Contents lists available at ScienceDirect

Computers & Operations Research

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

An evolutionary approach to generalized biobjective traveling salesperson problem

Murat Köksalan [a], Diclehan Tezcaner Öztürk [b,*]

[a] Department of Industrial Engineering, Middle East Technical University, Ankara, Turkey
[b] Department of Industrial Engineering, TED University, Ankara, Turkey

A R T I C L E   I N F O

Article history:
Received 30 September 2015
Received in revised form
22 March 2016
Accepted 26 April 2016

Keywords:
Evolutionary computations
Multiobjective decision making
Combinatorial optimization
Multiobjective traveling salesperson problem

A B S T R A C T

We consider the generalized biobjective traveling salesperson problem, where there are a number of nodes to be visited and each node pair is connected by a set of edges. The final route requires finding the order in which the nodes are visited (tours) and finding edges to follow between the consecutive nodes of the tour. We exploit the characteristics of the problem to develop an evolutionary algorithm for generating an approximation of nondominated points. For this, we approximate the efficient tours using approximate representations of the efficient edges between node pairs in the objective function space. We test the algorithm on several randomly-generated problem instances and our experiments show that the evolutionary algorithm approximates the nondominated set well.

1. Introduction

The generalized multiobjective traveling salesperson problem (MOTSP) finds efficient tours that visit all the nodes and return to the starting point under multiple objectives. In this problem, we need to determine both the order of visiting the nodes and the edge to use between consecutive node pairs. Finding the efficient edges between node pairs is a multiobjective shortest path problem (MOSPP). The overall problem can be considered as a generalized MOTSP with multiple efficient edges between nodes.

The generalized MOTSP can be considered in many domains; route planning of air vehicles, trucks, trains, and vessels. For the route planning problem of trains, the terminals can be considered as nodes and the alternative paths between terminals can be considered as edges between node pairs. For the route planning problem of air vehicles, the targets they aim to visit can be considered as nodes and the alternative paths between target pairs can be considered as edges. Minimizing total distance traveled, fuel consumption, detection avoidance, duration of travel are some of the objectives that can be used.

In the MOTSP literature, each pair of nodes has been assumed to be connected by a single edge. In a multiobjective context, however, there are typically many efficient edges between pairs of nodes; each edge representing a different tradeoff between objectives. A more general and realistic MOTSP is to consider the efficient tours that are composed of efficient edges. The differences of MOTSP with multiple efficient edges and classical MOTSP are discussed in Tezcaner and Köksalan [20].

The classical MOTSP and MOSPP are NP-hard [4]. There are a number of studies that develop heuristics for the classical MOTSP. Paquete and Stützle [17] develop a two-phase local search method, Jaszkiewicz and Zielniewicz [6] develop an algorithm that uses path relinking and Pareto local search, Lust and Teghem [13] propose a new two-phase Pareto local search for MOTSP and Lust and Jaszkiewicz [12] develop speed-up techniques for this heuristic for large MOTSPs. Ke et al. [7] propose a memetic algorithm for multiobjective combinatorial optimization problems and apply the algorithm to the MOTSP. Özpeynirci and Köksalan [15,16] study MOTSPs that have special structures. For a review on heuristics developed for MOTSP, we refer the reader to Lust and Teghem [13]. The generalized biobjective traveling salesperson problem (BOTSP) is studied in Tezcaner and Köksalan [20] and Tezcaner Öztürk and Köksalan [21]. They develop interactive algorithms that find the most preferred solution of a decision maker (DM) under linear preference functions [20] and quasiconvex preference functions [21]. They apply the algorithms on generalized BOTSPs.

In this study, we address the generalized BOTSP with a heuristic approach. We need to solve both the biobjective shortest path problem (BOSPP) and the BOTSP with multiple efficient edges between nodes. Since both BOTSP and BOSPP have been shown to be NP-hard, heuristic algorithms are required for addressing large instances. We develop an approach that generates an approximation for the nondominated set of the generalized BOTSP.

* Corresponding author.

http://dx.doi.org/10.1016/j.cor.2016.04.027
0305-0548/© 2016 Elsevier Ltd. All rights reserved.

We define the problem in Section 2 and develop the algorithm in Section 3. We give computational results in Section 4 and present our conclusions in Section 5.

2. Problem definition

We first present some notation and definitions that are adapted from Tezcaner and Köksalan [20].

Let x denote the decision variable vector, X denote the feasible set, Z denote the image of the feasible set in objective function space, and point $z(x)=(z_1(x), z_2(x),\ldots,z_p(x))$ be the objective function vector corresponding to the decision vector x, where p is the number of objectives and $z_k(x)$ is the performance of solution x in objective k. We assume, without loss of generality, that all objectives are to be minimized. A solution $x \in X$ is said to be *efficient* if there does not exist $x' \in X$ such that $z_k(x') \le z_k(x)$ $k = 1, \ldots, p$ and $z_k(x') < z_k(x)$ for at least one k. If there exists such an x', x is said to be *inefficient*. The set of all efficient solutions constitute the efficient set.

