Swarm and Evolutionary Computation 84 (2024) 101433

Contents lists available at ScienceDirect

Swarm and Evolutionary Computation

journal homepage: www.elsevier.com/locate/swevo

A comparative study of evolutionary approaches to the bi-objective dynamic Travelling Thief Problem

Daniel Herring [a,b,*], Michael Kirley [b], Xin Yao [a]

[a] School of Computer Science, University of Birmingham, Edgbaston, Birmingham, B15 2TT, UK
[b] School of Computing and Information Systems, University of Melbourne, Parkville, Melbourne, 3010, Victoria, Australia

ARTICLE INFO

Keywords:
Combinatorial optimization
Dynamic multi-objective optimization
Travelling Thief Problem
Evolutionary algorithms

ABSTRACT

Dynamic evolutionary multi-objective optimization is a thriving research area. Recent contributions span the development of specialized algorithms and the construction of challenging benchmark problems. Here, we continue these research directions through the development and analysis of a new bi-objective problem, the dynamic Travelling Thief Problem (TTP), including three modes of dynamic change: city locations, item profit values, and item availability. The interconnected problem components embedded in the dynamic problem dictate that the effective tracking of good trade-off solutions that satisfy both objectives throughout dynamic events is non-trivial. Consequently, we examine the relative contribution to the non-dominated set from a variety of population seeding strategies, including exact solvers and greedy algorithms for the knapsack and tour components, and random techniques. We introduce this responsive seeding extension within an evolutionary algorithm framework. The efficacy of alternative seeding mechanisms is evaluated across a range of exemplary problem instances using ranking-based and quantitative statistical comparisons, which combines performance measurements taken throughout the optimization. Our detailed experiments show that the different dynamic TTP instances present varying difficulty to the seeding methods tested. We posit the dynamic TTP as a suitable benchmark capable of generating problem instances with different controllable characteristics aligning with many real-world problems.

1. Introduction

Finding solutions to a problem with multiple and conflicting objective functions is a challenging task. This is made more difficult when we consider time-varying aspects of the problem as in dynamic multi-objective optimization problems (DMOPs). In the evolutionary computation community, a great deal of progress has been made in both single objective problems [1–4], dynamic multi-objective optimization problems (DMOPs), and specialized algorithms with which to solve them [5–8]. Typically, benchmark problem suites have been used to evaluate the efficacy of alternative algorithms [9–16]. Such contributions play an important role in progressing our understanding. However, we argue that intrinsic complexity embedded in many real-world problems is often missing from the benchmark suites examined, especially in the dynamic multi-objective optimization domain.

Dynamic changes in multi-objective optimization problems can be categorized in a number of ways. Usually, dynamic classifications are based on their effect on the Pareto Optimal Set (POS) and/or the Pareto Optimal Front (POF) [9] and on their magnitude, frequency and recurrence [17]. Farina et al. posits four categories for dynamics that change (I) only the POS, (II) both the POS and POF, (III) only the POF and (IV) neither. Many of the continuous benchmark problem suites (e.g., [9–16]) include examples from Farina's categories. Noteworthy examples of realistic problems with Type II dynamics include those mentioned in [11]; indoor heating control, greenhouse control and hospital resource management problems. In contrast, there is a paucity of complex dynamic combinatorial multi-objective problem benchmark suites. Previous examples are limited to the dynamic multi-objective Travelling Salesman Problem (DMTSP) instances in [18] and in [9], and the dynamic Knapsack Problem (DKP) in [9]. The work of [19] examines a real-world scenario and is close to this category, however the dynamic intervals in the problem are treated as separate instances solved in succession.

In this paper, we provide an important step towards understanding the efficacy of evolutionary algorithms for challenging, dynamic combinatorial multi-objective problems with real-world characteristics. We start by defining the bi-objective *Dynamic Travelling Thief Problem* (DTTP) with three modes of dynamic change: city locations, item profit values, and item availability. Specifically, we adapt and extend the

* Corresponding author at: School of Computer Science, University of Birmingham, Edgbaston, Birmingham, B15 2TT, UK.
E-mail address: d.g.herring@bham.ac.uk (D. Herring).

https://doi.org/10.1016/j.swevo.2023.101433
Received 12 August 2022; Received in revised form 2 October 2023; Accepted 12 November 2023
Available online 27 November 2023

well-known Travelling Thief problem (TTP) [20], defining the dynamic components and controlling the severity and frequency of change in the TSP (Travelling Salesperson Problem) and KP (Knapsack Problem) sub-problems. Importantly, our DTTP definition provides a challenging, but realistic framework, with controllable Type-II dynamics allowing for different instantiations of problem instances to be generated. These instances closely model optimization scenarios akin to problems in waste collection and mission planning, where dynamic events such as emergencies, traffic and weather are important.

It is widely accepted that mechanisms that encourage population diversity are an important feature of most dynamic evolutionary multi-objective algorithms [21,22]. When a change is detected, new or previously identified good solutions are often reintroduced into the evolving population to increase the diversity [7]. This is typically done using randomly reinitializing, parameter tuning, memory-based reinitializing, or prediction-based reinitializing [21,22]. More generally, seeding the evolving population with engineered solutions, rather than randomly generated ones, has been shown to be effective [23,24].

For the DTTP, domain-specific seeding of the evolving population raises many challenges as a direct consequence of the interconnected nature of the TSP and KP components. Whilst efficient exact solvers exist for the TSP and KP, no exact solver exists to find optimal TTP solutions. Despite this, many works follow the intuition that optimal solutions to the TSP and KP components provide a better starting point that random initialization [25–27].

Our approach to determine the impacts of biased population seeding for optimizing a DTTP instance systematically examines the possible approaches. Specifically, we contrast the performance of seeding the population using the following approaches for KP and TSP components respectively: an exact solver-based solution; a greedily constructed solution; a randomized solution; or combinations of these approaches. The seeding solutions are introduced into the population after dynamic changes for a range of DTTP instances with three different types of dynamics. We also compare the seeding approach with some TTP methods adapted for the DTTP. Analysis uses ranking-based and quantitative statistical comparisons of the post-change algorithm performance to compare the benefits of different seeding strategies.

Our contributions can be summarized: we present three dynamic formulations of the TTP based on the DTSP, the DKP and by dynamically changing the TTP's availability map. We investigate the initialization performance bias visible from exploiting problem knowledge by solving the TSP and KP solution components *a priori*. Furthermore, we determine the benefits of reactive population seeding with differently constructed solution sets for the proposed dynamic formulations. We provide a quantification of comparative performance that observes statistically significant differences at each time step to understand these benefits. A comparison with adapted static TTP methods enables the relative suitability of the seeding mechanisms as a dynamic response to be established.

The paper is organized as follows: Section 2 discusses DMOPs generally, the TTP and previous approaches for the static problem. Section 3 gives the definition of the DTTP problems and Section 4 presents our novel solution for the DTTP. Section 5 provides the results of computational experiments and conclusions are given in Section 7.

2. Background

2.1. Dynamic multi-objective evolutionary algorithms

Dynamic multi-objective optimization problems extend multi-objective problems to include time-variant parts, often to capture real-world behaviours. Eq. (1) gives one formulation of a DMOP with time-dependent objective functions.

$$\begin{aligned}\vec{\mathbf{x}} &= [x_1, x_2, \dots, x_n]\\ \vec{\mathbf{F}}(\vec{\mathbf{x}}, t) &= [f_1(\vec{\mathbf{x}}, t), f_2(\vec{\mathbf{x}}, t), \dots, f_M(\vec{\mathbf{x}}, t)] \\ h_1(\vec{\mathbf{x}}) &\le 0, h_2(\vec{\mathbf{x}}) = 0\end{aligned} \tag{1}$$

Dynamics can also be present in the decision variables or constraints (e.g. $x(t)$ or $h(\vec{\mathbf{x}}, t)$) and in the number of any of these.

The majority of works on DMOPs consider continuous problem domains, with fewer combinatorial problems considered despite the applicability to important sectors such as mobile and network communications. A few works consider the DMO-TSP problem [18,28] and in [9] a dynamic Knapsack formulation is presented. The work by [19] considers a dynamic power management example.

Many works exist on studying the properties of DMOPs [29]; on benchmarks [9,11–14,16], performance metrics [30,31], detection of changes [32] and algorithms to solve these types of problems [5,6,8]. A common design feature of most population-based algorithms for DMOPs is to exploit existing progress and to introduce new or previous solutions [7] to increase the diversity [21,22]. The introduced solutions can be constructed using predictions based on the nature of the changes [17,23,24,33]. Exploitation of problem or change knowledge has been effectively used in responses for dynamic problems. The TTP [20], which provides the basis for the dynamic bi-objective TTP presented here, exhibits exploitability by its definition.

