**	0.473719
100_50_1_25.txt	0.01587	0.203439	0.199989	0.203439	0.200126	**0.203439**	**0.215536**
100_50_1_50.txt	0.102754	0.404421	0.448393	0.404421	0.448533	**0.451313**	0.448577

100_50_10_25.txt	0.035391	0.212731	**0.230376**	0.212731	**0.230376**	**0.230376**	**0.230386**
100_50_10_50.txt	0.135988	0.458861	0.450596	0.458861	0.451173	**0.460441**	0.450929
100_50_10_75.txt	0.000033	0.040088	**0.474201**	0.040088	**0.474201**	**0.474201**	**0.474223**
100_100_8_25.txt	0.07637	**0.272944**	0.263156	**0.272944**	0.264592	**0.272944**	0.263166
100_100_8_50.txt	0.03828	0.420832	0.421887	0.420832	0.422439	**0.432367**	0.421896
100_100_9_75.txt	0.029793	0.394586	**0.532936**	0.394586	**0.532936**	**0.532936**	**0.533151**
100_100_10_50.txt	0.026899	**0.433721**	0.426672	0.433721	0.426766	**0.433721**	0.431498
100_100_10_75.txt	0.000389	0.080498	**0.437045**	0.080498	**0.437045**	**0.437045**	**0.443893**
100_150_1_25.txt	0.010453	0.118999	0.159682	0.118999	0.159684	**0.159995**	**0.161091**
100_150_3_75.txt	0.025138	0.374676	0.371426	0.374676	0.371613	**0.377208**	0.371431
100_150_4_25.txt	0.073498	0.462794	0.582336	0.462794	0.583059	**0.590208**	0.58235
100_150_10_25.txt	0.045609	0.259097	0.264577	0.259097	0.264773	**0.266795**	0.264588
100_150_10_50.txt	0.026217	0.192994	0.229307	0.192994	0.229519	**0.232024**	0.229961
100_150_10_75.txt	0.014404	0.390673	0.401244	0.390673	0.401267	**0.402626**	**0.40281**
100_25_8_50.txt	0.079496	0.454793	0.44522	0.454793	0.44649	**0.455142**	0.445266
100_150_1_50.txt	0.031647	0.342426	0.549945	0.342426	0.550033	**0.552579**	**0.554902**
Average	0.112084	0.328845	0.370609	0.328845	0.371788	0.375052	0.372788
# Best						64	39

As already mentioned, PWTP_DR has not been studied so far since it was proposed in [1] with the exception of a metaheuristic NSGAII_PWTP_DR [19] . The initial population for NSGAII_PWTP_DR [19] is generated using a single construction heuristic, namely *Cons*_3 and the test suite used for the experimentation contains only 20 instances. Therefore, for a fair and better comparative analysis, in this work, the test suite is extended to 100 instances, and NSGAII_PWTP_DR is executed with the same configuration as in [19] . The average hypervolume (*HV*) achieved over ten runs is documented in the last column, "NASGAII_PWTP_DR", of Table 1. In this table, the best *HV* values obtained from both *Cons*_2_3 and NSGAII_PWTP_DR are recorded in their respective columns, with the values obtained by NSGAII highlighted in bold italics if they are best in that row. Based on the data presented in Table 1, it is evident that *Cons*_2_3 outperformed NSGAII_PWTP_DR in 61 cases, while NSGAII_PWTP_DR surpassed *Cons*_2_3 in only 36 cases, and their performance was the same in three instances (10_3_2_15, 10_5_3_25, and 20_5_2_25). It is interesting to note that here the performance of a heuristic is being compared with that of a metaheuristic. The former is usually employed to create an initial set of solutions.