If x is efficient, then $z(x)$ is said to be *nondominated*, and if x is inefficient, $z(x)$ is said to be *dominated*. The set of nondominated points constitute the nondominated set. A nondominated point $z(x)$ is a *supported nondominated point* if there exists a positive linear combination of objectives that is minimized by x. Otherwise, $z(x)$ is an *unsupported nondominated point*. We define an *extreme nondominated point* as a supported nondominated point that has the minimum possible value in at least one of the objectives.

For the generalized BOTSP, we first find all efficient edges between node pairs utilizing algorithms developed for BOSPP. MOSPP has been studied well in the literature (see for example [5,18]). Evolutionary algorithms (EAs) have also been developed for MOSPP (see for example [14]). After finding all efficient edges connecting each node pair, we solve the formulation given below for the generalized BOTSP (as in [21]) to find efficient tours that use a subset of the efficient edges.

Let $G = (N, E)$ be an undirected graph with node set $N = \{1,2, \ldots\}$ and edge set E, and let R_{ij} be the index set of efficient edges between node pair (i, j). Let the binary decision variable x_{ijr} take value 1 if the rth efficient edge connecting nodes (i, j) is used, and 0 otherwise for $r \in R_{ij}$, c_{ijr}^k denote the kth objective value of the rth efficient edge between nodes i and j, and $P = \{(i, j) | i \in N, j \in N, i \ne j\}$ be the set of all node pairs. The formulation of the problem is as follows:

$$\text{Min} \quad z_1(x) = \sum_{(i,j)\in P} \sum_{r\in R_{ij}} c_{ijr}^1 x_{ijr} \tag{1}$$

$$\text{Min} \quad z_2(x) = \sum_{(i,j)\in P} \sum_{r\in R_{ij}} c_{ijr}^2 x_{ijr} \tag{2}$$

Subject to:

$$\sum_{j\in N} \sum_{r\in R_{ij}} x_{ijr}=1 \ i\in N \tag{3}$$

$$\sum_{i\in N} \sum_{r\in R_{ij}} x_{ijr}=1 \ j\in N \tag{4}$$

$$\sum_{i\in U} \sum_{j\in N/U} \sum_{r\in R_{ij}} x_{ijr}\ge 1 \ U \subset N, 2 \le |U| \le |N|-2 \tag{5}$$

$$x_{ijr}\in\{0, 1\} \ (i, j)\in P, \ r\in R_{ij} \tag{6}$$

Here we assume that $c_{ijr}^k \ge 0$ for $k = 1,2$ and $r \in R_{ij}$; i.e. the values of the edges constituting these two objectives are assumed to take nonnegative values. Eq. (3) ensures that one edge leaving each node is selected and Eq. (4) guarantees an edge entering each node is selected. Constraint (5) is a standard traveling salesperson problem subtour elimination constraint that assures at least one connection between two separate subsets of the nodes.

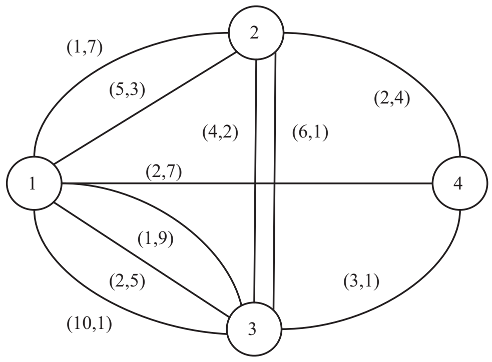

Fig. 1. Generalized BOTSP example.

An efficient solution corresponds to the efficient tour of the traveling salesperson problem (TSP) together with the specific efficient edges used between the consecutive nodes visited by that tour. Moving from node $i \in N$ to node $j \in N$, there may be many efficient edges. Let π be a cyclic permutation of set N and Π be the set of all tours. As we have many edge options between the nodes, all $\pi \in \Pi$ can be traversed by a number of combinations of the efficient edges between its connected nodes. In the rest of the paper, to differentiate among different edge options between two nodes, we denote the rth efficient edge between nodes i and j as e_{ijr} where $r\in R_{ij}$ and the set of edges as $E = \left\{e_{121}, e_{122},\ldots,e_{|N|,|N|-1,|R_{|N|,|N|-1}|}\right\}$. Similarly, we denote the uth efficient edge combination for tour π as $t_{\pi u}$ where $u\in U_\pi$, and its kth objective function value as $C_{\pi u}^k$, for $k = 1,2$. To demonstrate, consider the example in Fig. 1 with four nodes. The values written in parentheses next to each edge are the first and second objective function values, respectively, of the corresponding efficient edge. Between node pairs 1-4, 2-4, and 3-4, we have single connections (single efficient edges). Therefore, $|R_{14}|=|R_{24}|=|R_{34}|=1$. There are two efficient edges between node pairs 1-2 and 2-3 ($|R_{12}|=|R_{23}|=2$) and three efficient edges between node pair 1-3 ($|R_{13}|=3$). For example, the objective function values for the efficient edge between node pair 1-4 are $c_{14,1}^1=2$ and $c_{14,1}^2=7$, using our notation. We have three possible distinct tours to visit the four nodes, $\Pi = \{$1-2-3-4-1, 1-2-4-3-1, 1-3-2-4-1$\}$. For tour 1-2-3-4-1, there are four different routes considering the different edges between the nodes (all possible combinations of the two edges between node pair 1-2, two edges between node pair 2-3, one edge between node pair 3-4, and one edge between node pair 4-1). When all these combinations are enumerated and their objective function values are calculated, all four points turn out to be nondominated with the following objective function values: $C_{1-2-3-4-1,1}^1=10$, $C_{1-2-3-4-1,1}^2=17$, $C_{1-2-3-4-1,2}^1=12$, $C_{1-2-3-4-1,2}^2=16$, $C_{1-2-3-4-1,3}^1=14$, $C_{1-2-3-4-1,3}^2=13$, and $C_{1-2-3-4-1,4}^1=16$, $C_{1-2-3-4-1,4}^2=12$. Therefore, the nondominated set of tour 1-2-3-4-1 is composed of four points; (10,17), (12,16), (14,13) and (16,12).