2.2. The bi-objective travelling thief problem

The TTP is a superposition of Travelling Salesman and Knapsack problem components, interconnected by the behaviours of the 'thief'. The thief completes a Hamiltonian tour of the cities (x) and must collect items distributed across the cities according to a packing plan (z). The components are connected through knapsack-usage-dependent velocity and time-dependent degradation of item profits. The conflicting objectives are given in Eq. (2) and reflect a minimization of tour-time and maximization of item profit.

$$G(x,z) = \begin{cases} min \quad f(x,z) = \sum_{i=1}^{n-1} (t_{x_i,x_{i+1}}) + t_{x_n,x_1} \\ \qquad x = (x_1, \dots, x_n) \\ max \quad g(x,z) = \sum_{j \in z} p_j Dr^{\left\lceil \frac{T_j}{C} \right\rceil} \end{cases} \tag{2}$$

where:

$$t_{x_i,x_{i+1}} = \frac{d_{x_i,x_{i+1}}}{v_c}, \qquad v_c = (v_{max} - W_c \frac{v_{max} - v_{min}}{W}) \tag{3}$$

and:

$$C = \frac{ln\,(Dr) * E_t}{v_{min} * ln\left(\frac{rl}{u}\right)} \tag{4}$$

where $f(x,z)$ is the travel time of the tour accounting for item selection; $g(x,z)$ is the sum of the selected items' profits at the end of the tour; $d_{x_i,x_{i+1}}$ is the Euclidean distance between successive cities in the tour permutation; v_c is the current speed of travel; W_c and W are the current weight and maximum capacity of the thief's knapsack; v_{min} and v_{max} are the minimum and maximum travel velocity (0.1, 1); Dr is the dropping rate (0.9 - this is the profit degradation factor); T_j and p_j are the total time item j is carried during the tour and its profit value respectively. The constant C is calculated using the equation in [20] with $r = 0.45$ so as to generate reproducible results. Here, l and u are the minimum and maximum profit values across all items and E_t represents the shortest inter-city distance in the distance matrix D.

The distribution of items in the *availability map* is defined as part of the problem instance (see Section 4). A single solution is comprised of a permutation of the city indexes as a tour (x) and a KP solution (y) that respects the capacity constraint W. This is converted into a packing plan (z) based on the availability map.

Studies addressing DTTPs are greatly limited [34,35] despite the range of possible formulations and utility of generating a realistic and complex dynamic combinatorial problem.

Many meta-heuristic approaches have however been applied to the TTP. These works use simulated annealing and hill climbing [36] or evolutionary algorithms [25,37–39], including MATLS [40]. Ant Colony Optimization [41,42], Co-operative Coevolution [38,43] and Local Search [39] methods have also been applied to the single objective TTP. More information on all these algorithms can be found in [44]. Mei et al. [38] provides theoretical and empirical research on the interconnectedness of the problem components. As the objective function is not additively separable into the TSP and KP components, combination of components solutions solved in isolation is less effective than consideration of the whole TTP problem, as posited in the definitive work [20]. Despite this, Faulkner et al. [26] defines approximate heuristic methods, including the S5 heuristic, for solving the single objective TTP that are initialized with tours using the Chained Lin-Kernighan TSP solver [45]. Exact methods have also been employed in [27] to determine the performance, by comparison, to a range of approximate methods applied to TTP problems. Similarly, Dynamic Programming (DP) is used to find an optimal packing plan in the fixed tour scenario of the TTP: the Packing While Travelling (PWT) problem [46–48].

Relatively few works address the bi-objective version of the problem [25,48–51]. Blank et al.'s approach uses solvers combined with low level heuristics to provide solutions to a limited set of TTP instances [25]. Despite the reported applicable range of TTP instance sizes for EAs [39], the existing literature focuses on problems with 101 cities or fewer. The work of Mei et al. [38], specifically addresses large scale problems with at least 10,000 cities and 1,000,000 items, whilst the ACO-based MMAS approach in [42] is applied to instances with at most 1000 cities and 10,000 items. Informed analysis over a range of problems will provide insights for the proposed dynamic versions of TTP instances. Recent works by Chagas and co-authors [50,51] develops powerful algorithms that perform well on a selection of problems with between 280 and 33,810 cities.

3. The Dynamic Travelling Thief Problem (DTTP)

Three novel types of Type II dynamics (affecting POS and POF) with fixed magnitude and frequency are proposed. We introduce formulations of the DTTP in [34] and recent work examines formulations for the single objective TTP and considering single change events [35].

The city locations, item availability and the item values are examples of potential dynamic features reflecting realistic changes. Each dynamic modification is studied in isolation to enable a fundamental understanding of their impacts.

As before, a solution to the TTP takes the form of a tour (x) and a packing plan (z). The tour component is evaluated according to the objective functions in Eq. (2) using a symmetrical distance matrix, D, of Euclidean distances between cities. The packing plan is comprised of a sequence of items collected from specific cities during the tour; the location of each item is given by the problem-specific availability map (A). As with classical KP, each item (I) has a weight (I_w) and a profit (I_v) value.

3.1. Motivating real world scenarios

Motivating cases for the DTTP exist across logistics and routing domains that are relevant for both small and larger instances. For each of the following examples the dynamics and bi-objective motivations are included. In optimizing fishing catches, a vessel may need to retrieve devices (e.g. lobster pots) or catch or measure fish across many locations. Location changes are logical: shoals/pots may move or drift; availability changes may come from measurement updates on stock levels; value changes may come from the buy price or quality of the catch.

For drone-based item collection, a door-step item collection service may require visiting many locations if users do not need to actively request a pickup. Dynamics may present if users request pickup from different location (work instead of home), if the type of item is altered (food vs. mail) or if there is urgency attached to some items (e.g. test results). Inverting profits/item pickup may allow for modelling of drone-based fire suppression tasks, where visiting all locations allows for updating situation information and a limited water payload can be deployed in the most effective locations.

For ecological survey planning and remote vehicle exploration, e.g. mars/moon rovers, there are similar motivations. Many locations may require surveying but samples may not be collectable at all of them. Dynamic location change may be due to evolving hazards or reassessment of survey sites. Availability changes present as species/sample assignment information to locations is updated. Item profits may represent the importance associated with particular types of samples with varying priority. Many further opportunities exist, such as for the routing of service engineers that may need to collect appliances beyond repair, and in space debris removal tasks.

3.2. Dynamic city locations *(**Loc**)*

The DTSP literature describes dynamics in the changing locations of cities, the adding or removing of cities or the altering of specific distances between cities to simulate traffic. Further works will address these within the DTTP, however we focus presently on location changes alongside with dynamic changes in different subcomponents of the problem.

Where a city (x_i) is represented with Cartesian coordinates ($x_{\mathrm{x},i}, x_{\mathrm{y},i}$), and a distance matrix, D is constructed, the row and column elements of D must be updated when a city's location changes. The allowable translation limits for a city's location, the initial range in x and y directions is symmetrically increased by 5% in each direction whilst maintaining non-negative coordinate values (denoted as ϵ_{x}, ϵ_{y}). Given the model in Eq. (2), we redefine Eq. (3) to account for the changing city locations.

$$t_{x_i,x_{i+1}} = \frac{\sqrt{(x^{i+1}_{\mathrm{x}(\tau)} - x^{i}_{\mathrm{x}(\tau)})^2 + (x^{i+1}_{\mathrm{y}(\tau)} - x^{i}_{\mathrm{y}(\tau)})^2}}{v_c}$$

The new city locations are calculated via:

$$x_{\mathrm{x},i,\tau+1} = r_{\mathrm{x}} \quad , \quad x_{\mathrm{y},i,\tau+1} = r_{\mathrm{y}}$$

where τ is the dynamic interval counter and $r_{\mathrm{x}}, r_{\mathrm{y}}$ are drawn at random from the respective uniform discrete distributions:

$$r_{\mathrm{x}} \sim unif_{\mathrm{x}}\{\min(\min_{i=1,\dots,N}(x_{\mathrm{x},i,0}) - \epsilon_{\mathrm{x}}, 0), \max_{i=1,\dots,N}(x_{\mathrm{x},i,0}) + \epsilon_{\mathrm{x}}\}$$

$$r_{\mathrm{y}} \sim unif_{\mathrm{y}}\{\min(\min_{i=1,\dots,N}(x_{\mathrm{y},i,0}) - \epsilon_{\mathrm{y}}, 0), \max_{i=1,\dots,N}(x_{\mathrm{y},i,0}) + \epsilon_{\mathrm{y}}\}$$

A number of cities $d_{N,Loc}$ (magnitude of the change) are updated to feasible randomly generated locations and the distance matrix is updated.

Given a courier that must collect items along its route, city location changes can be interpreted as alternative depots (with the same items) being chosen. Some example motivations for this could be a closure or road incident preventing access to the original location.

3.3. Dynamic item availability *(**Ava**)*

The availability map defines the allocation of items across the cities. The work of Sachdeva et al. [35] considers the toggling of item availability, however a changing number of items between dynamic intervals introduces difficulties in comparing problem instances. Alternative mechanisms of availability changes can be formulated for specific application scenarios. As Eq. (2) shows, the packing plan z is constructed from the KP solution component according to the availability map; dynamic changes can be represented in the model as the time-dependent packing plan $z(\tau)$:

$$G(x,z) = \begin{cases} min \quad f(x,z(\tau)) = \sum_{i=1}^{n-1}(t_{x_i,x_{i+1}}) + t_{x_n,x_1} \\ \qquad x = (x_1,\dots,x_n) \\ max \quad g(x,z(\tau)) = \sum_{j\in z(\tau)} p_j Dr^{\left\lceil \frac{T_j}{C} \right\rceil} \end{cases}$$

Here a dynamic change in the availability map corresponds to a change in item-city assignments (I_{city}) for a number of items:

$$I_{city,\tau+1} = r$$

where r is a random city index drawn from the uniform discrete distribution $unif\{1,N\}$. The magnitude is controlled as a percentage of the items (as $d_{N,\mathbf{Ava}}$) which undergo an assignment change (since the number of items varies across problems).

Availability map changes can be contextually interpreted as stock shortages at the item's original city index and therefore sourcing from an alternative pickup location, a requested change in pickup location or the taking of a similar sample from a different location.

