
EVOLUTIONARY DIVERSITY OPTIMISATION FOR THE TRAVELING THIEF PROBLEM

Adel Nikfarjam

Optimisation and Logistics
School of Computer Science
The University of Adelaide
adel.nikfarjam@adelaide.edu.au

Aneta Neumann

Optimisation and Logistics
School of Computer Science
The University of Adelaide
aneta.neumann@adelaide.edu.au

Frank Neumann

Optimisation and Logistics
School of Computer Science
The University of Adelaide
frank.neumann@adelaide.edu.au

ABSTRACT

There has been a growing interest in the evolutionary computation community to compute a diverse set of high-quality solutions for a given optimisation problem. This can provide the practitioners with invaluable information about the solution space and robustness against imperfect modelling and minor problems' changes. It also enables the decision-makers to involve their interests and choose between various solutions. In this study, we investigate for the first time a prominent multi-component optimisation problem, namely the Traveling Thief Problem (TTP), in the context of evolutionary diversity optimisation. We introduce a bi-level evolutionary algorithm to maximise the structural diversity of the set of solutions. Moreover, we examine the inter-dependency among the components of the problem in terms of structural diversity and empirically determine the best method to obtain diversity. We also conduct a comprehensive experimental investigation to examine the introduced algorithm and compare the results to another recently introduced framework based on the use of Quality Diversity (QD). Our experimental results show a significant improvement of the QD approach in terms of structural diversity for most TTP benchmark instances.

Keywords Evolutionary diversity optimisation, multi-component optimisation problems, traveling thief problem

1 Introduction

Evolutionary Algorithms (EAs) are traditionally used to find a high quality solution, ideally optimum/near-optimum for a given optimisation problem. The diversification of a set of high-quality solutions has gained increasing attention in the literature of evolutionary computation in recent years. These studies are dominated mainly by multi-modal optimisation. They aim to explore the fitness landscape to find niches, usually through a diversity preservation mechanism. Moreover, several studies can be found focusing on exploring niches in a feature space. The paradigm is called Quality Diversity (QD), where the objective is to find a set of high-quality solutions that differ in terms of some user-defined features. QD has been mainly applied to the areas of robotics Rakicevic et al. [2021], Zardini et al. [2021], Cully [2020], and games Steckel and Schrum [2021], Fontaine et al. [2020, 2021].

Evolutionary Diversity Optimisation (EDO) is another concept in this area. In contrast to the previous paradigms, EDO explicitly seeks to maximise the structural diversity of solutions, generally subject to a quality constraint. Ulrich and Thiele [2011] first defined the outline of EDO. Afterwards, the concept has been utilised to evolve a diverse set of images and the Traveling Salesperson Problem's (TSP) instances in Alexander et al. [2017], Gao et al. [2021]. For the same purposes, the star discrepancy and indicators from the multi-objective optimisation frameworks have been studied in Neumann et al. [2018] and Neumann et al. [2019], respectively. The use of distance-based diversity measures and

entropy have been studied in Do et al. [2020], Nikfarjam et al. [2021b] to generate a diverse set of solutions for the TSP. Nikfarjam et al. [2021a] introduced a modified Edge Assembly Crossover (EAX) to achieve higher diversity in TSP tours. Moreover, EDO has been studied in the context of knapsack problem Bossek et al. [2021], minimum spanning tree problem Bossek and Neumann [2021], quadratic assignment problem Do et al. [2021] and the optimisation of monotone sub-modular functions Neumann et al. [2021].

Real-world optimisation problems often include several sub-problems interacting, where each sub-problem impacts not only the quality but also the feasibility of solutions of others. These kinds of problems are called multi-component optimisation problems. Traveling Thief Problem (TTP) can be classified into this category. TTP is the integration of TSP and the Knapsack Problem (KP), where the traveling cost between two cities depends on the distance between the cities and the weight of items collected Bonyadi et al. [2013]. A wide range of solution approaches have been proposed to TTP that includes co-evolutionary strategies Bonyadi et al. [2014], Yafrani and Ahiod [2015], swarm intelligence approaches Wagner [2016], Zouari et al. [2019], simulated annealing Yafrani and Ahiod [2018], and local search heuristics Polyakovskiy et al. [2014], Maity and Das [2020]. More recently, Nikfarjam et al. [2021c] introduced a Map-elite based algorithm to compute a set of high-quality solutions exploring niches in a feature space. They showed that the algorithm is capable of improving the best-known solutions for several benchmark instances.

1.1 Our Contribution

In this study, we investigate the EDO in the context of TTP. Several advantages can be found for having a diverse set of high-quality solutions for TTP. First, we can study the inter-dependency of the sub-problems in terms of structural diversity and find the best method to maximise it. Second, EDO provides us with invaluable insight into the solutions space. For example, it can show which elements of an optimal/near-optimal solution can be replaced easily and which elements are irreplaceable. Finally, it brings about robustness against the minor changes.

To the best of our knowledge, this study is the first to investigate EDO in the context of a multi-component problem. We first establish a method to calculate the structural diversity of TTP solutions. Then, we introduced a bi-level EA to maximise the diversity. The first level involves generating the TSP part of a TTP solution, whereby the second level is an inner algorithm to optimise the KP part of the solution with respect to the first part. Then, an EDO-based survival selection is exercised to maximise the diversity. We first examine the impact that incorporating different inner algorithms into the EA can make on the diversity of the solutions. Moreover, We empirically study the inter-dependency between the sub-problems and show how focusing on the diversity of one sub-problem affects the other's and determine the best method to obtain diversity. Interestingly, the results indicate that focusing on overall diversity brings about greater KP diversity than solely emphasising KP diversity. In addition, we compare the set of solutions obtained from the introduced algorithm with a recently developed QD-based EA in terms of structural diversity. The results show that the introduced bi-level EA can bring higher structural diversity for most test instances. We also conduct a simulation test to evaluate the robustness of populations obtained from the two algorithms against changes in the problem.

The remainder of the paper is structured as follows. We formally define the TTP problem and the diversity for a set of TTP solutions in Section 2. In Section 3, We introduce the two-stage EA. A comprehensive experimental investigation is conducted in Section 4. Finally, we finish with some concluding remarks.

2 Problem Definition

TTP is defined on the aggregation of the TSP and the KP. The TSP is formed by a complete graph $G = (V, E)$, where V is a set of cities, and E is pairwise edges that connect the cities. We denote the size of the cities set by $|V| = n$. There is also a non-negative distance $d(e)$ associated with each edge $e = (u, v) \in E$. In TSP, the goal is to find a permutation (a tour) $x : V \rightarrow V$ that minimise a cost function. The KP is defined on a set of items I , where $|I| = m$. Each item i associated with a weight w_i and a profit p_i . In KP, the objective is to find a selection of items $y = (y_1, \dots, y_m)$ that maximise profit complied with the wight of the selected items not exceeding a capacity of W . Note that y_i is a binary variable equal to 1 if i is included in packing list y ; otherwise is equal to 0.

The TTP is defined on the graph G and the set of items I . However, the items are scattered over the cities. Each city $j \in V \setminus v_1$ has a set of item M_j , where $M_j \subset I$. the thief visits each city exactly once and collect some items into the knapsack. Moreover, a rent of R should be paid for the knapsack per time unit, and ν_{\max} and ν_{\min} are the maximum and minimum speeds that the thief can travel, respectively. In the TTP, we aim to compute a solution including a tour x

and a packing list y maximising the following objective function $z(x, y)$:

$$z(x, y) = \sum_{j=1}^m p_j y_j - R \left(\frac{d(x_n, x_1)}{\nu_{max} - \nu W_{x_n}} + \sum_{i=1}^{n-1} \frac{d(x_i, x_{i+1})}{\nu_{max} - \nu W_{x_i}} \right)$$

subject to $\sum_{j=1}^m w_j y_j \leq W \quad y_j \in \{0, 1\}.$

where W_{x_i} is the cumulative weight of the items collected from the start of the tour up to city x_i , and $\nu = \frac{\nu_{max} - \nu_{min}}{W}$ is a constant.

This study aims to compute a diverse set of TTP solutions that all comply with a minimum quality threshold but differ in terms of structural properties. In other words, the objective is to maximise the diversity of the set of solutions subject to a quality constraint. Let denote the set of TTP solutions by $P = \{p_1, \dots, p_\mu\}$, where $|P| = \mu$. Therefore, we can formally formulate the problem as:

$$\begin{aligned} & \text{Maximize } H(P) \\ & \text{subject to} \\ & z_p \geq (1 - \alpha) z^* \quad \forall p \in P \\ & \sum_{j=1}^m w_j y_{jp} \leq W \quad \forall p \in P \quad y_{jp} \in \{0, 1\} \end{aligned}$$

Where $H(P)$ is a measure quantifying the diversity of P , z^* is the optimal or the best-known value of z for a given TTP instance, α is the acceptable quality threshold, and y_{jp} shows $y_j \in p$. In line with most of the studies in EDO literature, we assumed that the optimal or a high quality solution of TTP instances are already known.