When we enumerate the solutions of tour 1-2-4-3-1 and find their objective function values, we obtain six points, four of which are nondominated, disregarding the nondominated points of other tours. Similarly, for tour 1-3-2-4-1, we obtain six points, five of which are nondominated, again disregarding the nondominated points of other tours. When we evaluate all points of all tours together, some points turn out to be dominated. In our case, the overall nondominated set is made up of five points; (7,21), (8,17), (12,13), (20,9) and (16,12). First four of the nondominated points correspond to tour 1-2-4-3-1 and the last one corresponds to tour 1-2-3-4-1.

3. The evolutionary approach

We develop an evolutionary algorithm (EA) that approximates the nondominated set of generalized BOTSPs. EAs have various mechanisms, such as representation, fitness values, parent selection, crossover, mutation, etc. In our approach, we use a problem-specific representation where each individual corresponds to a tour. Since each tour has its own nondominated set, we do not assign a single fitness value to a tour. Instead, we develop an evaluation function that approximates the nondominated sets of the tours and represents each nondominated set by several points on the function. The genetic operators we use are also problem specific. The mechanisms we develop for solving the generalized BOTSP can be incorporated into any multiobjective EA.

We next develop these mechanisms.

3.1. Development of the algorithm

3.1.1. Representation

Each individual in our population corresponds to a tour (π) that determines the order of visit to the nodes. We need to have a representation scheme that characterizes each individual in the EA. There are different representation schemes for the TSP (see [11]). We use a scheme called the *path representation*. In this scheme, if node *i* is the *j*th node to be visited, it is placed in the *j*th position in the chromosome. For instance, the representation 1-5-4-2-3 shows that node 1 is visited first, followed by node 5 and so on.

3.1.2. Evaluation

To evaluate the individuals (tours) in the population, we need their objective function values. However, as explained in Section 2, each tour, π, is typically made up of many efficient solutions, $t_{\pi u}$, $u \in U_\pi$, depending on which efficient edges are used. Therefore, corresponding to each tour in the population, we may have many nondominated points corresponding to different efficient solutions. It is computationally difficult to find all $t_{\pi u}$ of each tour $\pi \in \Pi$. Furthermore, many of the tours in the population are likely to become inefficient as the populations progress, and it is a waste of effort to find all $t_{\pi_d u}$ of inefficient tours $\pi_d \in \Pi$. Instead of generating all $t_{\pi u}$ of a tour π, we approximate the nondominated set of that tour using a function that fits a curve passing through several nondominated points of that tour. The curve corresponds to a hypothetical frontier that approximates the spread of the tour in the objective function space. On each generated hypothetical frontier, we then select a number of points that represent the corresponding tour in the objective function space. The details of this approach are given below.

3.1.2.1. Fitting a function. We fit a function that passes through three nondominated points of tour π in the objective function space. We use the two extreme nondominated points ($C^k_{\pi L}$ denoting the *k*th objective value of the left extreme nondominated point with minimum value in objective 1 and $C^k_{\pi R}$ denoting the *k*th objective value of the right extreme nondominated point with the minimum value in objective 2, among the nondominated points of tour π) and a central nondominated point that balances the two objectives ($C^k_{\pi m}$ denoting the *k*th objective value of the central nondominated point) for $k = 1,2$.

We use an L_q distance function that was developed by Köksalan [8] for two objectives in a scheduling context and was later generalized by Köksalan and Lokman [9] to any number of objectives. The latter article demonstrates on many combinatorial problems that fitting an L_q distance function based on only a few nondominated points approximates the nondominated set well and that this function can approximate both convex and concave nondominated sets. The function was utilized within an EA context in a bicriteria hub location problem by Köksalan and Soylu [10]. The function is as follows:

$$(1-zf_1^m)^q+(1-zf_2^m)^q=1 \tag{7}$$

where

$zf^m=(zf_1^m, zf_2^m)=\left(\frac{C^1_{\pi m}-C^1_{\pi L}}{C^1_{\pi R}-C^1_{\pi L}}, \frac{C^2_{\pi m}-C^2_{\pi R}}{C^2_{\pi L}-C^2_{\pi R}}\right)$ are the normalized objective function values for a given (say central) solution. Firstly, we normalize the objective function values of the central solution using the two extreme nondominated points. We then fit an L_q distance function using Eq. (7). Since there is a single q value that satisfies (7), the computation of q requires a trial and error approach and we use binary search to find it. Alternatively, a nonlinear programming problem can also be solved with an auxiliary (or pseudo) objective function enforcing (7) as the only constraint. The accuracy of the fit of the L_q distance functions have been shown to be very good for a variety of multiobjective combinatorial optimization problems by Köksalan and Lokman [9]. In our computations, the high quality of the overall approximations of the real nondominated sets indicates that L_q distance functions represent the nondominated sets of individual tours well.