3.4. Dynamic item value (*Val*)

Dynamic Knapsack problems can have non-static capacities, numbers of items, weights or profits. In the context of many realistic problems, the non-static number of items and item profit make logical sense. Since varying the number of items requires updating the availability map, we consider novel dynamics in the item profits only. As before, the magnitude of each change is determined by the percentage of item profits that change $d_{N,\mathbf{Val}}$. Additionally, the *change factor, cf* gives the percentage and sign (chosen uniformly at random for each item) of change in each item's profit. The existing model can be reformulated as:

$$G(x,z) = \begin{cases} min \quad f(x,z) = \sum_{i=1}^{n-1}(t_{x_i,x_{i+1}}) + t_{x_n,x_1} \\ \qquad x = (x_1,\dots,x_n) \\ max \quad g(x,z) = \sum_{j\in z} p_j(\tau) Dr^{\left\lceil \frac{T_j}{C} \right\rceil} \end{cases}$$

with all parameters as previously. A subset of item's profit values p are subject to change between dynamic intervals, represented here by the time-dependency of the profit value of item j: $p_j(\tau)$. An item's profit value is updated as:

$$I_{p,\tau+1} = (1+cf) \times I_{p,\tau}$$

Item profit values may change for a variety of reasons that may include an update in the priority, utility or desirability of particular items.

4. Finding solutions to the DTTP

Combining optimal tours and KP-solutions does not give optimal TTP solutions [20] and longer tours may be necessary for higher profit TTP solutions [42]. However, the solved (or otherwise constructed) components provide better starting solutions than random initialization and can be calculated with relative ease.

Based on preliminary observations, the composition of the initial set provides differing non-dominated sets depending on the information used to construct the tours and packing plans (see Section 5.1 for details). We posit the use of population seeding methods for exploitation of problem information beyond the first interval of dynamic instances. Hence, we investigate the effectiveness of combinations of different initialization procedures to manage the impacts of dynamic changes.

Table 1
Responsive population construction methods for tour and packing plan solution components employed for the DTTP instances. *The *mN* algorithm uses the *mC* for initialization only.

Response strategy	TSP solution	Packing Plan
pS	solver	solver
pG	greedy	greedy
pR	random	random
mS	solver	solver/greedy/random
mG	greedy	solver/greedy/random
mR	random	solver/greedy/random
mC	solver/greedy/random	solver/greedy/random
mN	none*	none*

Table 2
KP component subtypes in problem set, see [52–54] for details.

Label	Items per city	Knapsack capacity	Weight/Profit Rel.
A	1	(low) $\frac{1}{11}\sum I_w$	strongly correlated
B	5	(med) $\frac{5}{11}\sum I_w$	similar
C	10	(high) $\frac{10}{11}\sum I_w$	uncorrelated

Algorithm 1 Responsive Seeding Algorithm

```
Require: $popsize, maxiter, SeedingMethod, d_N,$
        $DynType$
  $P \leftarrow SeedingMethod$   \\ Initialize population
  $EvaluateFitness(P)$
  $H.0 \leftarrow RecordHypervolume(P)$
  $iter \leftarrow 1$
  while $iter \le maxiter$ do
    if $iter\%(maxiter/d_N) == 0$ then
      Apply Dynamic Change of type $DynType$
      Update Problem Information
          \\Recalculate solver/greedy solution
          \\components if required
      $Q \leftarrow SeedingMethod$
    else
      $Q \leftarrow GeneticOperators(P)$
          \\Tournament selection, three part genetic
          \\operator application (see Section 4.1.4)
    end if
    $EvaluateFitness(Q)$
    $P \leftarrow P \cup Q$
    $RankAndNonDominatedSort(P)$
    Crop $P$ to $popsize$
    $H.iter \leftarrow RecordHypervolume(P)$
    $iter++$
  end while
  return $P, H$
```

4.1. Responsive seeding algorithm

Algorithm 1 shows the basic non-dominated sorting algorithm used (based on NSGA-II [55]). The dynamic response is denoted as *SeedingMethod* whereby solutions are constructed using one or a combination of methods, instead of generating offspring solutions by evolutionary operators. This *responsive seeding step* occurs only in the generation after a dynamic change. In total, eight different responsive seeding methods are investigated, shown in Table 1. Combinations of random, solver- and greedy-based construction methods, together with a combined approach and a passive (non-seeding algorithm) are used. Each of these methods are explained below.

4.1.1. Solver-based initialization

Several exact TSP solvers exist, for example Concorde [56] and Branch and Bound methods [57]. To remain consistent with the methods in [25], we use the Lin-Kernighan heuristic (LKH v2.0.9, [58]) for the optimal tour component. A simple dynamic programming approach solves the KP prior to the optimization. The optimal KP solution is transformed into a packing plan using the availability map. Dynamically recalculating an optimal KP solution for the largest problems is computationally infeasible and greedy or random KP solutions are used instead. A population of unique solutions is constructed using the optimal solution components by employing the mutation operators; Bitflip for KP solutions and Single Swap Mutation for the TSP tour.

4.1.2. Greedy initialization

Greedy methods can provide near-optimal solutions in some cases and little better than randomized solutions in others. Tours are constructed by iteratively from the first city (which has no items), selecting the minimum distance to any other city until a complete tour is formed.

For the KP solution, the items are sorted in descending order of their profit/weight ratio and the first items with a cumulative weight below the knapsack capacity W comprise the greedy solution. Similarly to the solver-based tours, the single solution is mutated using the algorithm operators to form a population.

4.1.3. Random initialization

The randomized population consists of random tour permutations and a random item permutations, truncated at the point where their cumulative weight exceeds the maximum capacity of the knapsack.

4.1.4. Algorithm parameters

The application of evolutionary operators follows [25]; the offspring population is a generated by combining equal proportions of solutions with operators applied to the tour, the packing plan and both components. An edge recombination crossover [59,60] and a single swap mutation [61] are used for tours and bitflip mutation and single point crossover [62] for the KP solutions before conversion to packing plans. After each dynamic change the initial tours constructed using the solver and greedy methods are updated; an Inver-Over Repair operator (as in [63]) is used as fast alternative to resolving.

Population size is fixed at 90. A single mutation and crossover event are guaranteed for offspring. Tournaments size for selecting crossover parents is $\frac{1}{10}popsize$. The maximum iterations are 1000, with five dynamic intervals.

4.1.5. Problem parameters and dynamics parameters

The examined problem set follows the format of competitions for the TTP [52–54]. A subset of the comprehensive TSPLIB-based benchmark set proposed in [39] is used[1] with three KP-component subtypes. These subtypes are given in Table 2; we denote these as *A, B & C* for clarity. From [39], evolutionary algorithms solving TTP instances can effectively cope with problems of up to 3000–5000 cities, however Wagner [42] suggests focusing on performance improvements on smaller instances first. Therefore our sample of problems fits this range ($N = berlin52, a280, rat783, u2319$). Problem variants are referred to by the number of cities and KP type (e.g. *52A*).

The schedule of dynamic changes is fixed at $d_{freq} = 200$ generations to allow population stability to return and for clearer observation of change impacts. A number of reproducible patterns of changes are pre-generated to guarantee the changes and allow for stochastic algorithm procedures. The magnitude of change is fixed at $d_{N,Loc} = 2, d_{N,Ava/Val} = 5\%$ to determine the preliminary impacts of the dynamics. Future work will address the impacts of frequency and severity in line with recent works [64,65].

[1] See: cs.adelaide.edu.au/_optlog/CEC2014COMP_InstancesNew/

4.2. Performance measurement & statistical testing

4.2.1. Measurements

Other studies on the bi-objective TTP measure hypervolume [25, 48], but no consistent nadir point is used. We use a reference point calculated as $(f^{\dagger}_{tour}, f^{\dagger}_{profit}) = [\overline{D} \times |D|, 0]$ to allow disparate coverage of the achieved sets to be compared, whilst ignoring the objective space dominated by the majority of competitive solutions. Any solution set that does not dominate this point is given a hypervolume of zero. The Zitzler Maximum Spread metric [66] and the mean crowding distance (MC) are also recorded (see Supplementary Material).

4.2.2. Profiles and ranking

The following procedure is applied separately to each problem. For 10 patterns of dynamic changes, the mean hypervolume across 30 repeats is calculated. For plotting the hypervolume profile, the median of these 10 instance-means is taken. Rankings are calculated comparing like-instances for all response methods on the 10 patterns. The median rank for each response is reported as the *composite median* in Fig. 3.

4.2.3. Statistical testing for quantitative comparisons of performance

A single value for comparative performance of algorithms that observes statistically relevant differences is calculated. For each algorithm a number of repeats are performed for each dynamic pattern, with the hypervolume recorded in each generation. At each numbered time step (generation), the measurements from repeats forms the sample set. These are used in pairwise one-tailed Wilcoxon ranksum comparisons, using Bonferroni correction ($n = 2$) and $\alpha = 0.05$. For example, using three methods *J, K & L*, a total of six tests are performed at each generation with the following alternative hypotheses:

- (1) $\mu_{t,J} > \mu_{t,K}$; (2) $\mu_{t,K} > \mu_{t,J}$;
- (3) $\mu_{t,J} > \mu_{t,L}$; (4) $\mu_{t,L} > \mu_{t,J}$;
- (5) $\mu_{t,K} > \mu_{t,L}$; (6) $\mu_{t,L} > \mu_{t,K}$;

Where $\mu_{t,J}, \mu_{t,K} \& \mu_{t,L}$ are the medians of the sample for algorithm *J, K & L* at generation t. The binary outcome of the tests are recorded.