2.1 Diversity in TTP

To maximise the diversity, we require a measure to quantify the diversity of a set of solutions. As mentioned, a TTP solution includes two different parts, a tour and a packing list. That means a function is required to calculate the structural diversity of tours and another one for packing lists. We adopt the well-known information-theoretic concept of entropy for this purpose.

We employ the diversity measure based on entropy from Nikfarjam et al. [2021b,a] to compute the entropy of the tours. Let P be a set of TTP solutions. Here, the diversity is defined on the proportion of edges including in $E(P)$, where $E(P)$ is the set of edges included in P . The edge entropy of P can be calculated from:

$$H_e(P) = \sum_{e \in E(P)} h(e) \text{ with } h(e) = - \left(\frac{f(e)}{2n\mu} \right) \ln \left(\frac{f(e)}{2n\mu} \right).$$

where $h(e)$ is the contribution of an edge $e \in E$ to the entropy, and $f(e)$ is the number of tours in P including e . The contribution of edges with zero frequency is equal to zero ($h(e) = 0 \iff f(e) = 0$). $2n\mu$ is the summation of the frequency of all edges over the population.

The same concept is adopted for calculation of the entropy of items. The diversity of packing list on the proportion of items being included in $P(I)$, where $P(I)$ is the set of items included in P . The item entropy of P can be compute from:

$$H_i(P) = \sum_{i \in P(I)} h(i) \text{ with } h(i) = - \left(\frac{f(i)}{\sum_{i \in I} f(i)} \right) \ln \left(\frac{f(i)}{\sum_{i \in I} f(i)} \right).$$

where $h(i)$ is the contribution of an item $i \in I$ to the entropy, and $f(i)$ is the number of packing lists in P including i . The contribution of items with zero frequency is equal to zero ($h(i) = 0 \iff f(i) = 0$).

A simple way to calculate the overall entropy is to sum up the entropy of edges and items. This is because H_e and H_i are basically the summation of contribution of edges and items. Therefor, we have: $H(P) = H_e(P) + H_i(P)$

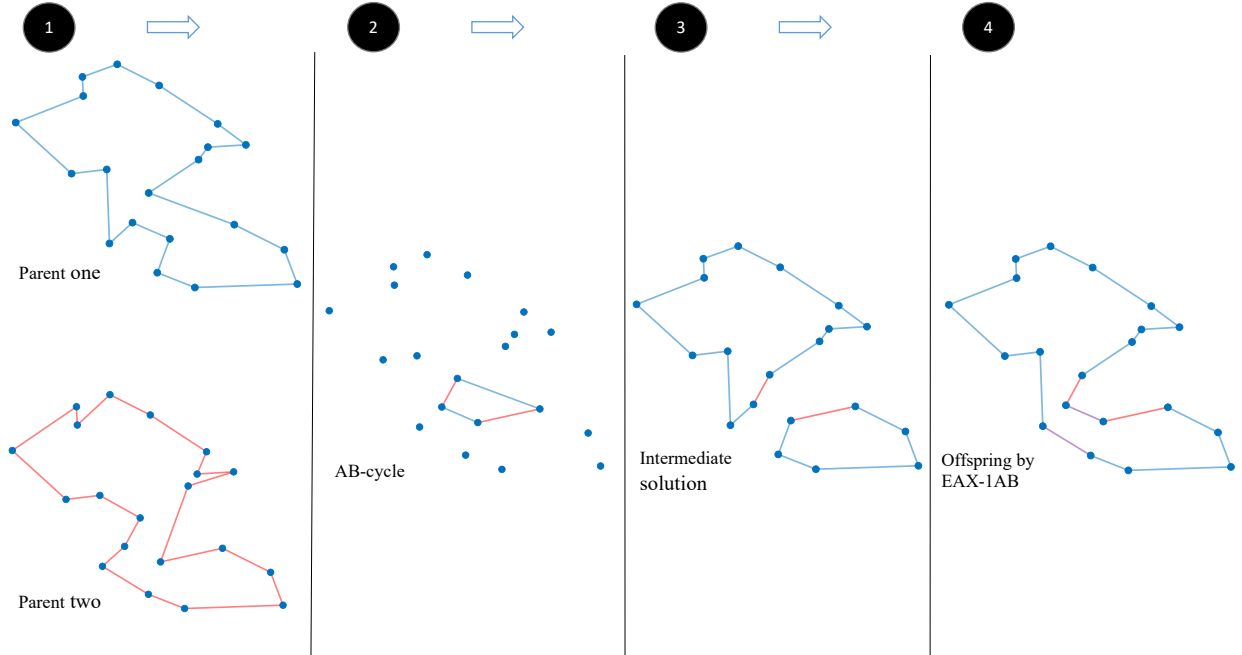


Figure 1: The representation of the steps to implement EAX Nikfarjam et al. [2021c]

3 Bi-level Evolutionary Algorithm

We introduce a bi-level EA to compute a diverse set of TTP solutions. The EA is started with an initial population that all individuals complying the quality constraint. The procedure to construct such a population will be explained later. Having selected two tours uniformly at random, the EA generates a new tour by crossover. Then, an inner algorithm is initiated to compute a corresponding packing list for the new tour in order to have a complete TTP solution; we refer the inner algorithms as the KP operators. If the TTP score of the new solution is higher than minimum requirement, it will be added to the population; otherwise, it will be discarded. Finally, an individual with minimum contribution to the diversity of population will be discarded if the size of population is $\mu + 1$. These steps are continued until a termination criterion is met. Algorithm 1 outlines the bi-level EA. In this study, we employ EAX as crossover and Dynamic Programming (DP) or alternatively $(1 + 1)$ EA as the KP operators. These operators are shown in Nikfarjam et al. [2021c] capable of computing high-quality TTP solutions efficiently.

3.1 The Edges Assembly Crossover (EAX)

The EAX is known to yield decent TSP tours and the GA using the EAX Nagata and Kobayashi [2013] is a high-performing EA in solving TSP. Several variations of the EAX can be found in the literature. This study utilises the EAX-1AB for its simplicity and efficiency compared to the other variants. Since we only use EAX-1AB, we refer it as the EAX. As shown in Figure 1, the crossover is formed by three steps as follows:

- **AB-cycle:** forming an AB-cycle from two tours by alternatively selecting edges from first and second tours until a cycle is formed (Fig 1.2).
- **Intermediate Solution:** Copying all edges from the first tour; then removing the Ab-cycle's edges belonging to the first tour, and adding the other edges of the AB-cycle. (Fig 1.3).
- **Completing the Tour:** Connecting sub-tours of the intermediate solution to have one complete tour (Fig 1.4).

We require a 4-tuples of edges to connect two sub-tours, one edge from each sub-tour to be removed and two new edges to connect each end of the discarded edges. We first select the sub-tour r with the minimum edge number. afterwards, we determine the 4-tuples of edges such that $\{e_1, e_2, e_3, e_4\} = \arg \min \{-d(e_1) - d(e_2) + d(e_3) + d(e_4)\}$. Note that $e_1 \in E(r)$ and $e_2 \in E(t) \setminus E(r)$, where $E(r)$ and $E(t)$ denote the set of edges included in sub-tour r and the

Algorithm 1 Two-stage-EA**Input:** Population P , minimal quality threshold c_{min}

- 1: **while** termination criterion is not met **do**
- 2: Choose x_1 and $x_2 \in P$ uniformly at random, and generate one tour x_3 by crossover.
- 3: Generate a corresponding packing list (y_3) by a KP operator to have a complete TTP solution ($p_3(x_3, y_3)$)
- 4: **if** $z(p_3) \geq z_{min}$ **then**
- 5: Add p_3 to P .
- 6: **if** $|P| = \mu + 1$ **then**
- 7: Remove one individual p from P , where $p = \arg \max_{q \in P} H(P \setminus \{q\})$.

intermediate solution t , respectively. The interested readers are referred to Nagata and Kobayashi [2013] for more details about implementation of the EAX.

3.2 The KP operators

Having generated a new tour, we employ a DP approach to compute a packing list for the generated tour and form a high-quality TTP solution. DP is classically utilised to solve the KP. Polyakovskiy et al. [2014] introduced a DP approach to solve the Packing While Traveling problem (PWT), which is a simplified variant of the TTP. the difference between the PWD and the TTP is that the tour is fixed in the PWD and a solution only includes a packing list. Here, we use the same DP to compute the optimal packing list for the tour generated by EAX.