We explain the generation of the L_q distance function on an example tour in Fig. 2. For demonstration purposes we also

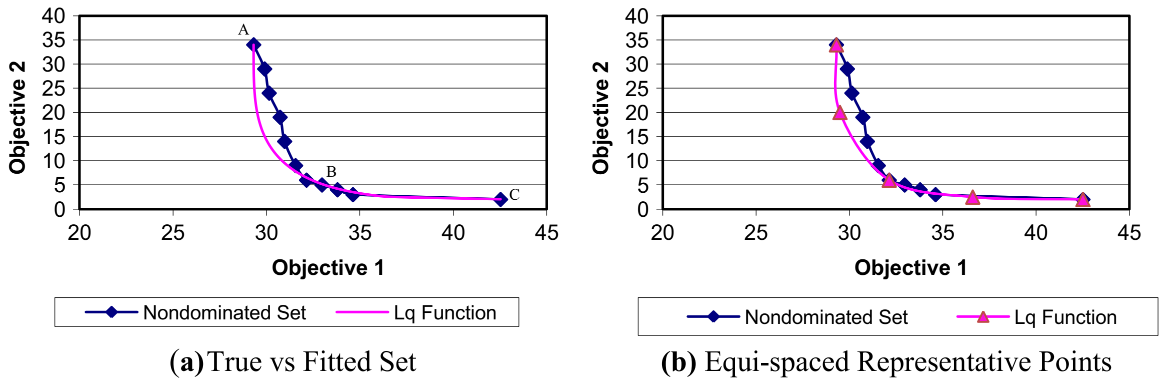

Fig. 2. L_q distance function approximation of a tour.

generated the true nondominated set of the tour by finding all efficient edges between the connected nodes of the tour and enumerating all possible combinations of these efficient edges. The combinations that turn out to be nondominated constitute the nondominated set for that tour, which is shown with the set of dark points and a curve combining them in Fig. 2.

In Fig. 2(a), we generate the L_q distance function using points A, B and C where points A and C correspond to the left and right extreme nondominated points, respectively, and B corresponds to the central nondominated point. The L_q distance function (the light colored curve) approximates the original nondominated set closely in this case. We then place equally spaced points on the L_q curve to represent the tour in the EA as shown in Fig. 2(b). We will further discuss these representative points when presenting the details of the algorithm.

3.1.2.2. Finding extreme nondominated points. Recall $z_k(x)$ denotes the kth objective function, for $k = 1,2$, where both are to be minimized. Let $U(z)=wz_1(x)+(1-w)z_2(x)$ be a composite objective function formed by combining the two objectives for some $0 \le w \le 1$. Minimizing $U(z)$ for $w = 1-\rho$, we obtain the solution that has the best value in the first objective function that corresponds to the left extreme nondominated point. Similarly, minimizing $U(z)$ for $w = \rho$, we find the solution that has the best value in the second objective function that corresponds to the right extreme nondominated point. Here, ρ is a sufficiently small positive constant to prevent obtaining inefficient solutions (see [19], pp. 422–430).

In order to fit an L_q distance function, we need to find the extreme nondominated points of each tour. For a given tour π, we need to determine which edges to use between the connected nodes of that tour. Therefore, the problem reduces to shortest path problems between the connected node pairs of the tour.

Let c_{ijL}^k and c_{ijR}^k denote the kth objective values of the left and right extreme nondominated points of the edges connecting nodes i and j, respectively, for $k = 1,2$. Then the kth objective function values of the corresponding extreme nondominated points of tour π can be found as:

$$C_{\pi L}^k = \sum_{(i,j)\in\pi} c_{ijL}^k \quad k=1,2 \tag{8}$$

$$C_{\pi R}^k = \sum_{(i,j)\in\pi} c_{ijR}^k \quad k=1,2 \tag{9}$$

Knowing the extreme nondominated points for the edges between the connected node pairs, we compute the values of the extreme nondominated points for the tours in (8) and (9).

3.1.2.3. Finding the central nondominated point. We find the central nondominated point utilizing the two extreme nondominated points of tour π. Given the values of $C_{\pi L}^k$ and $C_{\pi R}^k$ for $k = 1,2$, we obtain:

$$w' = \frac{C_{\pi L}^2 - C_{\pi R}^2}{C_{\pi L}^2 - C_{\pi R}^2 + C_{\pi R}^1 - C_{\pi L}^1} \tag{10}$$

The weight, w', defines an objective function contour, $U(z)$, that passes through the two extreme nondominated points. These two extreme nondominated points yield the same composite objective function value with these weights. By minimizing $U(z)$ using $w = w'$, we obtain a central solution. Let c_{ijm}^k denote the kth objective value of the central nondominated point of the edge connecting the nodes i and j, for $k = 1,2$. Similar to the extreme nondominated points, the values of the central nondominated point of tour π are found as follows:

$$C_{\pi m}^k = \sum_{(i,j)\in\pi} c_{ijm}^k \quad k=1,2 \tag{11}$$

To find the central nondominated point corresponding to the edge between i and j, we find the shortest path between that node pair minimizing the objective $U(z)=w'z_1(x)+(1-w')z_2(x)$.