Tests are repeated for every generation and the percentage of positive results (alternative hypothesis is accepted) is recorded. In each pair of comparisons (J vs. K, K vs. J) the maximum sum of these values will be 100% but is likely to be less than this as there are generations where performance between two algorithms is not significantly different. If algorithm J is better in more generations than the comparison algorithm K, a larger value will be seen for the J vs. K comparison.

5. Computational studies

The following results detail the comparative performance of different seeding strategies on the test problems.

5.1. Solution localization by initial population construction

The results in Fig. 1 illustrate the localization of non-dominated sets of solutions achieved by different population initializations.

Constructing the initial set based on information about the problem components has been used previously for the TTP [25–27], despite optimal solutions to the problem components not necessary yielding optimal TTP solutions [20]. Greedily constructed solutions ($\boldsymbol{g}$), solver-based solutions ($\boldsymbol{s}$) and randomly generated solutions ($\boldsymbol{r}$) are examined in various combinations.

Each of the series in Fig. 1 represents the aggregated set of 30 repeats. The localization of the non-dominated sets appears driven primarily by the construction method used for the tour component of the initial set. Solver-based tour initialization achieves a high density coverage in the minimum-tour region of the objective space, whilst greedily constructed solutions with longer tours achieve higher profits. Representation from the sets with randomly initialized tours is limited

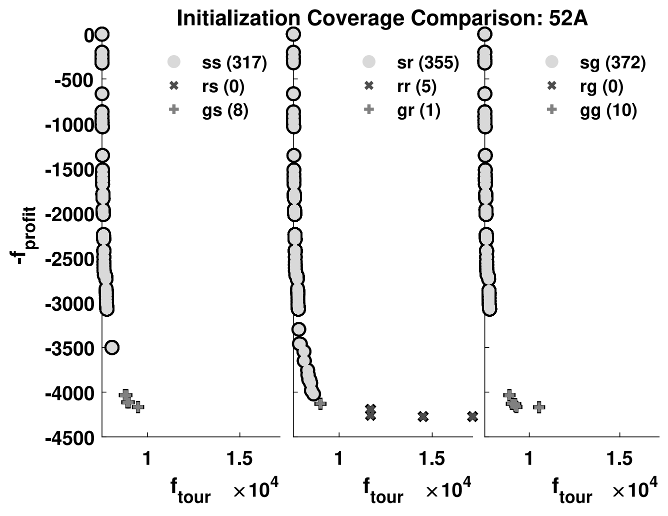

Fig. 1. Comparison of non-dominated set coverage with different initial population compositions. Each series corresponds to an initial set construction method (solver, random or greedy) for tours (first character) and KP-solutions (second character). Results featured for 30 independent runs of each initialization on the *52A* problem. Localization can be seen most clearly depending on tour initialization method; from left to right solver, greedy and random initial tours enable localized sets of non-dominated solutions.

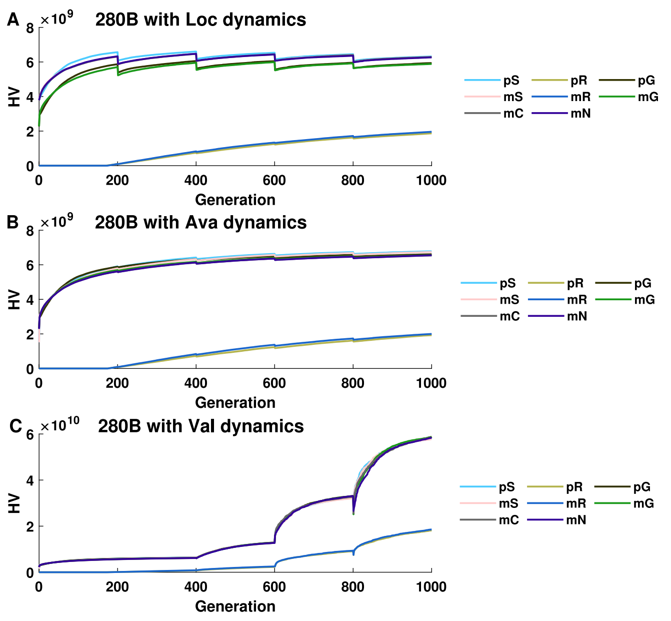

Fig. 2. Median hypervolume profiles across 10 patterns of dynamic changes (with 30 repeats for each) for eight different dynamic responses on the *280B* problem. Change frequency is 200 generations. (A) Response mechanism hypervolumes for the **Loc** dynamics. (B) Response mechanism hypervolumes for the **Ava** dynamics. The profiles of **Loc** and **Ava** hypervolumes are characteristic DMOP measurement curve and an attenuated-impact version, respectively. (C) Response mechanism hypervolumes for the **Val** dynamics. Each type of dynamic changes generates a visibly different effect on the problem but signed item profit changes (**Val**) show marked impact on existing solutions with successively increasing magnitudes.

to a few solutions with high profits and much longer tours. It should be noted this coverage is absent in most examined problems with larger KP or TSP components, whilst similar localization of greedy and solver-based solutions is generally present.

Based on these observations, we construct eight different initialization methods to be deployed in response to dynamics changes. These differently exploit problem information to try to maximize post-change algorithm performance. By replacing the offspring generation step with a set-construction step after a dynamic change, the population is seeded with solutions that will be relevant to the new dynamic interval. The goal is to mitigate the impacts of change and improve coverage of the non-dominated set in each interval. As before, the different seeding mechanisms are described in Table 1.

5.2. Impacts of dynamics in the DTTP

Hypervolume (HV) profiles can illustrate the high level features of the impacts of problem dynamics; each novel dynamic formulation affects the problem with different character and severity. Fig. 2 illustrates the HV profiles achieved for each of the eight seeding methods for the *280B* problem. The profiles plotted are the median of the ten dynamic instances, for each of which the mean of 30 repeats is calculated. Although not included here, these characteristic features are also identifiable in profiles of the other examined problems.

The different types of dynamics affect the ability of seeding responses to mitigate the impacts of dynamics. As these impacts are reliant on not just the type of dynamics, but their frequency and magnitude, the insights drawn here are preliminary and indicative of trends rather than concrete assessments. Characteristic-in-shape decreases in hypervolume after each of the five changes, can be seen in the **Loc** dynamics, whilst an attenuated version of these is seen in the HV profiles for the **Ava** dynamics. This is intuitive from the 'relative directness' of the tour and packing plan solution components; a change in the city locations impacts every solution in the population (since a valid solution must visit all cities), whilst a change in item availability immediately affects only those solutions containing the items that have been altered.

For the **Val** dynamics, the impact appears more varied and the performance of the responses more volatile. Similarly to the **Ava** dynamics, as every item may not included in a current solution (in the population) the impacts of changes is lessened. However, it appears that even a small change to the values of a small percentage of randomly selected items can evince a large change in the solution set's hypervolume.

5.3. Performance comparison of responsive population seeding methodologies for the DTTP

The polar plots in Fig. 3 illustrate the varying performance of the responsive seeding method across the set of problems with different types of dynamics.

A prominent feature of these results is the poor performance of the random seeding *pR*. Together with *mR*, these consistently achieve the lowest ranks on every problem and for each type of dynamics. The consistency between *pR* and *mR* indicates that diversifying the packing plan information (*mR* uses random, solver-based and greedily constructed knapsack solutions together with random tours) does not substantially improve performance. This reiterates the relative control the solution components excise on the optimization; good performance is primarily driven by good TSP solution components.

These results also indicate that the TSP-component of the examined problem can influence the most effective responsive seeding method. This is demonstrated in *pS* and *pG* methods for the *52 A,B&C* and *280 A,B&C* problems. Responsive seeding derived purely from solver solutions (*pS*) achieves better performance on the *280 A&B* problems, whilst greedy-constructed problems are better for the *52 A,B&C* problems. These statements apply across all three types of dynamics. The best response method for the *280C* problem appears to change with the type of dynamics; *pS* in **Loc** dynamics, and *pG* in **Ava** & **Val** dynamics.

Compared with each of the *pS* and *pG*, the diversification of the packing plan solution components has mixed results. The *mS* strategy, achieves lower ranks than *pS* for most problems; implying that solved sub-components are preferable to more diverse packing plan solution