The DP includes a table β size of $(W \times m)$. Here, we are processing the items based on the order of the appearance of their corresponding nodes in the tour. In other words, I_i is processed sooner than I_j if I_i belongs to a node visited prior I_j . If I_i and I_j belong to the same node, they are processed based on the their indices. The $\beta_{i,j}$ shows the maximal profit among all combinations of items I_k with $I_k \preceq I_i$ result in the weight exactly equal to j . If there are no combinations bringing about the weight j , $\beta_{i,j}$ is set to $-\infty$.

Let denote the profit of an empty set and the set including only item I_i by $B(\emptyset)$ and $B(I_i)$, respectively. For the first row of the table we have:

$$\beta_{i,0} = B(\emptyset), \quad \beta_{i,w_i} = B(I_i), \quad \beta_{i,j} = -\infty, \forall j \notin \{0, w_i\}$$

Let I_k be the predecessor of I_i . We can calculate $\beta_{i,j}$ from $\max(\beta_{k,j}, T)$, where T is computed as follows:

$$T = \beta_{k,j-w_i} + p_i - R \sum_{l=1}^n d_l \left(\frac{1}{\nu_{max} - \nu_j} - \frac{1}{\nu_{max} - \nu_j - w_i} \right)$$

The $\max_j \beta_{m,j}$ is the maximum profit that we can get from the given tour. As mentioned, the DP results in the optimal packing list for the given tour; However, the run-time of the DP is quit long. Moreover, The DP is an exact solver since we aim to increase the diversity of items, it would be interesting to compare the results of the DP with a random algorithm such as an EA. Thus, we introduce a simple $(1 + 1)$ EA as an alternative for the DP.

The $(1 + 1)$ EA is initialised with the packing list of the first parent used for generating the tour. Then, a new packing list is generated by mutation. If the new packing list results in a higher profit $z(x, y)$, it will be replaced with the old one; otherwise, it will be discarded. These steps are continued until a termination criterion is met. Here, we consider the bit-flip mutation. Each bit is independently filled by the probability of $(1/m)$.

3.3 Initial Population

As mentioned, we assumed that we know the optimal/near-optimal solution for given TTP instances; such an assumption is in line with most studies in the literature of EDO. The procedure is initialised with a single high-quality solution p in P , where $z(p) \in ((1 - \alpha)z^*, z^*)$. First, an individual $p \in P$ is selected uniformly at random. Then, the tour of the individual ($p(x)$) is mutated by 2-OPT, which is a well-known random neighborhood search in TSP. Afterwards, we compute a packing list y' by KP and match it with the mutated tour x' to have a TTP solution p' . If p' complies the quality constraint, it will be added to P ; otherwise, it is discarded. We continue these steps until $|P| = \mu$. Note that We used the algorithm introduced in Nikfarjam et al. [2021c] to obtain the initial p . Algorithm 2 outlines the initialising procedure.

Algorithm 2 Initial Population Procedure**Input:** A TTP solution p complying the quality criterion, population size μ

-
- ```

1: while $|P| < \mu$ do
2: Choose $p \in P$ uniformly at random, generate a tour x' by mutation.
3: Compute a packing list y' by the DP to form $p'(x', y')$
4: if $z(p') \geq z_{min}$ then
5: Add p' to P .
```
- 

| Number | Original Name                        |
|--------|--------------------------------------|
| 01     | eil51_n50_bounded-strongly-corr_01   |
| 02     | eil51_n150_bounded-strongly-corr_01  |
| 03     | eil51_n250_bounded-strongly-corr_01  |
| 04     | eil51_n50_uncorr-similar-weights_01  |
| 05     | eil51_n150_uncorr-similar-weights_01 |
| 06     | eil51_n250_uncorr-similar-weights_01 |
| 07     | eil51_n50_uncorr_01                  |
| 08     | eil51_n150_uncorr_01                 |
| 09     | eil51_n250_uncorr_01                 |
| 10     | pr152_n151_bounded-strongly-corr_01  |
| 11     | pr152_n453_bounded-strongly-corr_01  |
| 12     | pr152_n151_uncorr-similar-weights_01 |
| 13     | pr152_n453_uncorr-similar-weights_01 |
| 14     | pr152_n151_uncorr_01                 |
| 15     | pr152_n453_uncorr_01                 |
| 16     | a280_n279_bounded-strongly-corr_01   |
| 17     | a280_n279_uncorr-similar-weights_01  |
| 18     | a280_n279_uncorr_01                  |

Table 1: The names of the TTP instances are used in the paper.

## 4 Experimental Investigation

In this section, we conduct an comprehensive experimental investigation on the introduced framework to analyse the inter-dependency of the TTP's sub-problems in terms of structural diversity and find the best method to maximise it. First, we compare the two KP search operators, DP and  $(1 + 1)$ EA; then, we incorporate the  $H$ ,  $H_e$ , and  $H_i$  into the algorithm as the fitness function, and analyse the populations obtained. Finally, we conduct a comparison on the introduced framework with a recently introduced QD-based EA Nikfarjam et al. [2021c] in terms of structural diversity and robustness against small changes in availability of edges and items. In terms of experimental setting, we used 18 TTP instances from Polyakovskiy et al. [2014], and the algorithms are terminated after 10000 iterations. Table 1 shows the name of benchmarks instances. The internal termination criterion for the  $(1 + 1)$ EA is set to  $2m$  based on preliminary experiments. We consider 10 independent runs for each algorithms on each test instances.

### 4.1 Comparison in KP search operators operators

In this section, we compute two set of solutions for each test instance, one by use of DP and another with  $(1 + 1)$ EA, and scrutinise the diversity of the sets. Here,  $H$  serves as fitness function and  $\alpha$  and  $\mu$  are set to 0.1 and 50, respectively. Table 2 summarises the results. As Table 2 shows, the use of DP results in a population with higher diversity in edges ( $H_e$ ), while  $H_i$  is higher in the population obtained from  $(1 + 1)$ EA in most of cases. Turning to overall diversity ( $H$ ), the use of DP brings about populations with higher diversity in 4 out of 18 cases. On the other hand, there are 2 cases, that  $(1 + 1)$ EA outperforms the DP. There are found no significant differences in overall diversity for the rest of the test instances. One may ask the question why using DP results in a higher  $H_e$ , while EAX is used to generate new tours in both competitors. One explanation is that DP compute the same packing list for two identical tours. This is while,  $(1 + 1)$ EA can generate different packing lists which results in a higher  $H_i$  but a lower  $H_e$ .

Table 2: Comparison of the KP operators. In columns Stat the notation  $X^+$  means the median of the measure is better than the one for variant  $X$ ,  $X^-$  means it is worse, and  $X^*$  indicates no significant difference. Stat shows the results of Mann-Whitney U-test at significance level 5%