In contrast, the latter is a step-by-step procedure through which the population evolves using a variety of operators such as crossover, mutation, and local search, to name a few. It is further noted that as the number of cities and items increases, NSGAII_PWTP_DR tends to deliver superior outcomes, which is quite obvious. To further substantiate, a t-test for paired two sample mean is applied to the results obtained by *Cons*_2_3 and NSGAII_PWTP_DR at 5% $(\alpha = 0.05)$ level of significance to justify our claim. The calculated P value for two two-tailed t-tests is 0.0401, which is less the α. Therefore, we can infer that *Cons*_2_3 is the preferred choice for obtaining an initial set of solutions to be employed in a metaheuristic. At the same time, it alone is able to provide a good solution for smaller instances. The reason behind obtaining higher *HV* values by the proposed construction heuristics as compared to the metaheuristic NSGAII_PWTP_DR can be summarized as follows: In the latter case, three different heuristics are tested, and only the best one is employed to generate the initial population. However, this population is based on a single tour; therefore, it is not as diverse as desired. Even the crossover or mutation operators fail to improve its quality significantly. On the other hand, when three different heuristics are providing solutions to the population together, it is bound to be of good quality. The diversity of such a population is obviously high as the solutions are constructed using distinct tours. Even though one of the heuristics fails to provide non-dominated solutions to the pool, it helps maintain the diversity of the population. Both these factors together are responsible for the population with higher *HV* value without even being improved further by the evolutionary operators. Clearly thus, it is a cost-effective strategy to be used for finding good solutions of PWTP_DR. It is worth noting that the outcome of our experiments could also prove beneficial for other metaheuristics that may be developed for this problem in future.

6. Conclusion

In this paper, we have done a comparative study of state-of-the-art heuristics for the Packing While Travelling Problem with dropping rate. This work not only explores three known construction heuristics for PWTP_DR but also explores the outcomes of the heuristics through extensive experiments when they are utilized simultaneously to pump in the solutions to an initial population. For this comparison, three construction heuristics and their combinations are experimented with to choose the best among heuristics and the combination heuristics. It is also seen that a combination of heuristics (*Cons*_1_2_3 and *Cons*_2_3) serves a better purpose instead of using a standalone heuristic in a population because this combination surpasses the performance of the only available metaheuristic NSGAII_PWTP_DR.

It is evident that for larger instances, using metaheuristics is necessary to address such problems. However, employing a combination of construction heuristics is beneficial in generating a diverse initial population, which plays an important role in preventing premature convergence in the metaheuristic process. For future work, we intend to extend our study by designing more construction heuristics/metaheuristics and improving the Nadir and Ideal vectors. The comparative study may be extended to the metaheuristics, where the combination of heuristics will be used for the initial population.

References

[1] Mohammad Reza Bonyadi, Zbigniew Michalewicz, and Luigi Barone, “The Travelling Thief Problem: The First Step in the Transition from Theoretical Problems to Realistic Problems,” *2013 IEEE Congress on Evolutionary Computation*, Cancun, Mexico, pp. 1037-1044, 2013. [CrossRef] [Google Scholar] [Publisher Link]

[2] David Pisinger, “A Minimal Algorithm for the 0-1 Knapsack Problem,” *Operations Research*, vol. 45, no. 7, pp. 758-767, 1997. [CrossRef] [Google Scholar] [Publisher Link]

[3] David Applegate, William Cook, and André Rohe, “Chained Lin-Kernighan for Large Traveling Salesman Problems,” *Informs Journal on Computing*, vol. 15, no. 1, pp. 82-92, 2013. [CrossRef] [Google Scholar] [Publisher Link]

[4] S. Polyakovskiy, and F. Neumann, “The Packing while Travelling Problem,” *European Journal of Operational Research*, vol. 258, no. 2, pp. 424-439, 2017. [CrossRef] [Google Scholar] [Publisher Link]

[5] Sergey Polyakovskiy et al., “A Comprehensive Benchmark Set and Heuristics for The Travelling Thief Problem,” *GECCO ‘14: Proceedings of the 2014 Annual Conference on Genetic and Evolutionary Computation*, pp. 477-484, Vancouver BC Canada, 2014. [CrossRef] [Google Scholar] [Publisher Link]