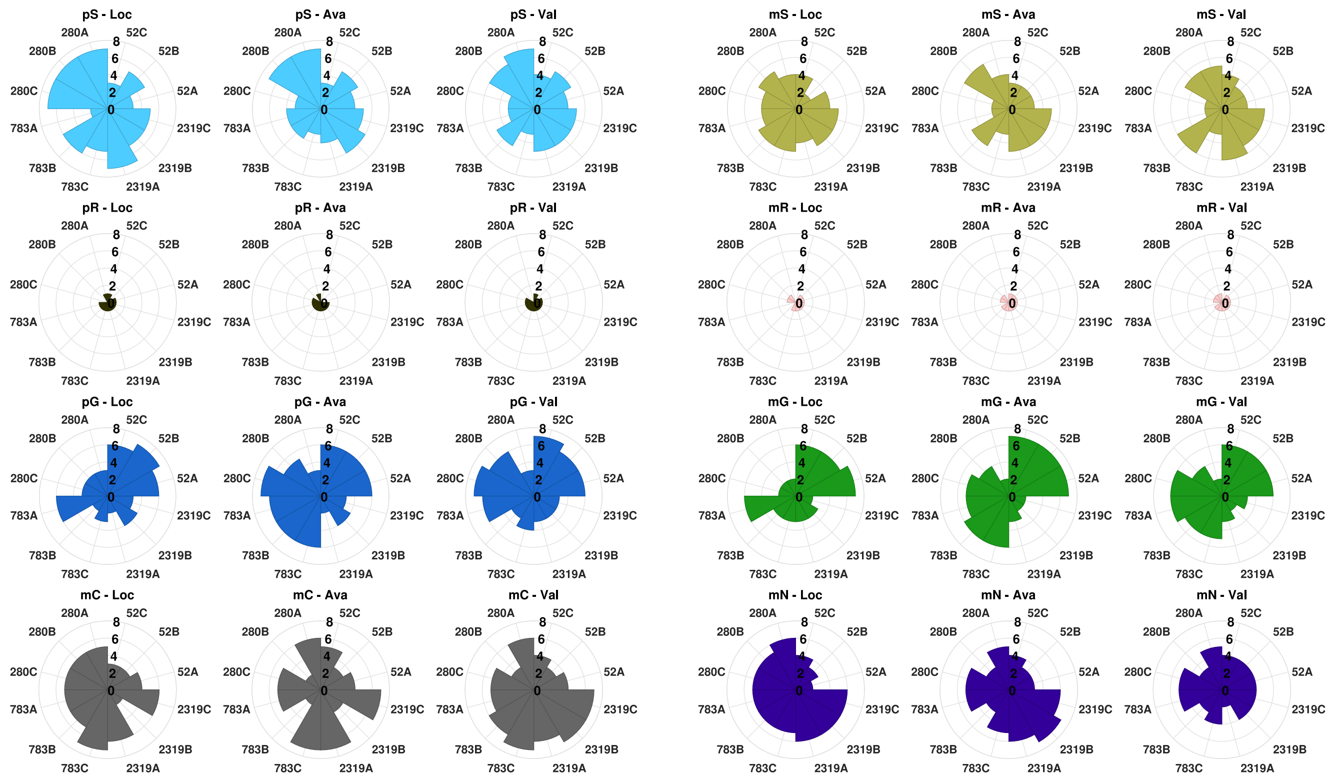

Fig. 3. Composite-median end of interval hypervolume rankings for the three types of dynamics, grouped into the eight different seeding methods (as listed in Table 1) on the 12 problems in the test set (plotted radially on each axes as the number of cities and KP type). Ranks are relative to other seeding methods. See Section 4.2.2 for the calculation of the composite-median. The **Loc**, **Ava** & **Val** labels indicate the type of dynamics in the problems. Solver-based responses (first row) show suitability for all 280 variants and some *2319* variants; randomized initialization (second row) consistently achieves the worst ranks on all problems; greedily-constructed seeding (third row) gives the best response on *52* variants and others depending on the type of dynamics. Combined (fourth row, left) and passive (fourth row, right) methods have complex trends depending on the type of dynamics and problem components.

components. For *mG*, for *52 A,B&C* with **Ava** dynamics, the maximum rank is achieved over *pG*. For other problems, *mG* ranks are below those achieved by *pG*; only for some problems a diversified packing plan set is a beneficial.

The *mN* method provides important insights into the comparative difficulty of each presented dynamic formulation. In the *mN* method the population is not seeded in response to a change and allows for comparisons with a 'do nothing' approach. Initialization with a diverse population of both tours and packing plans, as in the *mC* method, is used.

Clearly visible in the **Loc** dynamics (leftmost of bottom-right triplet in Fig. 3), is a steady increase in ranks as the size of the TSP-component grows. For problems with the same TSP-size, there is a similar increase between *A,B&C* (1, 5 & 10 items per city respectively). The other response methods increasingly struggle to do better than a passive approach as the size of both problem components increases. It is important to note that for **Loc** dynamics, the magnitude of the change is constant regardless of the TSP-component size. For the **Ava** dynamics, a similar trend is present with *mN* achieving high ranks on the largest problems. The ranks achieved for the **Val** dynamics imply they have a non-uniform effect on the range of problems without a clearly discernible trend.

Finally, the rankings achieved by the *mC* response method (bottom-left triplet in Fig. 3) inform on the utility of diversity in response to dynamic changes of different types. On the smaller and mid-sized problems (all *52 & 280* variants and all types of dynamics), *mC* achieves reasonable rankings, however the performance of *mC* is deflated by the good performance of the *pG, mG & pS* methods on these problems. Consistently high rankings can be seen for the larger problems in the set, (*783 & 2319* variants). Interestingly, the *2319B* problem is ineffectively handled by *mC* under both **Loc** and **Ava** dynamics. Lower rankings are also achieved an all B-type problems with **Ava** dynamics as well. This type of KP-problem has 5 items per city with similar weights and profits. Further investigation may elucidate the interactions between types of dynamics and KP-components. As the *mC* method provides reasonable performance across the problem set, we compare it with adapted methods for the static TTP.

6. Performance comparison with adapted TTP methods

We compare the mC strategy, named here as 'SeedEA', on 12 DTTP instances each with 10 patterns of city location changes, with two foundational methods for the static TTP: S5 [26] and MATLS [38]. Designed for non-dynamic single objective TTP problems, both employ TSP solvers and therefore the minimum time TTP solution is trivial compared with the high-profit solutions. S5 works on iterative improvement over a single solution via a parameter search; we allow the same number of iterations as generations for SeedEA. MATLS maintains a population of solutions but ha no dominance assessment compatible for optimizing for the bi-objective case. Therefore, both methods preferentially replace for high-profit solutions. As MATLS is population based (with *popsize* equal to SeedEA), we examine two versions; in **r**MATLS, the population is reinitialized at the beginning of each dynamic interval, whereas in **k**MATLS the population is kept in the next dynamic interval. Space limitations afford that comparison with other recently proposed methods, including WSM [51] and NDS-BRKGA [50], is reserved for further work.

6.1. S5 heuristic

The S5 heuristic was constructed from its description and the pseudocode provided in [26]. A score is calculated for each item based on

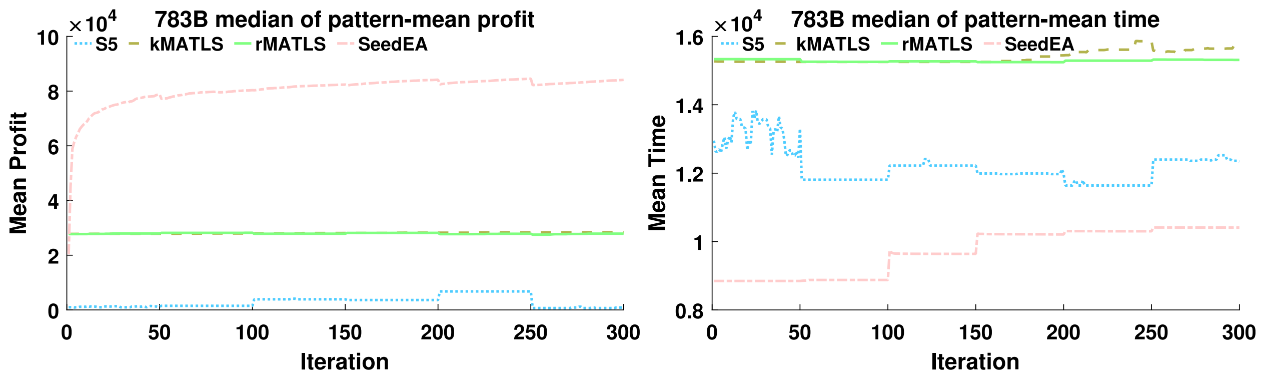

Fig. 4. (left) Median profile pattern-mean highest profit objective value in each iteration achieved by each method on the 783B problem. (right) as left but for lowest time objective value.

the benefit and impact on the remainder of the tour. Items are then selected iteratively and the scores updated until no further improvement can be found. The scoring parameters are varied during the heuristic to narrow the search for the best packing plan. Initialization occurs using the Chained Lin-Kernighan TSP solver for the tour component. The KP component is constructed iteratively by the heuristic.

6.2. MATLS (rMATLS and kMATLS)

The Memetic Algorithm with Two-Stage Local Search method was also constructed based on the pseudocode and description [38]. The basic format involves a local search step for the TSP and then the KP component of a small population of solutions. The tours are initialized using the Chained LKH TSP solver. The KP components are generated by an heuristic algorithm in the methodology proposed in the paper.

6.3. SeedEA

The SeedEA method is constructed using NSGA-II style [55] Pareto-dominance and crowding-distance based replacement tiebreaking. The *mC* response strategy from 1 is employed as previously. Both the S5 and the MATLS methods utilize a heuristic packing algorithm designed to select the best items after a tour is generated. To allow for competitive results, after a change event an approximate high profit solution is constructed by first building the terminal end of the tour based on the highest profit solutions consecutively chosen in reverse, accounting for the time-adjusted-profit and weight-adjusted-time of item selection. The shortest path for the remaining cities is found by LKH solver and combined with the terminal end of the solution. This was added to the post change offspring population.

Fig. 4 illustrates the attainment of the two objectives. Since the comparison methods were not designed for the bi-objective version of the problem, measurements such as hypervolume cannot fairly be compared. Moreover, since the use of exact TSP solvers is common to all methods, finding the minimum-time solution is trivial. Here however, we show the minimum tour solution alongside the maximum profit solution in each generation (left vs. right subplots) to highlight that SeedEA can effectively find a high-profit solution whilst maintaining a lower-time tour simultaneously.