For convex nondominated sets, the linear function results in a distinct central nondominated point. However, in minimizing $U(z)=w'z_1(x)+(1-w')z_2(x)$, there is a possibility to obtain a solution with objective values equal to either of the two extreme nondominated points, rather than a distinct solution with different objective values. This indicates that either the nondominated set is concave, or there are only two nondominated points in that tour's nondominated set. In such a case, a central nondominated point can be searched for by minimizing a Tchebycheff function ([19], p. 424). If neither a linear nor a Tchebycheff function results in a new central nondominated point, this indicates that the nondominated set contains only two points. We did not encounter such a situation in our experiments or in the literature on multiobjective combinatorial optimization.

3.1.2.4. Generating representative points. The fitted L_q distance function of a tour represents an approximate hypothetical frontier of that tour. We generate several equi-spaced hypothetical points on the fitted function to represent the tour. In Fig. 2(b), we show the five equi-spaced points on the L_q distance function that are selected to represent the tour. The two extreme nondominated points are always included among the representative points. The remaining points are chosen by breaking the curve into equi-spaced segments and it is unlikely for the central nondominated point to be one of the representative points. In Fig. 2(b), the two extreme nondominated points A and C are included among the representative points; whereas, the central nondominated point B is not.

3.1.2.5. Implementation in an EA. Let M denote the population size in an EA. In the initialization step, we generate M individuals. To evaluate each individual, we generate an L_q distance function as explained above. For each individual, we need the values of three points (two extreme nondominated points and a central nondominated point) which we compute using the nondominated points of the edges between the connected nodes in that tour. In a problem with eight nodes, each tour would be made up of eight connected node pairs. To obtain three points on the nondominated set of each tour, we need to find three points for each node pair in the tour. As a result, we need to find $24M$ points to evaluate the population. The left and right extreme nondominated points can be found at the start for all possible node pairs and utilized for every population member throughout the algorithm. For eight nodes, there are $8!/(6!2!) = 28$ possible node pairs. However, finding all possible central nondominated points is computationally prohibitive. For eight nodes, there are a total of $(8-1)!/2 = 2520$ possible tours for the symmetric TSP. Since each tour has its own w' value, we need to find the edge corresponding to w' between each of the eight node pairs for each of the 2520 tours if we want to initialize for all possible central points at the beginning. This adds up to finding a total of 20,160 $(= 2520 \times 8)$ points. One may think that we may not encounter all distinct tours and it may be more efficient to find the central points for the tours in our population. This may lead to duplication as we may encounter previous tours as new population members. Even if we can have an efficient book-keeping mechanism to avoid duplication, the number of distinct tours encountered in the population would still be computationally prohibitive for finding the corresponding central point for each of them. Instead, we use an approximation procedure to substantially reduce the computational requirement.

We initialize all the necessary information to fit the L_q distance functions throughout the algorithm. For each node pair, we solve 11 shortest path problems optimizing $U(z)$, setting w to the following 11 different values: ρ, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9 and $1-\rho$. We provide the two objective function values that correspond to the optimal edge between the node pairs for each w value as input. Once a tour becomes a population member, we find the left extreme nondominated point using (8) where the objective function values between the nodes correspond to the edges are found using $w=1-\rho$, and the right extreme nondominated point using (9) in a similar manner with $w=\rho$. In the derivation of the tour's central point, we first find w' using (11). Then we find which of the following 11 w values, ρ, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, $1-\rho$ is closest to w'. Using the objective function values of the node pairs corresponding to that w, we approximate the central nondominated point of the tour by (11).

After computing these nondominated points, we find the q value for the L_q distance function in Eq. (7) using binary search.

The next step is the selection of the representative points on the fitted L_q distance function. As explained above, we generate a predefined number of equi-spaced points on this function. While selecting the representative points, we need to make sure that these points are as close to the true nondominated points as possible for a better approximation of the set. Therefore, we also include the central nondominated point that we used in the approximation of the L_q distance function in our representative points since it is already a nondominated point. In order not to increase the number of representative points presented, we eliminate the representative point that is closest to the central point.

Let the number of representative points be s. After the selection of the representative points, we have a population of size $M{\cdot}s$. We refer to this population as the secondary population. For each individual in the initial population, we have s representatives in the secondary population with different objective values. In order to further save computational effort we reduce the population size to M selecting one representative point for each tour. We use the idea developed in the Elitist Non-Dominated Sorting Genetic Algorithm (NSGA-II) [3] for the selection of the representative point. In this method, they classify the population of individuals into fronts. The first front contains the nondominated points of the current population and they are all assigned a rank of 1. The points that become nondominated when the first front is removed are in the second front and are assigned a rank of 2. The other fronts are created in a similar manner. In our approach, we first rank all of the $M.\,s$ points for all tours. Then, we choose the representative point with the best rank to represent the tour. If more than one representative point has the same best rank, we choose one of them randomly. As a result, we obtain a population that has the same size as the initial population. All individuals in this population are represented with a point that is in the best front.