| Int | DP (1) |      | EA (2) |      | DP (1) |      | EA (2) |      | DP (1) |      | EA (2) |      |
|-----|--------|------|--------|------|--------|------|--------|------|--------|------|--------|------|
|     | $H$    | Stat | $H$    | Stat | $H_e$  | Stat | $H_e$  | Stat | $H_i$  | Stat | $H_i$  | Stat |
| 1   | 8.5    | 2*   | 8.3    | 1*   | 5.4    | 2-   | 5.7    | 1+   | 3      | 2+   | 2.6    | 1-   |
| 2   | 9      | 2+   | 8.8    | 1-   | 5.2    | 2+   | 5      | 1-   | 3.8    | 2*   | 3.8    | 1*   |
| 3   | 9.5    | 2+   | 9.3    | 1-   | 5.1    | 2+   | 5      | 1-   | 4.3    | 2*   | 4.4    | 1*   |
| 4   | 7.1    | 2*   | 7.2    | 1*   | 5.3    | 2+   | 5.2    | 1-   | 1.9    | 2*   | 2      | 1*   |
| 5   | 8.7    | 2+   | 8.3    | 1-   | 5.2    | 2+   | 5      | 1-   | 3.4    | 2+   | 3.3    | 1-   |
| 6   | 8.9    | 2+   | 8.8    | 1-   | 5.2    | 2+   | 5      | 1-   | 3.8    | 2*   | 3.8    | 1*   |
| 7   | 7.9    | 2*   | 7.8    | 1*   | 5.3    | 2*   | 5.3    | 1*   | 2.5    | 2*   | 2.5    | 1*   |
| 8   | 8.6    | 2*   | 8.6    | 1*   | 5      | 2*   | 5      | 1*   | 3.5    | 2-   | 3.6    | 1+   |
| 9   | 9.2    | 2*   | 9.2    | 1*   | 5.1    | 2*   | 5      | 1*   | 4.1    | 2-   | 4.1    | 1+   |
| 10  | 9.8    | 2*   | 9.8    | 1*   | 6      | 2+   | 5.9    | 1-   | 3.8    | 2-   | 3.9    | 1+   |
| 11  | 10.7   | 2-   | 10.7   | 1+   | 6      | 2*   | 6      | 1*   | 4.7    | 2*   | 4.7    | 1*   |
| 12  | 8.6    | 2-   | 8.8    | 1+   | 6      | 2*   | 5.9    | 1*   | 2.7    | 2-   | 2.8    | 1+   |
| 13  | 9.9    | 2*   | 10     | 1*   | 6      | 2+   | 5.8    | 1-   | 3.9    | 2-   | 4.2    | 1+   |
| 14  | 9.4    | 2*   | 9.4    | 1*   | 5.9    | 2*   | 5.9    | 1*   | 3.4    | 2*   | 3.5    | 1*   |
| 15  | 10.6   | 2*   | 10.6   | 1*   | 6      | 2*   | 6      | 1*   | 4.6    | 2-   | 4.6    | 1+   |
| 16  | 10.7   | 2*   | 10.7   | 1*   | 6.5    | 2+   | 6.4    | 1-   | 4.3    | 2-   | 4.4    | 1+   |
| 17  | 10.1   | 2*   | 10.1   | 1*   | 6.5    | 2+   | 6.4    | 1-   | 3.6    | 2*   | 3.8    | 1*   |
| 18  | 10.7   | 2*   | 10.7   | 1*   | 6.5    | 2+   | 6.4    | 1-   | 4.2    | 2-   | 4.2    | 1+   |

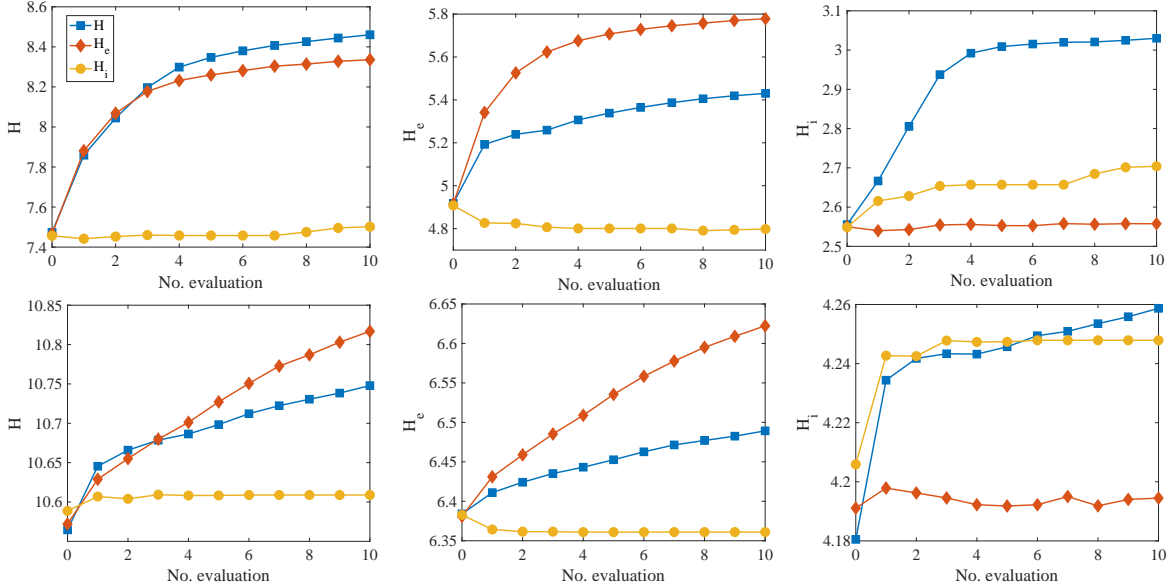


Figure 2: Representation of trajectories of incorporation of the different fitness functions,  $H$ ,  $H_e$ , and  $H_i$  over test instance 1 (first row), and test instance 16 (second row)

## 4.2 Comparison in fitness functions

Next, we investigate the use of  $H_e$  or  $H_i$  as the fitness functions instead of  $H$ . Table 3 compares three algorithms using the fitness functions where DP is used as KP operator. The table shows there are no significant differences in overall diversity when either of  $H$  or  $H_e$  serve as the fitness function. However, the EA using item diversity ( $H_i$ ) results in the populations with an overall entropy significantly less than the other EAs. It also gets outperformed by the EA using  $H$  in terms of items diversity. This is because the introduced framework is a bi-level optimisation procedure where it generates a tour first; then, it computes the packing list based on the tour. Therefore, the use of diversity in

Table 3: Comparison of different fitness function (DP used as the KP operator). Stat shows the results of Kruskal-Wallis statistical test at significance level 5% and Bonferroni correction. The notations are in line with Table 2

| Ins | $H$  | (1)  | $H_e$ | (2)  | $H_i$ | (3)  | $H$   | (1)  | $H_e$ | (2)  | $H_i$ | (3)  | $H$   | (1)  | $H_e$ | (2)  | $H_i$ | (3)  |
|-----|------|------|-------|------|-------|------|-------|------|-------|------|-------|------|-------|------|-------|------|-------|------|
|     | $H$  | Stat | $H$   | Stat | $H$   | Stat | $H_e$ | Stat | $H_e$ | Stat | $H_e$ | Stat | $H_i$ | Stat | $H_i$ | Stat | $H_i$ | Stat |
| 1   | 8.5  | 2*3+ | 8.3   | 1*3+ | 7.5   | 1-2- | 5.4   | 2-3+ | 5.8   | 1+3+ | 4.8   | 1-2- | 3     | 2+3+ | 2.6   | 1-3* | 2.7   | 1-2* |
| 2   | 9    | 2*3+ | 9.1   | 1*3+ | 8.6   | 1-2- | 5.2   | 2-3+ | 5.4   | 1+3+ | 4.8   | 1-2- | 3.8   | 2+3+ | 3.7   | 1-3* | 3.8   | 1-2* |
| 3   | 9.5  | 2*3+ | 9.6   | 1*3+ | 9.1   | 1-2- | 5.1   | 2-3+ | 5.3   | 1+3+ | 4.8   | 1-2- | 4.3   | 2*3* | 4.3   | 1*3* | 4.3   | 1*2* |
| 4   | 7.1  | 2*3+ | 6.8   | 1*3+ | 6.4   | 1-2- | 5.3   | 2*3+ | 5.4   | 1*3+ | 4.8   | 1-2- | 1.9   | 2+3* | 1.5   | 1-3* | 1.6   | 1*2* |
| 5   | 8.7  | 2+3+ | 8.3   | 1-3* | 7.7   | 1-2* | 5.2   | 2-3+ | 5.4   | 1+3+ | 4.8   | 1-2- | 3.4   | 2+3+ | 2.8   | 1-3* | 2.9   | 1-2* |
| 6   | 8.9  | 2*3+ | 8.7   | 1*3+ | 8.3   | 1-2- | 5.2   | 2-3+ | 5.3   | 1+3+ | 4.8   | 1-2- | 3.8   | 2+3+ | 3.4   | 1-3* | 3.5   | 1-2* |
| 7   | 7.9  | 2*3+ | 7.8   | 1*3+ | 7.3   | 1-2- | 5.3   | 2*3+ | 5.4   | 1*3+ | 4.8   | 1-2- | 2.5   | 2+3+ | 2.4   | 1-3* | 2.5   | 1-2* |
| 8   | 8.6  | 2-3+ | 8.7   | 1+3+ | 8.2   | 1-2- | 5     | 2-3+ | 5.2   | 1+3+ | 4.7   | 1-2- | 3.5   | 2+3+ | 3.5   | 1-3* | 3.5   | 1-2* |
| 9   | 9.2  | 2*3+ | 9.3   | 1*3+ | 8.8   | 1-2- | 5.1   | 2*3+ | 5.2   | 1*3+ | 4.7   | 1-2- | 4.1   | 2*3* | 4.1   | 1-3* | 4.1   | 1*2* |
| 10  | 9.8  | 2*3+ | 9.9   | 1*3+ | 9.6   | 1-2- | 6     | 2-3+ | 6.2   | 1+3+ | 5.8   | 1-2- | 3.8   | 2+3* | 3.7   | 1-3- | 3.8   | 1*2+ |
| 11  | 10.7 | 2*3+ | 10.8  | 1*3+ | 10.5  | 1-2- | 6     | 2*3+ | 6.1   | 1*3+ | 5.8   | 1-2- | 4.7   | 2+3* | 4.7   | 1-3* | 4.7   | 1*2* |
| 12  | 8.6  | 2*3+ | 8.6   | 1*3+ | 8.5   | 1-2- | 6     | 2*3+ | 6     | 1*3+ | 5.8   | 1-2- | 2.7   | 2+3* | 2.6   | 1-3- | 2.7   | 1*2+ |
| 13  | 9.9  | 2*3+ | 9.9   | 1*3+ | 9.7   | 1-2- | 6     | 2*3+ | 6.1   | 1*3+ | 5.8   | 1-2- | 3.9   | 2+3* | 3.8   | 1-3- | 3.9   | 1*2+ |
| 14  | 9.4  | 2*3+ | 9.4   | 1*3+ | 9.2   | 1-2- | 5.9   | 2*3+ | 6     | 1*3+ | 5.8   | 1-2- | 3.4   | 2+3* | 3.4   | 1-3- | 3.4   | 1*2+ |
| 15  | 10.6 | 2*3+ | 10.6  | 1*3+ | 10.4  | 1-2- | 6     | 2*3+ | 6     | 1*3+ | 5.8   | 1-2- | 4.6   | 2*3* | 4.6   | 1*3* | 4.6   | 1*2* |
| 16  | 10.7 | 2*3+ | 10.8  | 1*3+ | 10.6  | 1-2- | 6.5   | 2*3+ | 6.6   | 1*3+ | 6.4   | 1-2- | 4.3   | 2+3* | 4.2   | 1-3* | 4.2   | 1*2* |
| 17  | 10.1 | 2*3* | 9.9   | 1*3* | 9.9   | 1*2* | 6.5   | 2*3+ | 6.6   | 1*3+ | 6.4   | 1-2- | 3.6   | 2+3* | 3.4   | 1-3- | 3.6   | 1*2+ |
| 18  | 10.7 | 2*3+ | 10.8  | 1*3+ | 10.6  | 1-2- | 6.5   | 2*3+ | 6.6   | 1*3+ | 6.4   | 1-2- | 4.2   | 2+3* | 4.2   | 1-3- | 4.2   | 1*2+ |