[6] Julian Blank, Kalyanmoy Deb, and Sanaz Mostaghim, “Solving the Bi-objective Traveling Thief Problem with Multi-objective Evolutionary Algorithms,” *9th International Conference on Evolutionary Multi-Criterion Optimization, Evolutionary Multi-Criterion Optimization*, Münster, Germany, pp. 46-60, 2017. [CrossRef] [Google Scholar] [Publisher Link]

[7] Mohammad Reza Bonyadi et al., “Socially Inspired Algorithms for the Travelling Thief Problem,” *GECCO ‘14: Proceedings of the 2014 Annual Conference on Genetic and Evolutionary Computation*, Vancouver BC Canada, pp. 421-428, 2014. [CrossRef] [Google Scholar] [Publisher Link]

[8] Yi Mei, Xiaodong Li, and Xin Yao, “On Investigation of Interdependence Between Sub-Problem of the Travelling Thief Problem,” *Soft Computing*, vol. 20, pp. 157-172, 2016. [CrossRef] [Google Scholar] [Publisher Link]

[9] Junhua Wu et al., “Exact Approaches for the Travelling Thief Problem,” *11th International Conference Simulated Evolution and Learning*, Shenzhen, China, pp. 110-121, 2017. [CrossRef] [Google Scholar] [Publisher Link]

[10] Mohamed El Yafrani et al., “A Fitness Landscape Analysis of the Travelling Thief Problem,” *GECCO ‘18: Proceedings of the Genetic and Evolutionary Computation Conference*, Kyoto Japan, pp. 277-284, 2018. [CrossRef] [Google Scholar] [Publisher Link]

[11] Alenrex Maity, and Swagatam Das, “Efficient Hybrid Local Search Heuristics for Solving the Travelling Thief Problem,” *Applied Soft Computing*, vol. 93, 2020. [CrossRef] [Google Scholar] [Publisher Link]

[12] Daniel Rodríguez et al., “A Sequence-Based Hyper-Heuristic for Travelling Thieves,” *Applied Sciences*, vol. 13, no. 1, pp. 1-23, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[13] Junfeng Chen et al., “Influence of Subproblem Solutions on the Quality of Traveling Thief Problem Solutions,” *Journal of Intelligent and Fuzzy Systems*, vol. 44, no. 2, pp. 1943-1956, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[14] Junhua Wu et al., “Evolutionary Computation Plus Dynamic Programming for the Bi-Objective Travelling Thief Problem,” *GECCO ‘18: Proceedings of the Genetic and Evolutionary Computation Conference*, Kyoto, Japan, pp. 777-784, 2018. [CrossRef] [Google Scholar] [Publisher Link]

[15] Maciej Laszczyk, and Paweł B. Myszkowski, “A Specialized Evolutionary Approach to the bi-objective Travelling Thief Problem,” *Proceedings of the 2019 Federated Conference on Computer Science and Information Systems*, vol. 18, pp. 47-56, 2019. [CrossRef] [Google Scholar] [Publisher Link]

[16] Rani Kumari, and Kamal Srivastava, “Variable Neighbourhood Search for Bi-objective Travelling Thief Problem,” *2020 8th International Conference on Reliability, Infocom Technologies and Optimization (Trends and Future Directions) (ICRITO)*, Noida, India, pp. 47-51, 2020. [CrossRef] [Google Scholar] [Publisher Link]

[17] Jonatas B. C. Chagas et al., “A Non-Dominated Sorting Based Customized Random-Key Genetic Algorithm for the Bi-Objective Traveling Thief Problem,” *Journal of Heuristics*, vol. 27, pp. 267-301, 2020. [CrossRef] [Google Scholar] [Publisher Link]

[18] Jonatas B. C. Chagas, and Markus Wagner, “A Weighted-Sum Method for Solving the Bi-Objective Traveling Thief Problem,” *Computers and Operations Research*, vol. 138, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[19] Rani Kumari, and Kamal Srivastava, “NSGAII for Travelling Thief Problem with Dropping Rate,” *7th International Conference on Intelligent Computing and Control Systems (ICICCS)*, Madurai, India, 2023. [CrossRef] [Google Scholar] [Publisher Link]