In our tests, we used 16 and 32-node problem instances, and generated five representative points on the L_q distance functions. In these tests, we rarely had to choose the representative points randomly. Specifically, for the 16-node problems, there was no need for random selection (since there was only one candidate representative point) for 85.16% of the tours. In the remaining cases, the representative point was selected between two, three, and four alternatives for 12.22%, 2.45%, and 0.17% of the tours, respectively. The results are similar for the 32-node problems. We did not consider an additional selection criterion since it is not clear an additional criterion would be useful when selecting a single point for each tour and since there are very few cases where we would need to use this criterion in our experiments.

3.1.3. *Crossover*

The genetic operators for the TSP are generally related with the representation scheme used. For path representation, there are many crossover operators in the literature. For a review of the operators, the reader is referred to Larranaga et al. [11]. One of the crossover operators used for the TSP is position-based crossover (POS). We use POS in our solution approach since the offspring take the order of visits from both parents. Let c denote the maximum number of crossover positions to be used. For any two parents, c positions are randomly selected. The nodes visited at those positions are directly passed to the offspring (from parent 1 to offspring 2 and from parent 2 to offspring 1). The maximum number of positions that can be selected is previously defined by the user; and each position is selected with crossover probability, *pcross*. The remaining bits of the chromosome for offspring i $(i=1{,}2)$ are filled from parent i $(i=1{,}2)$ maintaining the order of those bits in the parent. For example, assume parent 1 corresponds to tour (1-3-2-4-5) and parent 2 to tour (1-2-5-3-4). Let $pcross=1$ and $c=2$. Assume the two randomly selected positions are 1 and 4. Offspring 1 inherits bits 1 and 4 from parent 2 as (1-*-*-3-*) and offspring 2 from parent 1 as (1-*-*-4-*). Filling offspring 1 from parent 1, we obtain (1-2-4-3-5) and similarly filling offspring 2 from parent 2, we obtain (1-2-5-4-3).

3.1.4. *Mutation*

The mutation operator used is also specific to path representation. To keep feasibility, two nodes are randomly selected and their places are exchanged with mutation probability (*pmut*). This is the exchange mutation operator [11].

3.1.5. *Termination*

The algorithm stops when a predetermined number of generations are completed. It aims at finding efficient tours upon termination. However, since EAs are heuristic approaches and we approximate the tours by a combination of true and hypothetical points, it is not guaranteed that the obtained solutions are efficient. Some tours may be efficient relative to other tours in the final population just because of their hypothetical points. We therefore denote the resulting tours as potentially efficient tours.

3.2. *Implementation of the approach*

Our approach can be incorporated into any multiobjective EA. We demonstrate our approach by implementing it with the Elitist Non-Dominated Sorting Genetic Algorithm (NSGA-II) developed by Deb et al. [3]. Our algorithm utilizes the main structure of NSGA-II and adapts it to our situation by developing necessary mechanisms. The steps of the algorithm are given below.

3.2.1. *Elitist non-dominated sorting genetic algorithm (NSGA-II)*

NSGA-II is a multiobjective EA developed by Deb et al. [3]. It uses an elitism-preserving mechanism to favor good individuals based on ranking (as explained above) and a crowding distance measure. They refer to this method as a crowded tournament selection method (see [2], p. 235). Let the population size be M and the number of individuals in front i be F_i. Firstly the individuals in the first front are carried to the next population. If the population limit is not filled totally by the first (current) front $(M-F_1>0)$, the individuals in the next (second) front are carried into the population, and the procedure continues. When the addition of individuals in front i exceeds the population size, the solutions in this front that will be admitted is decided based on a crowding distance measure. This measure calculates the distance of an individual to its closest individuals in the objective function space. The individuals that have a bigger crowding distance; i.e. that are in a less crowded region, are carried to the next generation until the population limit is reached.

3.2.2. Steps of the algorithm

Step 0. Initialization
Generate an initial population of size M.
Step 1. Evaluation
1.1. Generate an L_q distance function for each individual in the population.
1.2. Produce s representative points on this L_q distance function to have a secondary population of size Ms.
1.3. Determine the rank of each point in the secondary population.
1.4. Select the representative point with the highest rank to represent the individual.
Step 2. Selection
Perform crowded tournament selection to choose the parents.
Step 3. Form the new population
3.1. Perform position based crossover.
3.2. Perform exchange mutation.

Step 4. Termination
Terminate if the predetermined number of generations have been completed. Otherwise, go to Step 1.