Table 4: Comparison of different fitness function (EA used as the KP operator).The notations are in line with Table 3

| Ins | $H$  | (1)  | $H_e$ | (2)  | $H_i$ | (3)  | $H$   | (1)  | $H_e$ | (2)  | $H_i$ | (3)  | $H$   | (1)  | $H_e$ | (2)  | $H_i$ | (3)  |
|-----|------|------|-------|------|-------|------|-------|------|-------|------|-------|------|-------|------|-------|------|-------|------|
|     | $H$  | Stat | $H$   | Stat | $H$   | Stat | $H_e$ | Stat | $H_e$ | Stat | $H_e$ | Stat | $H_i$ | Stat | $H_i$ | Stat | $H_i$ | Stat |
| 01  | 8.3  | 2*3+ | 8.3   | 1*3+ | 7.5   | 1-2- | 5.7   | 2*3+ | 5.8   | 1*3+ | 4.9   | 1-2- | 2.6   | 2+3* | 2.5   | 1-3- | 2.6   | 1*2+ |
| 02  | 8.8  | 2*3+ | 9     | 1*3+ | 8.4   | 1-2- | 5     | 2-3+ | 5.3   | 1+3+ | 4.7   | 1-2- | 3.8   | 2+3* | 3.7   | 1-3- | 3.8   | 1*2+ |
| 03  | 9.3  | 2*3+ | 9.4   | 1*3+ | 9     | 1-2- | 5     | 2-3+ | 5.2   | 1+3+ | 4.7   | 1-2- | 4.4   | 2+3* | 4.2   | 1-3- | 4.3   | 1*2+ |
| 04  | 7.2  | 2+3+ | 6.9   | 1-3+ | 6.4   | 1-2- | 5.2   | 2-3+ | 5.4   | 1+3+ | 4.8   | 1-2- | 2     | 2+3+ | 1.5   | 1-3* | 1.6   | 1-2* |
| 05  | 8.3  | 2*3+ | 8.1   | 1*3+ | 7.7   | 1-2- | 5     | 2-3* | 5.4   | 1+3+ | 4.7   | 1-2- | 3.3   | 2+3+ | 2.7   | 1-3* | 3     | 1-2* |
| 06  | 8.8  | 2+3+ | 8.6   | 1-3* | 8.2   | 1-2* | 5     | 2-3+ | 5.3   | 1+3+ | 4.7   | 1-2- | 3.8   | 2+3* | 3.3   | 1-3- | 3.6   | 1*2+ |
| 07  | 7.8  | 2*3+ | 7.8   | 1*3+ | 7.3   | 1-2- | 5.3   | 2*3+ | 5.4   | 1*3+ | 4.8   | 1-2- | 2.5   | 2+3* | 2.4   | 1-3* | 2.5   | 1*2* |
| 08  | 8.6  | 2*3+ | 8.7   | 1*3+ | 8.3   | 1-2- | 5     | 2-3+ | 5.2   | 1+3+ | 4.7   | 1-2- | 3.6   | 2+3* | 3.5   | 1-3- | 3.6   | 1*2+ |
| 09  | 9.2  | 2*3+ | 9.3   | 1*3+ | 8.8   | 1-2- | 5     | 2-3+ | 5.2   | 1+3+ | 4.7   | 1-2- | 4.1   | 2+3* | 4.1   | 1-3- | 4.1   | 1*2+ |
| 10  | 9.8  | 2*3+ | 10    | 1*3+ | 9.6   | 1-2- | 5.9   | 2-3+ | 6.2   | 1+3+ | 5.7   | 1-2- | 3.9   | 2+3* | 3.8   | 1-3- | 3.9   | 1*2+ |
| 11  | 10.7 | 2*3+ | 10.8  | 1*3+ | 10.5  | 1-2- | 6     | 2*3+ | 6.1   | 1*3+ | 5.7   | 1-2- | 4.7   | 2*3* | 4.7   | 1*3- | 4.8   | 1*2+ |
| 12  | 8.8  | 2*3* | 8.7   | 1*3* | 8.7   | 1*2* | 5.9   | 2*3+ | 6     | 1*3+ | 5.7   | 1-2- | 2.8   | 2+3* | 2.6   | 1-3- | 2.9   | 1*2+ |
| 13  | 10   | 2+3* | 9.9   | 1-3* | 9.9   | 1*2* | 5.8   | 2-3+ | 6.1   | 1+3+ | 5.7   | 1-2- | 4.2   | 2+3* | 3.8   | 1-3- | 4.2   | 1*2+ |
| 14  | 9.4  | 2*3+ | 9.4   | 1*3+ | 9.2   | 1-2- | 5.9   | 2-3+ | 6     | 1+3+ | 5.8   | 1-2- | 3.5   | 2+3* | 3.4   | 1-3- | 3.5   | 1*2+ |
| 15  | 10.6 | 2*3+ | 10.6  | 1*3+ | 10.3  | 1-2- | 6     | 2*3+ | 6     | 1*3+ | 5.7   | 1-2- | 4.6   | 2+3* | 4.6   | 1-3* | 4.6   | 1*2* |
| 16  | 10.7 | 2*3* | 10.8  | 1*3+ | 10.7  | 1*2- | 6.4   | 2-3+ | 6.6   | 1+3+ | 6.3   | 1-2- | 4.4   | 2+3* | 4.2   | 1-3- | 4.4   | 1*2+ |
| 17  | 10.1 | 2+3* | 9.9   | 1-3- | 10.1  | 1*2+ | 6.4   | 2-3* | 6.6   | 1+3+ | 6.3   | 1*2- | 3.8   | 2+3* | 3.3   | 1-3- | 3.7   | 1*2+ |
| 18  | 10.7 | 2*3* | 10.7  | 1*3+ | 10.6  | 1*2- | 6.4   | 2*3+ | 6.6   | 1*3+ | 6.3   | 1-2- | 4.2   | 2+3* | 4.2   | 1-3- | 4.2   | 1*2+ |

edge can aid in increasing the diversity of items, especially where the EA uses DP. Figure 2 depicts the trajectories of the EAs using the three fitness functions over 10000 iterations in the test instances 1 and 16. The figure explicitly confirms the previous observations; using  $H_i$  as the fitness function makes the EA incapable of maximising overall and edge diversity. It also gets outperformed in terms of item entropy  $H_i$ . On the other hand, incorporating  $H_e$  as the fitness functions results in decent overall and edge diversity. However, it can not increase the item entropy. Figure 2 also shows the EAs using  $H$  and  $H_e$  as fitness function do not converge in 10000 iterations for the test instance 16 (a280\_n279\_bounded-strongly-corr\_01). Overall, if we aim to increase the total or edge diversity, using  $H_e$  as the fitness function would be better. This is because we achieve similar total diversity with the use  $H_e$ , but it results in