SeedEA maintains pressure on achieving high profit solutions whilst preserving the diversity to achieve good objective space coverage. The additional diversity of this method allows for improved exploration of the high-profit solutions through exploitation of decision variable diversity contained in the population. The construction of the approximate high profit solution heuristic reveals the importance of the *dropConstant* in achieving high profit solutions. This value represents weight intervals at which the velocity of the thief is slowed by the weight of items; it may be possible to optimize to each threshold for a period of the tour — it may provide an exploitable feature for future heuristics. In terms of the SeedEA, by maintaining a greater diversity of solutions that comprise tours with items selected at a variety of locations, exploit this unhindered velocity threshold may be more common. Wagner [42] also states that high-profit solutions may require longer tours, of which the SeedEA maintains a selection.

Within Fig. 5 we compare performance of the methods across DTTP instances. We calculate the proportions of significant 'wins' per iteration of each method over the others across each pattern of dynamics and for each problem. A Bonferroni correction of $n = 3$ is applied. Due to persistent TSP solver divergence issues, the 2319 problems were substituted for variants of the rl1889 problem.

The radial bars represent the significant improvements over the other methods; the radial axis represents the proportion of iterations (across all repeats and patterns) in which a method achieves a significantly higher profit (top row) or significantly lower time (bottom row). The shading represents a significantly better solution over one (lightest), two (mid shade) or all three (darkest) of the other methods. For example, for kMATLS on the 1889A problem, in 100% of iterations, the profits are significantly better than one of the other methods (we can infer it is the S5 method). In approximately 70% of iterations, the profits are significantly better than two methods (S5 and rMATLS) and in 30% iterations kMATLS finds significantly higher profits over all three other methods.

Generally, the MATLS methods perform well on the A-type variant problems, but their performance declines greatly on B- and C-type problems. Results for kMATLS are better than rMATLS in achieving higher profits, whereas achieving both higher profit and low time solutions is better handled by rMATLS. S5 shows the opposite with no competitive performance on A-type problems but some limited significant achievement in profits and times in the B- and C-type variants.

The very good performance of kMATLS on the A-type variants reduces the overall significant performance of the SeedEA's darkest area for profit on these problems. S5's performance on 280C is unrivalled and it achieves good performance on 52B and 52C as well; these also contribute to reducing SeedEA's darkest shade area.

Given the comparison methods were not designed for the dynamic or multi-objective problem, they still provide significant improvements in high profit solutions on a limited subset of the problems examined. Another important consideration is the relative execution time and objective function evaluations that these methods consume.

The number of function evaluations, using a280 A,B & C as examples in Fig. 6, remains similar for SeedEA, and relatively lower than for rMATLS and kMATLS. For these, the value increases 10-fold from the A-type (1 item-per-city) to B-type (5 items-per-city), then 3-fold from B-type to C-type. For S5, the number of function evaluations remains lower across the different KP types.

The MATLS methods involve two local search stages, one for each of the TSP and KP problem components. With more items in the problem,

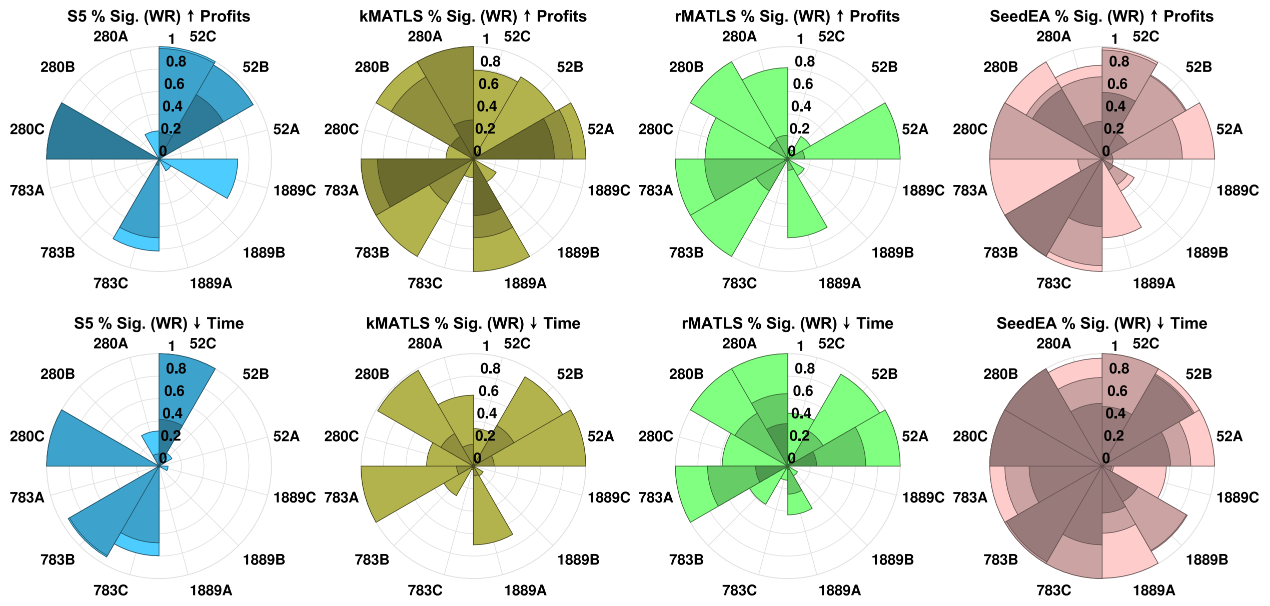

Fig. 5. Plots showing the problem set coverage of adapted TTP methods and SeedEA in terms of the proportion of iterations containing solutions with significantly higher profit (top row) and significantly lower time objective values (bottom row). Colour shades on each plot correspond to one, two and three significant improvements (from lightest to darkest) over the other methods. Statistical tests are carried out separately different dynamic patterns for each problem and aggregated in the percentage calculation. *WR: Wilcoxon ranksum.*

the neighbourhood of the local search increases and therefore the number of required evaluations to search it is inflated as the problem size increases. SeedEA's mechanism is independent of the problem size and therefore the number remains relatively constant across the problems.

In addition to function evaluations, each method requires differing numbers of TSP and KP solver calls. For example, the populations for the MATLS methods were initialized according to the original paper's protocol — for the TSP components 10 solutions initialized by LKH solver and 40 via Minimum Spanning Tree method. For rMATLS, this process is repeated after each dynamic change, greatly increasing the relative execution time.

Fig. 7 shows the execution times of each method with the line joining the median points for each problem. All experiments were run using MATLAB 2018b, an Intel-i7 processor (3.80 GHz) and 16 GB Memory.

On A-type problems, kMATLS has similar execution times to SeedEA and Fig. 5 depicts good significant improvements on these problems. However, as shown by Fig. 6 as size of the KP component increases (from A- to B- to C-type), SeedEA and the MATLS methods become increasingly different. For example, the largest of the problems 1889C, the median execution time for a single run of SeedEA was 2.7 h, for rMATLS this was 138 h. Comparison methods have redundant calls of TSP solvers, which can result in extreme inflation of execution times. Also greatly increasing the number of objective function evaluations, as in rMATLS and kMATLS also contributes to this difference.

6.4. Limitations and evaluation

Responding to dynamic changes through the introduction of diversity using randomly generated solutions are not always effective; here we have considered intuitive alternatives. Engineering the generation of solution sets that exploit known information about the problem (in this case using solvers for the problem components) has been shown to been useful in static environments and now here in some dynamic problem environments. We illustrate the difficulty in crafting an effective response to simplistic changes in such a complex problem with realistic characteristics. Nevertheless, we have demonstrated through the comparison of significant improvements in per-generation hypervolumes, that there is a benefit to using responses in different problem cases. The elucidation of the relationship between problem component characteristics and the efficacy profiles of the different responses is a non-trivial task whose understanding will allow for general insights for optimization algorithms for dynamic problems.

Specifically, further algorithm development to more intelligently compose reactive solution sets for the DTTP is required. Algorithms such as MOEA/D [67] have different dominance strategies, however preliminary results (see Supplementary Material) indicated that a non-dominated sorting framework may be better suited for TTP instances. Comparisons with other algorithms for the TTP, including the WMS [51] and BRKGA [50] methods for the bi-objective TTP are key targets for future comparisons and inspirations for implementing dynamic responses for the DTTP instances defined here. Larger DTTP instances, whilst potentially harder to realistically motivate, have been studied in the static TTP [38,50,51], will require longer computation times but should be investigated.

7. Conclusions

The outcomes of this study provide a step towards meaningful and realistic dynamic formulations of multi-objective optimization problems. Understanding how initialization can be used to guide and attain different solutions is particularly useful on these complex and difficult problems. Furthermore, being able to quantify with statistical evidence the benefits of different methods is key; there are multiple significant contributions from this work.

Firstly, we provide an observation of the different localization and coverage contributions achieved by a variety of initialization methods for the bi-objective TTP. We verify the general unsuitability of randomization initialization for the DTTP: constructed solutions that exploit problem knowledge provide indispensable improvements.

Secondly, this realization is applied to three novel and contextually-relevant DTTP formulations with time varying city locations, item availability and item values. Impacts of the dynamics are observed through different characteristic features in hypervolume measurement profiles.