Using hypothetical points for the tours has computational advantages compared to using the true nondominated points of these tours. Firstly, potentially efficient tours are found without evaluating all efficient edges between nodes in that tour. This excludes many unpromising tours. As explained above, a problem with eight nodes has a total of 2520 different tours. If each connected node pair in each tour contains two efficient edges, there are 645,120 (2520×2^8) different alternatives to be considered. Finding and evaluating all the alternatives is computationally hard. In our approach we consider only three nondominated points of a tour and approximate the remaining nondominated points with hypothetical points. If a tour turns out to be inefficient, it is not evaluated even in the initial generations. This saves substantial computational effort without losing much information.

We use 11 different weights to find a set of efficient edges between each node pair. These single-objective shortest path problems are computationally easy to solve. When some tours are found to be inefficient and discarded from further evaluation, the corresponding edges between some nodes also become unnecessary. For example, if two nodes are not connected in any of the resulting potentially efficient tours, then there is no need to determine any of the efficient edges between these nodes. Finding the true efficient edges of only the potentially efficient tours saves substantial computational effort. Attempting to find all efficient edges of all node pairs at the start would be a prohibitive task for any practical problem instance.

3.3. Finding nondominated set of EA

As there is no guarantee for an EA to yield only efficient tours, we refer to the tours found by our EA as potentially efficient tours. Once we find a set of potentially efficient tours, we generate the true nondominated sets corresponding to these tours. This problem is computationally manageable using the procedure we explain below:

Step 1. Determine all node pairs that are connected in any of the potentially efficient tours.
Let the set of node pairs used in any of the potentially efficient tours be P'.
Step 2. Find all efficient edges between each connected node pair using a biobjective shortest path algorithm.
Step 3. Generate the nondominated set of the evolutionary algorithm.

For this step, we develop two methods. The first one is an exact method that computes all nondominated points for the potentially efficient tours. The second is an approximation method that approximates all nondominated sets of tours using L_q distance function, and finds a subset of the nondominated points.

For the first method, we develop the following model. Let Π_{pe} denote the set of potentially efficient tours, and $a_{\pi ij}$ be a parameter that takes value 1 if nodes (i,j) are consecutive in tour $\pi\in\Pi_{pe}$, 0 otherwise. Let y_{ijr} be a binary decision variable that takes value 1 if rth efficient edge is used between nodes i and j, and 0 otherwise, and v_π be a binary decision variable that takes value 1 if potentially efficient tour $\pi\in\Pi_{pe}$ is used, and 0 otherwise. To generate the whole nondominated set of the EA, the following biobjective model is constructed:

$$\text{Min}\quad z_1(y)=\sum_{(i,j)\in P'}\sum_{r\in R_{ij}} c^1_{ijr}y_{ijr} \tag{12}$$

$$\text{Min}\quad z_2(y)=\sum_{(i,j)\in P'}\sum_{r\in R_{ij}} c^2_{ijr}y_{ijr} \tag{13}$$

Subject to:

$$\sum_{\pi\in\Pi_{pe}} v_\pi=1 \tag{14}$$

$$\sum_{r\in R_{ij}} y_{ijr}=\sum_{\pi\in\Pi_{pe}} a_{\pi ij}v_\pi \quad (i,j)\in P' \tag{15}$$

$$y_{ijr}\in\{0,1\}\,(i,j)\in P',\ r\in R_{ij} \tag{16}$$

$$v_\pi\in\{0,1\}\ \pi\in\Pi_{pe} \tag{17}$$

For each nondominated point, the model selects one tour to be used (constraint (14)) and assigns one of the efficient edges to consecutive node pairs for that tour (constraint (15)).

To solve model (12)–(17), we use the ε-constraint method (see [1]) to sequentially generate the nondominated set of the EA. We treat one objective as a constraint defining an upper bound for that objective function. The other objective is treated as the objective function of the resulting single-objective model. Systematically changing the upper bound, we obtain all nondominated points. To implement the ε-constraint method, we write an upper bound, ε, on the second objective function as follows:

$$\sum_{(i,j)\in P'}\sum_{r\in R_{ij}} c^2_{ijr}y_{ijr}\le\varepsilon \tag{18}$$

We set this upper bound to a slightly lower value than the value obtained in the most recent solution for the second objective function. To guarantee generating all nondominated points, we set the reduction factor in the constraint right-hand-side to the smallest possible increment between the second objective function values. In our problems, the smallest possible increment between the objective function values is 0.001. Therefore, we set our reduction factor to this value and guarantee finding all true nondominated points. The ε-constraint method is very efficient to generate the true nondominated points for the general biobjective discrete problems. The complexity is equal to the complexity of the underlying single objective model and a distinct nondominated point is found with each solution of the model. Each time, we enforce the additional constraints (14)–(18) of the model and use

$$\text{Min}\quad \sum_{(i,j)\in P'}\sum_{r\in R_{ij}} c^1_{ijr}y_{ijr}+\delta\sum_{(i,j)\in P'}\sum_{r\in R_{ij}} c^2_{ijr}y_{ijr} \tag{19}$$

as the objective function. The second term in the objective function is added to prevent obtaining inefficient solutions where δ is a very small positive constant. We choose δ small enough to initially optimize the first objective and find the smallest second objective value for the corresponding solution (see [19], pp. 422–430).