Table 5: Comparison of the EDO and QD (DP used as the KP operator). The notations are in line with Table 2.

| Int | EDO (1) |                | QD (2) |                | EDO (1) |                | QD (2) |                | EDO (1) |                | QD (2) |                |
|-----|---------|----------------|--------|----------------|---------|----------------|--------|----------------|---------|----------------|--------|----------------|
|     | $H$     | Stat           | $H$    | Stat           | $H_e$   | Stat           | $H_e$  | Stat           | $H_i$   | Stat           | $H_i$  | Stat           |
| 01  | 9.1     | 2 <sup>+</sup> | 8.1    | 1 <sup>-</sup> | 5.9     | 2 <sup>+</sup> | 5.1    | 1 <sup>-</sup> | 3.2     | 2 <sup>+</sup> | 3      | 1 <sup>-</sup> |
| 02  | 9.8     | 2 <sup>+</sup> | 9.1    | 1 <sup>-</sup> | 5.6     | 2 <sup>+</sup> | 5.1    | 1 <sup>-</sup> | 4.2     | 2 <sup>+</sup> | 3.9    | 1 <sup>-</sup> |
| 03  | 10.2    | 2 <sup>+</sup> | 9.5    | 1 <sup>-</sup> | 5.5     | 2 <sup>+</sup> | 5.1    | 1 <sup>-</sup> | 4.6     | 2 <sup>+</sup> | 4.4    | 1 <sup>-</sup> |
| 04  | 9.1     | 2 <sup>+</sup> | 7.7    | 1 <sup>-</sup> | 5.9     | 2 <sup>+</sup> | 5.2    | 1 <sup>-</sup> | 3.2     | 2 <sup>+</sup> | 2.5    | 1 <sup>-</sup> |
| 05  | 9.7     | 2 <sup>+</sup> | 8.7    | 1 <sup>-</sup> | 5.7     | 2 <sup>+</sup> | 5.1    | 1 <sup>-</sup> | 4       | 2 <sup>+</sup> | 3.5    | 1 <sup>-</sup> |
| 06  | 10.3    | 2 <sup>+</sup> | 9.2    | 1 <sup>-</sup> | 5.8     | 2 <sup>+</sup> | 5.1    | 1 <sup>-</sup> | 4.5     | 2 <sup>+</sup> | 4.1    | 1 <sup>-</sup> |
| 07  | 8.6     | 2 <sup>+</sup> | 7.8    | 1 <sup>-</sup> | 5.8     | 2 <sup>+</sup> | 5.1    | 1 <sup>-</sup> | 2.8     | 2 <sup>+</sup> | 2.7    | 1 <sup>-</sup> |
| 08  | 9.3     | 2 <sup>+</sup> | 8.8    | 1 <sup>-</sup> | 5.5     | 2 <sup>+</sup> | 5.1    | 1 <sup>-</sup> | 3.8     | 2 <sup>+</sup> | 3.7    | 1 <sup>-</sup> |
| 09  | 9.9     | 2 <sup>+</sup> | 9.3    | 1 <sup>-</sup> | 5.6     | 2 <sup>+</sup> | 5.1    | 1 <sup>-</sup> | 4.3     | 2 <sup>+</sup> | 4.2    | 1 <sup>-</sup> |
| 10  | 10      | 2 <sup>+</sup> | 9.9    | 1 <sup>-</sup> | 6.1     | 2 <sup>+</sup> | 6      | 1 <sup>-</sup> | 3.9     | 2 <sup>*</sup> | 3.9    | 1 <sup>*</sup> |
| 11  | 11.1    | 2 <sup>+</sup> | 11     | 1 <sup>-</sup> | 6.2     | 2 <sup>+</sup> | 6      | 1 <sup>-</sup> | 5       | 2 <sup>*</sup> | 5      | 1 <sup>*</sup> |
| 12  | 9.6     | 2 <sup>+</sup> | 9.5    | 1 <sup>-</sup> | 6.1     | 2 <sup>+</sup> | 6      | 1 <sup>-</sup> | 3.5     | 2 <sup>*</sup> | 3.5    | 1 <sup>*</sup> |
| 13  | 10.7    | 2 <sup>*</sup> | 10.6   | 1 <sup>*</sup> | 6.1     | 2 <sup>+</sup> | 6      | 1 <sup>-</sup> | 4.5     | 2 <sup>*</sup> | 4.6    | 1 <sup>*</sup> |
| 14  | 9.8     | 2 <sup>+</sup> | 9.5    | 1 <sup>-</sup> | 6.3     | 2 <sup>+</sup> | 6      | 1 <sup>-</sup> | 3.5     | 2 <sup>-</sup> | 3.6    | 1 <sup>+</sup> |
| 15  | 10.9    | 2 <sup>+</sup> | 10.8   | 1 <sup>-</sup> | 6.3     | 2 <sup>+</sup> | 6      | 1 <sup>-</sup> | 4.7     | 2 <sup>-</sup> | 4.8    | 1 <sup>+</sup> |
| 16  | 10.9    | 2 <sup>-</sup> | 11.2   | 1 <sup>+</sup> | 6.5     | 2 <sup>-</sup> | 6.6    | 1 <sup>+</sup> | 4.4     | 2 <sup>*</sup> | 4.5    | 1 <sup>*</sup> |
| 17  | 10.8    | 2 <sup>*</sup> | 10.7   | 1 <sup>*</sup> | 6.5     | 2 <sup>-</sup> | 6.7    | 1 <sup>+</sup> | 4.3     | 2 <sup>+</sup> | 4      | 1 <sup>-</sup> |
| 18  | 10.8    | 2 <sup>-</sup> | 11.1   | 1 <sup>+</sup> | 6.5     | 2 <sup>-</sup> | 6.7    | 1 <sup>+</sup> | 4.3     | 2 <sup>-</sup> | 4.4    | 1 <sup>+</sup> |

Table 6: Comparison of the EDO and QD (EA used as the KP operator). The notations are in line with Table 2.