Thirdly, application of different population seeding methods as responsive mechanisms for the DTTP are studied. End-of-interval hypervolume rankings show suitability of solver-based and greedy methods

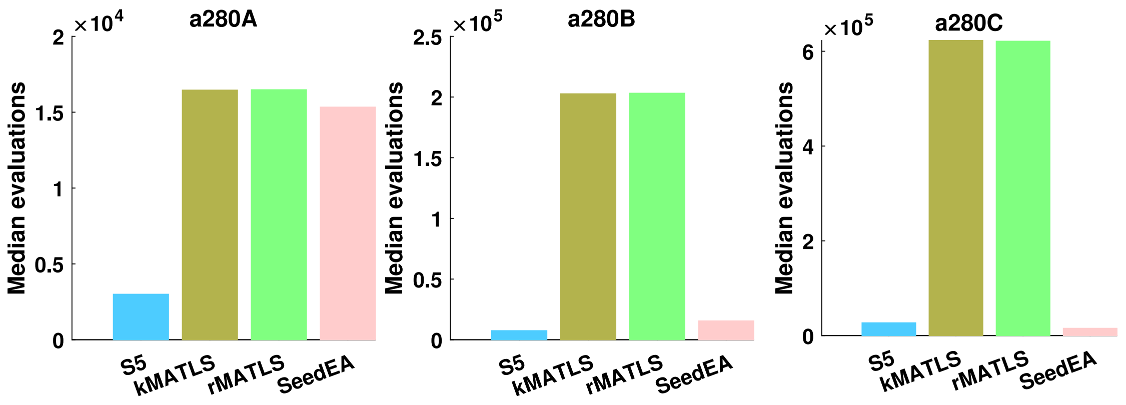

Fig. 6. The number of objective function evaluations; the median across the (up to) 30 repeats for each of 10 patterns of dynamics for the 280 TSP component with A, B and C KP variants for each of the four methods being compared.

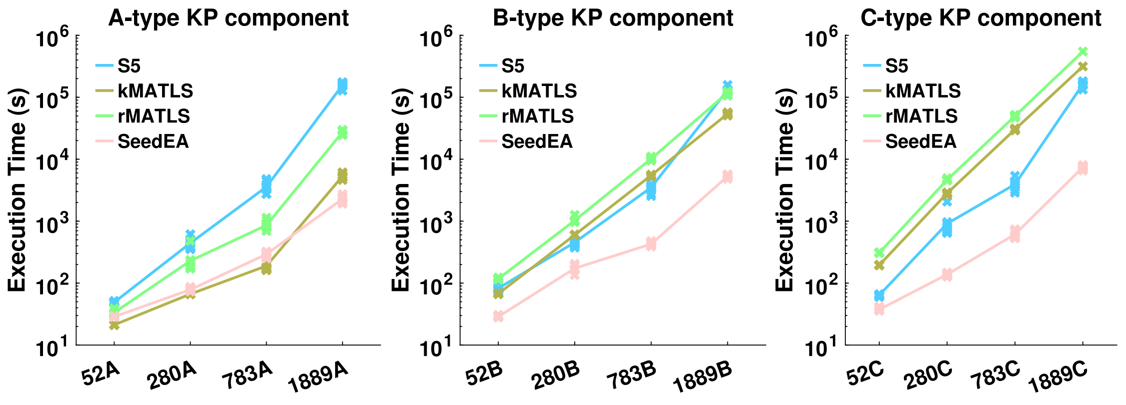

Fig. 7. Execution times for each comparison method. The mean for each dynamic pattern is given as a separate point and the median of these points for each problem is connected for clarity. The problems are grouped by their KP-component.

to mostly exclusive subsets of problems and requires further investigation. Comparison with random, passive and combined responses, highlights benefits of 'constructed diversity' for dynamic changes of all types.

Finally, the competitive performance of a diverse seeding method for the DTTP is contrasted with adapted heuristics from the literature. The capability of SeedEA for finding solutions with significantly higher profits and times is illustrated. More generally, informatively distilling temporal performance is no longer limited to averaging measurements over time; a novel and informative visualization of significant improvements is provided. Further application of this analysis to dynamic instances with varying magnitudes and frequencies remains a future task.

CRediT authorship contribution statement

Daniel Herring: Conceptualization, Methodology, Software, Investigation, Writing – original draft, Writing – review & editing, Visualization. **Michael Kirley:** Conceptualization, Resources, Writing – review & editing, Supervision, Funding acquisition. **Xin Yao:** Conceptualization, Writing – review & editing, Supervision, Funding acquisition.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data and code currently being used for additional research.

Acknowledgements

This research was partially funded by the Australian Government through the Australian Research Council Industrial Transformation Training Centre in Optimisation Technologies, Integrated Methodologies and Applications (OPTIMA), Project ID IC200100009.

Appendix A. Supplementary data

Supplementary material related to this article can be found online at https://doi.org/10.1016/j.swevo.2023.101433.

References

[1] J. Branke, Evolutionary optimization in dynamic environments, in: 2009 IEEE 5th International Conference on Intelligent Computer Communication and Processing, Springer US, 2002.
[2] S. Yang, X. Yao, Evolutionary Computation for Dynamic Optimization Problems, Vol. 490, 2013.
[3] R. Tinós, S. Yang, Analysis of fitness landscape modifications in evolutionary dynamic optimization, Inform. Sci. 282 (2014) 214–236.
[4] P. Rohlfshagen, X. Yao, Evolutionary dynamic optimization: Challenges and perspectives, in: S. Yang, X. Yao (Eds.), Evolutionary Computation for Dynamic Optimization Problems. Studies in Computational Intelligence, Vol 490, Springer, Berlin, Heidelberg, 2013.
[5] R. Azzouz, S. Bechikh, L. Ben Said, Dynamic multi-objective optimization using evolutionary algorithms: A survey, in: S. Bechikh, R. Datta, A. Gupta (Eds.), Recent Advances in Evolutionary Multi-Objective Optimization, Springer International, Cham, 2017, pp. 31–70.
[6] T. Nguyen, S. Yang, J. Branke, Evolutionary dynamic optimization: A survey of the state of the art, Swarm Evol. Comput. 6 (2012).

[7] J. Branke, H. Schmeck, Designing evolutionary algorithms for dynamic optimization problems, in: A. Ghosh, et al. (Eds.), Advances in Evolutionary Computing, 2003, pp. 239–262.

[8] S. Jiang, J. Zou, S. Yang, X. Yao, Evolutionary dynamic multi-objective optimisation : A survey, ACM Comput. Surv. 55 (4) (2021) 1–47

[9] M. Farina, K. Deb, P. Amato, Dynamic multiobjective optimization problems: Test cases approximation and applications, in: Evolutionary Multi-Criterion Optimization. Second International Conference EMO 2003, Vol. 8, No. 5, 2003, pp. 311–326.

[10] W.T. Koo, C.K. Goh, K.C. Tan, A predictive gradient strategy for multiobjective evolutionary algorithms in a fast changing environment, Memet. Comput. 2 (2) (2010) 87–110.

[11] M. Helbig, A.P. Engelbrecht, Benchmarks for dynamic multi-objective optimisation, in: Proceedings of the 2013 IEEE Symposium on Computational Intelligence in Dynamic and Uncertain Environments, Vol. 46, No. 3, SSCI 2013, 2013, pp. 84–91.

[12] S. Jiang, S. Yang, Evolutionary dynamic multiobjective optimization: Benchmarks and algorithm comparisons, IEEE Trans. Cybern. 47 (1) (2017) 198–211.

[13] D. Yazdani, M.N. Omidvar, R. Cheng, J. Branke, T.T. Nguyen, X. Yao, Benchmarking continuous dynamic optimization: Survey and generalized test suite, IEEE Trans. Cybern. 52 (5) (2022) 3380–3393.

[14] S. Jiang, M. Kaiser, S. Yang, S. Kollias, N. Krasnogor, A scalable test suite for continuous dynamic multiobjective optimization, IEEE Trans. Cybern. (2019) 1–13.

[15] G. Ruan, J. Zheng, J. Zou, Z. Ma, S. Yang, A random benchmark suite and a new reaction strategy in dynamic multiobjective optimization, Swarm Evol. Comput. 63 (December 2020) (2021) 100867.

[16] S.B. Gee, K.C. Tan, H.A. Abbass, A benchmark test suite for dynamic evolutionary multiobjective optimization, IEEE Trans. Cybern. 47 (2) (2017) 461–472.

[17] R. Azzouz, S. Bechikh, L.B. Said, A dynamic multi-objective evolutionary algorithm using a change severity-based adaptive population management strategy, Soft Comput. 21 (4) (2017) 885–906.

[18] M. Yang, L. Kang, J. Guan, Multi-algorithm co-evolution strategy for dynamic multi-objective TSP, in: IEEE Congress on Evolutionary Computation, CEC, 2008, pp. 466–471.

[19] C.M. Colson, M.H. Nehrir, S.A. Pourmousavi, Towards real-time microgrid power management using computational intelligence methods, in: IEEE PES General Meeting, No. January 2015, PES 2010, 2010.

[20] M.R. Bonyadi, Z. Michalewicz, L. Barone, The travelling thief problem: The first step in the transition from theoretical problems to realistic problems, in: 2013 IEEE Congress on Evolutionary Computation, CEC 2013, 2013, pp. 1037–1044.

[21] K. Deb, N. Udaya Bhaskara Rao, S. Karthik, Dynamic multi-objective optimization and decision-making using modified NSGA-II: A case study on hydro-thermal power scheduling, in: EMO'07 Proceedings of the 4th International Conference on Evolutionary Multi-Criterion Optimization, 2007, pp. 803–817.

[22] M. Mavrovouniotis, S. Yang, Ant algorithms with immigrants schemes for the dynamic vehicle routing problem, Inform. Sci. 294 (2015) 456–477.