| Int | EDO (1) |                | QD (2) |                | EDO (1) |                | QD (2) |                | EDO (1) |                | QD (2) |                |
|-----|---------|----------------|--------|----------------|---------|----------------|--------|----------------|---------|----------------|--------|----------------|
|     | $H$     | Stat           | $H$    | Stat           | $H_e$   | Stat           | $H_e$  | Stat           | $H_i$   | Stat           | $H_i$  | Stat           |
| 01  | 9       | 2 <sup>+</sup> | 8.1    | 1 <sup>-</sup> | 5.8     | 2 <sup>+</sup> | 5.1    | 1 <sup>-</sup> | 3.2     | 2 <sup>+</sup> | 3      | 1 <sup>-</sup> |
| 02  | 9.7     | 2 <sup>+</sup> | 9.1    | 1 <sup>-</sup> | 5.5     | 2 <sup>+</sup> | 5.1    | 1 <sup>-</sup> | 4.2     | 2 <sup>+</sup> | 3.9    | 1 <sup>-</sup> |
| 03  | 10.1    | 2 <sup>+</sup> | 9.5    | 1 <sup>-</sup> | 5.5     | 2 <sup>+</sup> | 5.1    | 1 <sup>-</sup> | 4.6     | 2 <sup>+</sup> | 4.4    | 1 <sup>-</sup> |
| 04  | 9       | 2 <sup>+</sup> | 7.7    | 1 <sup>-</sup> | 5.8     | 2 <sup>+</sup> | 5.2    | 1 <sup>-</sup> | 3.2     | 2 <sup>+</sup> | 2.5    | 1 <sup>-</sup> |
| 05  | 9.6     | 2 <sup>+</sup> | 8.7    | 1 <sup>-</sup> | 5.5     | 2 <sup>+</sup> | 5.1    | 1 <sup>-</sup> | 4.1     | 2 <sup>+</sup> | 3.5    | 1 <sup>-</sup> |
| 06  | 10.2    | 2 <sup>+</sup> | 9.2    | 1 <sup>-</sup> | 5.6     | 2 <sup>+</sup> | 5.1    | 1 <sup>-</sup> | 4.6     | 2 <sup>+</sup> | 4.1    | 1 <sup>-</sup> |
| 07  | 8.5     | 2 <sup>+</sup> | 7.8    | 1 <sup>-</sup> | 5.8     | 2 <sup>+</sup> | 5.1    | 1 <sup>-</sup> | 2.8     | 2 <sup>+</sup> | 2.7    | 1 <sup>-</sup> |
| 08  | 9.2     | 2 <sup>+</sup> | 8.8    | 1 <sup>-</sup> | 5.4     | 2 <sup>+</sup> | 5.1    | 1 <sup>-</sup> | 3.8     | 2 <sup>+</sup> | 3.7    | 1 <sup>-</sup> |
| 09  | 9.9     | 2 <sup>+</sup> | 9.3    | 1 <sup>-</sup> | 5.5     | 2 <sup>+</sup> | 5.1    | 1 <sup>-</sup> | 4.3     | 2 <sup>+</sup> | 4.2    | 1 <sup>-</sup> |
| 10  | 10      | 2 <sup>+</sup> | 9.9    | 1 <sup>-</sup> | 6       | 2 <sup>+</sup> | 6      | 1 <sup>-</sup> | 4       | 2 <sup>+</sup> | 3.9    | 1 <sup>-</sup> |
| 11  | 11.1    | 2 <sup>*</sup> | 11     | 1 <sup>*</sup> | 6       | 2 <sup>*</sup> | 6      | 1 <sup>*</sup> | 5       | 2 <sup>*</sup> | 5      | 1 <sup>*</sup> |
| 12  | 9.6     | 2 <sup>*</sup> | 9.5    | 1 <sup>*</sup> | 5.9     | 2 <sup>-</sup> | 6      | 1 <sup>+</sup> | 3.7     | 2 <sup>+</sup> | 3.5    | 1 <sup>-</sup> |
| 13  | 10.6    | 2 <sup>*</sup> | 10.6   | 1 <sup>*</sup> | 5.9     | 2 <sup>-</sup> | 6      | 1 <sup>+</sup> | 4.7     | 2 <sup>+</sup> | 4.6    | 1 <sup>-</sup> |
| 14  | 9.7     | 2 <sup>+</sup> | 9.5    | 1 <sup>-</sup> | 6.1     | 2 <sup>+</sup> | 6      | 1 <sup>-</sup> | 3.5     | 2 <sup>*</sup> | 3.6    | 1 <sup>*</sup> |
| 15  | 10.9    | 2 <sup>+</sup> | 10.8   | 1 <sup>-</sup> | 6.3     | 2 <sup>+</sup> | 6      | 1 <sup>-</sup> | 4.6     | 2 <sup>-</sup> | 4.8    | 1 <sup>+</sup> |
| 16  | 11      | 2 <sup>-</sup> | 11.2   | 1 <sup>+</sup> | 6.4     | 2 <sup>-</sup> | 6.6    | 1 <sup>+</sup> | 4.6     | 2 <sup>*</sup> | 4.5    | 1 <sup>*</sup> |
| 17  | 10.9    | 2 <sup>+</sup> | 10.7   | 1 <sup>-</sup> | 6.4     | 2 <sup>-</sup> | 6.7    | 1 <sup>+</sup> | 4.5     | 2 <sup>+</sup> | 4      | 1 <sup>-</sup> |
| 18  | 10.8    | 2 <sup>-</sup> | 11.1   | 1 <sup>+</sup> | 6.4     | 2 <sup>-</sup> | 6.7    | 1 <sup>+</sup> | 4.4     | 2 <sup>-</sup> | 4.4    | 1 <sup>+</sup> |

higher entropy in the edges, and more importantly, it requires less calculation. However,  $H$  works the best if we focus on the diversity of items or a more balanced diversity between items and edges.

Now, we conduct the same experiments with  $(1 + 1)EA$  to observe the changes in the results. Table 4 summarises the results for this round of experiments. Here, one can observe that using  $H$  slightly outperforms  $H_e$  if we aim for total diversity. The underlying reason is that DP is an exact algorithm that results in the same packing list for identical tours. Thus, identical tours have no contribution to the diversity of edges or items. This is while the  $(1 + 1)EA$  can return different packing lists for identical tours and contribute to the diversity of items and overall diversity. Thus, overall diversity is slightly higher when  $H$  is used as the fitness function when we incorporate  $(1 + 1)EA$  as the inner algorithm.

Table 7: Comparison of the robustness of the populations obtained from the EDO-based EA (1) and the QD-based EA (2). The  $E$  and  $I$  denotes the percentage of times the population has at least one alternative for the eliminated edges and item, respectively. The Stat notations are in line with Table 2.

| Int | EDO (1) |                | QD (2) |                | EDO (1) |                | QD (2) |                |
|-----|---------|----------------|--------|----------------|---------|----------------|--------|----------------|
|     | E       | Stat           | E      | Stat           | I       | Stat           | I      | Stat           |
| 1   | 99.6    | 2 <sup>+</sup> | 87.5   | 1 <sup>-</sup> | 70      | 2 <sup>+</sup> | 51.6   | 1 <sup>-</sup> |
| 2   | 98.2    | 2 <sup>+</sup> | 92     | 1 <sup>-</sup> | 60.7    | 2 <sup>+</sup> | 43.9   | 1 <sup>-</sup> |
| 3   | 97.3    | 2 <sup>+</sup> | 85.5   | 1 <sup>-</sup> | 56.2    | 2 <sup>+</sup> | 43.6   | 1 <sup>-</sup> |
| 4   | 99.8    | 2 <sup>+</sup> | 90.2   | 1 <sup>-</sup> | 51.2    | 2 <sup>+</sup> | 36.8   | 1 <sup>-</sup> |
| 5   | 99.8    | 2 <sup>+</sup> | 89     | 1 <sup>-</sup> | 41.6    | 2 <sup>+</sup> | 32.5   | 1 <sup>-</sup> |
| 6   | 99.6    | 2 <sup>+</sup> | 86.9   | 1 <sup>-</sup> | 43.8    | 2 <sup>+</sup> | 33.2   | 1 <sup>-</sup> |
| 7   | 98.4    | 2 <sup>+</sup> | 90.2   | 1 <sup>-</sup> | 30.4    | 2 <sup>+</sup> | 26.4   | 1 <sup>-</sup> |
| 8   | 98.2    | 2 <sup>+</sup> | 85.1   | 1 <sup>-</sup> | 28.7    | 2 <sup>+</sup> | 22.5   | 1 <sup>-</sup> |
| 9   | 99      | 2 <sup>+</sup> | 89.8   | 1 <sup>-</sup> | 28.6    | 2 <sup>+</sup> | 26.1   | 1 <sup>-</sup> |
| 10  | 64.1    | 2 <sup>+</sup> | 54.4   | 1 <sup>-</sup> | 27.7    | 2 <sup>-</sup> | 31.1   | 1 <sup>+</sup> |
| 11  | 61.5    | 2 <sup>+</sup> | 49.9   | 1 <sup>-</sup> | 30.8    | 2 <sup>*</sup> | 34.9   | 1 <sup>*</sup> |
| 12  | 67      | 2 <sup>+</sup> | 54.6   | 1 <sup>-</sup> | 25.4    | 2 <sup>-</sup> | 28.9   | 1 <sup>+</sup> |
| 13  | 68.8    | 2 <sup>+</sup> | 54     | 1 <sup>-</sup> | 23.5    | 2 <sup>-</sup> | 26.2   | 1 <sup>+</sup> |
| 14  | 67.8    | 2 <sup>+</sup> | 47     | 1 <sup>-</sup> | 7.7     | 2 <sup>-</sup> | 15.5   | 1 <sup>+</sup> |
| 15  | 71.4    | 2 <sup>+</sup> | 52.8   | 1 <sup>-</sup> | 7.7     | 2 <sup>-</sup> | 18.4   | 1 <sup>+</sup> |
| 16  | 29      | 2 <sup>-</sup> | 67.4   | 1 <sup>+</sup> | 24.8    | 2 <sup>-</sup> | 35.8   | 1 <sup>+</sup> |
| 17  | 31      | 2 <sup>-</sup> | 74.4   | 1 <sup>+</sup> | 31      | 2 <sup>+</sup> | 26.5   | 1 <sup>-</sup> |
| 18  | 33.4    | 2 <sup>-</sup> | 76.3   | 1 <sup>+</sup> | 16.7    | 2 <sup>-</sup> | 22     | 1 <sup>+</sup> |

### 4.3 Comparison of EDO and QD

We compare the introduced EDO-based framework with the QD-based EA in this section. We first run the QD-based EA for 10000 iterations. Then, we set the quality threshold to the minimum quality found in the population obtained by the EA and set  $\mu$  to the size of the set of solutions obtained. Having set the input parameters, we run the introduced EDO-based algorithm for the same number of iterations. Finally, we compare the two populations in terms of structural diversity ( $H$ ,  $H_e$ , and  $H_i$ ).

In line with the previous section, we first employ DP as the KP operator; then,  $(1 + 1)$ EA is replaced with DP to analyse the impact of using different KP search operators in the results. Table 5 shows the results when DP is employed. Compared to the QD-based algorithm, the introduced EA results in a higher  $H$ ,  $H_e$ , and  $H_i$  in 14, 15, and 9 cases out of 18, respectively. Table 6 summarises the results when  $(1 + 1)$ EA is used as the KP operator. The results are almost in line with Table 5. Here, the performance of EDO-based EA improves in increasing entropy of items, while it deteriorates in overall and edge diversity. The table shows that the number of cases in favour of EDO-based EA increases to 13 cases taking  $H_i$  into account. On the other hand, there is a fall of 1 and 3 cases in terms of  $H$  and  $H_i$ , respectively.

Furthermore, we conduct an experiment to test the robustness of populations obtained from the EDO and QD-based EAs against changes in the availability of edges and items. In this series of experiments, we make an edge of the best solution of the population unavailable and look into the population to check if there is a solution not using the excluded edge. For items, we look for solutions behaving the opposite of the best solution. For example, if item  $i$  is included in the packing list of the best solution, we check if there is a solution excluding the item  $i$ , and vice versa. We repeat the experiments for all edges and items of the best solution. Table 7 summarises the results of the robustness experiment. The results show the EDO-based EA results in more robust sets of solutions. In the edges entropy  $H_e$ , it strongly outperforms the QD-based EA in 15 out of 18 test instances, while the figure is 10 for the entropy of items  $H_i$ . The results of the small instances (the first 9) where the EDO-based EA converges in 10000 iterations indicate that EDO-based EA can provide a highly robust set of solutions if given sufficient time. Also, we can improve the robustness in edges if we alter the focus on the diversity of edges by using  $H_e$  as the fitness function; however, the robustness in items is likely to decrease in this case.

## 5 Conclusion

We introduced a framework to generate a set of high-quality TTP solutions differing in structural diversity. We examined the inter-dependency of TTP's sub-problems, TSP and KP, and determined the best method to achieve a highly diverse set of solutions. Moreover, We empirically analysed the introduced framework and compared the results with a recently-developed QD-based algorithm in terms of diversity. The results showed a considerable improvement in the diversity of the population compared to the QD-based algorithm. Finally, we conduct a simulation test to evaluate the robustness of the population obtained from the two frameworks.

For future study, it is intriguing to incorporate indicators from multi-objective optimisation frameworks into the algorithm to focus on diversities of edges and items and compare them to the incumbent method. Moreover, several multi-component real-world problems such as patient admission scheduling problem and vehicle routing problem, can be found in the literature, where a set of diverse solutions is beneficial.

## 6 Acknowledgements

This work has been supported by the Australian Research Council (ARC) through grants DP190103894, FT200100536, and by the South Australian Government through the Research Consortium “Unlocking Complex Resources through Lean Processing”.

## References

- B. Alexander, J. Kortman, and A. Neumann. Evolution of artistic image variants through feature based diversity optimisation. In *GECCO*, pages 171–178. ACM, 2017.
- M. R. Bonyadi, Z. Michalewicz, and L. Barone. The travelling thief problem: The first step in the transition from theoretical problems to realistic problems. In *IEEE Congress on Evolutionary Computation*, pages 1037–1044. IEEE, 2013.
- M. R. Bonyadi, Z. Michalewicz, M. R. Przybylek, and A. Wierzbicki. Socially inspired algorithms for the travelling thief problem. In *GECCO*, pages 421–428. ACM, 2014.
- J. Bossek and F. Neumann. Evolutionary diversity optimization and the minimum spanning tree problem. In *GECCO*, pages 198–206. ACM, 2021.
- J. Bossek, A. Neumann, and F. Neumann. Breeding diverse packings for the knapsack problem by means of diversity-tailored evolutionary algorithms. In *GECCO*, pages 556–564. ACM, 2021.
- A. Cully. Multi-emitter map-elites: Improving quality, diversity and convergence speed with heterogeneous sets of emitters. *CoRR*, abs/2007.05352, 2020.
- A. V. Do, J. Bossek, A. Neumann, and F. Neumann. Evolving diverse sets of tours for the travelling salesperson problem. In *GECCO*, pages 681–689. ACM, 2020.
- A. V. Do, M. Guo, A. Neumann, and F. Neumann. Analysis of evolutionary diversity optimisation for permutation problems. In *GECCO*, pages 574–582. ACM, 2021.
- M. C. Fontaine, J. Togelius, S. Nikolaidis, and A. K. Hoover. Covariance matrix adaptation for the rapid illumination of behavior space. In *GECCO*, pages 94–102. ACM, 2020.
- M. C. Fontaine, R. Liu, A. Khalifa, J. Modi, J. Togelius, A. K. Hoover, and S. Nikolaidis. Illuminating mario scenes in the latent space of a generative adversarial network. In *AAAI*, pages 5922–5930. AAAI Press, 2021.
- W. Gao, S. Nallaperuma, and F. Neumann. Feature-based diversity optimization for problem instance classification. *Evol. Comput.*, 29(1):107–128, 2021.
- A. Maity and S. Das. Efficient hybrid local search heuristics for solving the travelling thief problem. *Appl. Soft Comput.*, 93:106284, 2020.
- Y. Nagata and S. Kobayashi. A powerful genetic algorithm using edge assembly crossover for the traveling salesman problem. *INFORMS J. Comput.*, 25(2):346–363, 2013.
- A. Neumann, W. Gao, C. Doerr, F. Neumann, and M. Wagner. Discrepancy-based evolutionary diversity optimization. In *GECCO*, pages 991–998. ACM, 2018.
- A. Neumann, W. Gao, M. Wagner, and F. Neumann. Evolutionary diversity optimization using multi-objective indicators. In *GECCO*, pages 837–845. ACM, 2019.

- A. Neumann, J. Bossek, and F. Neumann. Diversifying greedy sampling and evolutionary diversity optimisation for constrained monotone submodular functions. In *GECCO*, pages 261–269. ACM, 2021.
- A. Nikfarjam, J. Bossek, A. Neumann, and F. Neumann. Computing diverse sets of high quality TSP tours by eax-based evolutionary diversity optimisation. In *FOGA*, pages 9:1–9:11. ACM, 2021a.
- A. Nikfarjam, J. Bossek, A. Neumann, and F. Neumann. Entropy-based evolutionary diversity optimisation for the traveling salesperson problem. In *GECCO*, pages 600–608. ACM, 2021b.
- A. Nikfarjam, A. Neumann, and F. Neumann. On the use of quality diversity algorithms for the traveling thief problem. *CoRR*, abs/2112.08627, 2021c.
- S. Polyakovskiy, M. R. Bonyadi, M. Wagner, Z. Michalewicz, and F. Neumann. A comprehensive benchmark set and heuristics for the traveling thief problem. In *GECCO*, pages 477–484. ACM, 2014.
- N. Rakicevic, A. Cully, and P. Kormushev. Policy manifold search: exploring the manifold hypothesis for diversity-based neuroevolution. In *GECCO*, pages 901–909. ACM, 2021.
- K. Steckel and J. Schrum. Illuminating the space of beatable lode runner levels produced by various generative adversarial networks. In *GECCO Companion*, pages 111–112. ACM, 2021.
- T. Ulrich and L. Thiele. Maximizing population diversity in single-objective optimization. In *GECCO*, pages 641–648. ACM, 2011.
- M. Wagner. Stealing items more efficiently with ants: A swarm intelligence approach to the travelling thief problem. In *ANTS Conference*, volume 9882 of *Lecture Notes in Computer Science*, pages 273–281. Springer, 2016.
- M. E. Yafrani and B. Ahiod. Cosolver2b: An efficient local search heuristic for the travelling thief problem. In *AICCSA*, pages 1–5. IEEE Computer Society, 2015.
- M. E. Yafrani and B. Ahiod. Efficiently solving the traveling thief problem using hill climbing and simulated annealing. *Inf. Sci.*, 432:231–244, 2018.
- E. Zardini, D. Zappetti, D. Zambrano, G. Iacca, and D. Floreano. Seeking quality diversity in evolutionary co-design of morphology and control of soft tensegrity modular robots. In *GECCO*, pages 189–197. ACM, 2021.
- W. Zouari, I. Alaya, and M. Tagina. A new hybrid ant colony algorithms for the traveling thief problem. In *GECCO (Companion)*, pages 95–96. ACM, 2019.