[23] A. Zhou, Y. Jin, Q. Zhang, A population prediction strategy for evolutionary dynamic multiobjective optimization, IEEE Trans. Cybern. 44 (1) (2014) 40–53.

[24] I. Hatzakis, D. Wallace, Dynamic multi-objective optimization evolutionary algorithms: a forward-looking approach, in: Proceedings of ACM GECCO, Vol. 4, 2006, pp. 1201–1208.

[25] J. Blank, K. Deb, S. Mostaghim, Solving the bi-objective traveling thief problem with multi-objective evolutionary algorithms, in: H. Trautmann, G. Rudolph, K. Klamroth, O. Schütze, M.M. Wiecek, Y. Jin, C. Grimme (Eds.), Evolutionary Multi-Criterion Optimization EMO 2017, Proceedings, in: LNCS, vol. 10173, Springer, 2017, pp. 46–60.

[26] H. Faulkner, S. Polyakovskiy, T. Schultz, M. Wagner, Approximate approaches to the traveling thief problem, 2015, pp. 385–392.

[27] J. Wu, M. Wagner, S. Polyakovskiy, F. Neumann, Exact approaches for the travelling thief problem, Lecture Notes in Comput. Sci. 10593 LNCS (2017) 110–121.

[28] W. Li, M. Feng, A parallel procedure for dynamic multi-objective TSP, in: Proceedings of the 2012 10th IEEE International Symposium on Parallel and Distributed Processing with Applications, ISPA 2012, 2012, pp. 1–8.

[29] C. Raquel, X. Yao, Dynamic multi-objective optimization: A survey of the state-of-the-art, in: S. Yang, X. Yao (Eds.), Evolutionary Computation for Dynamic Optimization Problems, 2013, pp. 85–106.

[30] M. Helbig, A.P. Engelbrecht, Performance measures for dynamic multi-objective optimisation algorithms, Inform. Sci. 250 (2013) 61–81.

[31] M. Cámara, J. Ortega, F. De Toro, Performance measures for dynamic multi-objective optimization, Lecture Notes in Comput. Sci. 5517 LNCS (PART 1) (2009) 760–767.

[32] R. Morrison, Designing Evolutionary Algorithms for Dynamic Environments, in: Natural Computing Series, Springer Berlin Heidelberg, 2013.

[33] A. Muruganantham, K.C. Tan, P. Vadakkepat, Evolutionary dynamic multiobjective optimization via Kalman filter prediction, IEEE Trans. Cybern. 46 (12) (2016) 2862–2873.

[34] D. Herring, M. Kirley, X. Yao, Optimization of the dynamic multi-objective travelling thief problem, 2020, arXiv:2002.02636.

[35] R. Sachdeva, F. Neumann, M. Wagner, The Dynamic Travelling Thief Problem: Benchmarks and Performance of Evolutionary Algorithms, 2020, arXiv:2004.12045.

[36] M. El Yafrani, B. Ahiod, Efficiently solving the Traveling Thief Problem using hill climbing and simulated annealing, Inform. Sci. 432 (2018) 231–244.

[37] C. Wachter, Solving The Travelling Thief Problem with an Evolutionary Algorithm (Ph.D. thesis), Technische Universität Wien, 2015.

[38] Y. Mei, X. Li, X. Yao, Improving efficiency of heuristics for the large scale traveling thief problem, Lecture Notes in Comput. Sci. 8886 (2014) 631–643.

[39] S. Polyakovskiy, M.R. Bonyadi, M. Wagner, Z. Michalewicz, F. Neumann, A comprehensive benchmark set and heuristics for the traveling thief problem, in: Proceedings of the 2014 Annual Conference on Genetic and Evolutionary Computation, GECCO '14, 2014, pp. 477–484.

[40] Y. Mei, X. Li, X. Yao, On investigation of interdependence between sub-problems of the travelling thief problem, Soft Comput. 20 (1) (2016) 157–172.

[41] R. Birkedal, Design, Implementation and Comparison of Randomized Search Heuristics for the Travelling Thief Problem (Masters thesis), Technical University of Denmark, 2015.

[42] M. Wagner, Stealing items more efficiently with ants: A swarm intelligence approach to the travelling thief problem, Lecture Notes in Comput. Sci. 9882 LNCS (2016) 273–281.

[43] M.R. Bonyadi, Z. Michalewicz, M.R. Przybyłek, A. Wierzbicki, Socially inspired algorithms for the traveling thief problem, in: GECCO 2014 - Proceedings of the 2014 Genetic and Evolutionary Computation Conference, 2014, pp. 421–428.

[44] M. Wagner, M. Lindauer, S. Nallaperuma, F. Hutter, M. Mısır, A case study of algorithm selection for the traveling thief problem, J. Heuristics 24 (2018) 295–320.

[45] D. Applegate, W. Cook, A. Rohe, Chained Lin-Kernighan for large traveling salesman problems, INFORMS J. Comput. 15 (1) (2003) 82–92.

[46] S. Polyakovskiy, F. Neumann, Packing while traveling: Mixed integer programming for a class of nonlinear knapsack problems, Lecture Notes in Comput. Sci. 9075 (2015) 332–346.

[47] S. Polyakovskiy, F. Neumann, The packing while traveling problem, European J. Oper. Res. 258 (2) (2017) 424–439.

[48] J. Wu, S. Polyakovskiy, M. Wagner, F. Neumann, Evolutionary computation plus dynamic programming for the bi-objective travelling thief problem, in: Proceedings of the Genetic and Evolutionary Computation Conference, GECCO '18, 2018, pp. 777–784.

[49] M.E. Yafrani, S. Chand, A. Neumann, B. Ahiod, M. Wagner, Multi-objectiveness in the single-objective traveling thief problem, in: Proceedings of the Genetic and Evolutionary Computation Conference Companion, GECCO '17, 2017, pp. 107–108.

[50] J.B.C. Chagas, J. Blank, M. Wagner, M.J.F. Souza, K. Deb, A non-dominated sorting based customized random-key genetic algorithm for the bi-objective traveling thief problem, J. Heuristics 27 (3) (2021) 267–301.

[51] J.B. Chagas, M. Wagner, A weighted-sum method for solving the bi-objective traveling thief problem, Comput. Oper. Res. 138 (2022) 1055–1060.

[52] W. Gao, S. Polyakovskiy, M. Wagner, Optimisation of problems with multiple interdependent components, 2017, URL https://cs.adelaide.edu.au/~optlog/TTP2017Comp/.

[53] J. Blank, M. Wagner, GECCO2019 - Bi-objective traveling thief competition documentation, 2019, URL https://www.egr.msu.edu/coinlab/blankjul/gecco19-thief/#blank-2017-sbt-3088676-3088680.

[54] J. Blank, M. Wagner, EMO2019 - ranking — Thief 1.0.0 documentation, 2019, URL https://www.egr.msu.edu/coinlab/blankjul/emo19-thief/.

[55] K. Deb, A. Pratap, S. Agarwal, T. Meyarivan, A fast and elitist multiobjective genetic algorithm: NSGA-II, IEEE Trans. Evol. Comput. 6 (2) (2002) 182–197.

[56] D. Applegate, R. Bixby, V. Chvátal, W. Cook, On the solution of traveling salesman problems, Docum. Math. J. Deutschen Mathematiker-Vereinigung Int. Congress Mathe. (1998) 645–656.

[57] J.D.C. Little, K.G. Murty, D.W. Sweeney, C. Karel, An algorithm for the traveling salesman problem, Oper. Res. 11 (6) (1963) 972–989.

[58] K. Helsgaun, Effective implementation of the Lin-Kernighan traveling salesman heuristic, European J. Oper. Res. 126 (1) (2000) 106–130.

[59] I.M. Oliver, D.J. Smith, J.R.C. Holland, A study of permutation crossover operators on the traveling salesman problem, in: Proceedings of the Second International Conference on Genetic Algorithms on Genetic Algorithms and their Application, 1987, pp. 224–230.

[60] L.D. Whitley, T. Starkweather, D. Fuquay, Scheduling problems and traveling salesmen: The genetic edge recombination operator, in: Proceedings of the 3rd International Conference on Genetic Algorithms, 1989, pp. 133–140.

[61] W. Banzhaf, The molecular traveling salesman, Biol. Cybernet. 64 (1990) 7–14.
[62] X. Yu, M. Gen, Introduction to Evolutionary Algorithms, in: Ser. Descision Engineering, Springer International Publishing, 2010.
[63] A. Zhou, L. Kang, Z. Yan, Solving dynamic TSP with evolutionary approach in real time, in: 2003 Congress on Evolutionary Computation, CEC 2003 - Proceedings, Vol. 2, 2003, pp. 951–957.
[64] D. Herring, M. Kirley, X. Yao, Reproducibility and baseline reporting for dynamic multi-objective benchmark problems, in: Genetic and Evolutionary Computation Conference, GECCO '22, 2022.
[65] D. Herring, M. Kirley, X. Yao, An iterative machine learning approach to informative performance reporting in dynamic multi-objective optimization, in: Proceedings of the Companion Conference on Genetic and Evolutionary Computation, GECCO '23 Companion,New York, NY, USA, 2023, pp. 367–370.
[66] E. Zitzler, Evolutionary Algorithms for Multiobjective Optimization: Methods and Applications (Doctoral thesis), ETH Zurich, 1999, pp. 1–122,
[67] Q. Zhang, H. Li, MOEA/D: a multiobjective evolutionary algorithm based on decomposition, IEEE Trans. Evol. Comput. 11 (2008) 712–731.