In the second method, we find an approximate nondominated set of the EA using L_q distance function. For this, we develop the following steps:

Step 1. Fit L_q distance function to each potentially efficient tour, $\pi \in \Pi_{pe}$.
Step 2. Generate ND points on the L_q distance function for each potentially efficient tour, $\pi \in \Pi_{pe}$.
Step 3. Find the nondominated set of all points generated after Step 2. Divide the overall nondominated set to regions, where each region corresponds to one of the potentially efficient tours.

For this step, let the set of regions be G, where each region $g \in G$ is defined with a lower and an upper bound on its second objective value, $l(g)$ and $u(g)$, respectively. Let the potentially efficient tour corresponding to region $g \in G$ be $\pi(g)$.

Step 4. Fit an overall L_q distance function for the nondominated set.
Step 5. Generate H points on the L_q distance function and find the nondominated point corresponding to each point $h \in H$.

Let each point, $h \in H$, be denoted with its first and second objective values, h_1 and h_2, respectively. Since the points $h \in H$ are hypothetical, we find nondominated points corresponding to them. To find the nondominated point corresponding to each point h, we first find the region that point h lies. We check the lower and upper bounds defining the regions, and find the region, g_h, of h satisfying $g_h = \{g \in G | l(g) \le h_2 \le u(g)\}$. We then solve the following assignment model to find the point on the nondominated set of tour $\pi(g_h)$ corresponding to h.

Recall that y_{ijr} takes value 1 if the efficient edge r is used between nodes i and j, and 0 otherwise.

$$\text{Min } z_1(y) = \sum_{(i,j)\in\pi(g_h)} \sum_{r\in R_{ij}} c^1_{ijr} y_{ijr} + \delta \sum_{(i,j)\in\pi(g_h)} \sum_{r\in R_{ij}} c^2_{ijr} y_{ijr} \tag{20}$$

Subject to:

$$\sum_{(i,j)\in\pi(g_h)} \sum_{r\in R_{ij}} c^2_{ijr} y_{ijr} \le h_2 \tag{21}$$

$$\sum_{r\in R_{ij}} y_{ijr} = 1 \quad (i,j) \in \pi(g_h) \tag{22}$$

$$y_{ijr} \in \{0,1\} \quad (i,j) \in \pi(g_h),\ r \in R_{ij} \tag{23}$$

Each time model (20)–(23) is solved, a new nondominated point on the nondominated set of $\pi(g_h)$, corresponding to point h is found. (20) comprises the first objective function augmented with the second objective function. This augmentation prevents finding dominated solutions as explained above. We set h_2 as an upper bound on the second objective value of the nondominated point in (21). Constraint (22) indicates that for each node pair of tour $\pi(g_h)$, one of the efficient edges is selected.

Let the nondominated point corresponding to point h be $b(h)$. Since we are approximating the nondominated sets of all potentially efficient tours, some of $b(h)$, $h \in H$, may turn out to be dominated. After finding all $b(h)$, $h \in H$, and eliminating the dominated ones, we have the resulting approximate nondominated set of the EA.

We use both methods, *finding EA's nondominated set exactly* and *approximating EA's nondominated set*, in our computational studies.

4. Computational study

The generalized BOTSP is new in the literature and there are no established benchmark sets yet. Therefore, we developed the procedure to generate the problem instances for our computational studies. We test our algorithm on 10 different instances of symmetric generalized BOTSPs with 16, 32, and 64 nodes. For all problems, we have 20 efficient edges between each node pair; whose objective values are generated from a uniform distribution with range [0,100]. We present the results for different problem sizes in Sections 4.1–4.3. For the 16 and 32-node problems, we compare our EA's results with the true nondominated sets. For the 64-node problems, we compare our results with a well-distributed subset of the true nondominated set. The EA can easily handle much larger instances but creating benchmarks for the true nondominated sets of such instances is computationally prohibitive.

In addition to measuring the performance of our EA against the true nondominated set, we also compare it with a straightforward EA in Section 4.4.

We implemented our algorithm modifying the original C code of NSGA-II provided at http://www.iitk.ac.in/kangal/deb.shtml. We perform the tests on a PC with Intel Core i7-4770S CPU, 16 GB RAM. We use CPLEX solver for models (1)–(6), (12)–(17), and (20)–(23).

4.1. 16-node BOTSP instances

We first generate the true nondominated sets of all problem instances using the ε-constraint method. We treat the second objective as a constraint and change the bound on it between the second objective values of the extreme nondominated points.

We use a typical set of parameters for the EA that are based on the values proposed for NSGA-II [3]. We check the convergence behavior of the algorithm for 16 and 32-node problems. We compare the nondominated sets of the instances in every 5000 generations with the true nondominated set using the hypervolume (HV) indicator [22]. For the calculation of the HV indicator, we scale the points in the nondominated set with respect to the range of the true nondominated set. The results are given in Fig. 3 for the average of five 16-node and five 32-node problems. For both problem types, the HV indicator values are satisfactory for 10,000 and larger generations and the improvement beyond 30,000 generations is slight in general. We therefore run the EA for 30,000 generations for all instances. We use a crossover probability of 0.9 and a mutation probability of 0.1. We set the maximum number of crossover positions, c, to 10 and (M, s) to $(100,5)$. We make five replications for each test instance changing the random seed each time we run the algorithm.

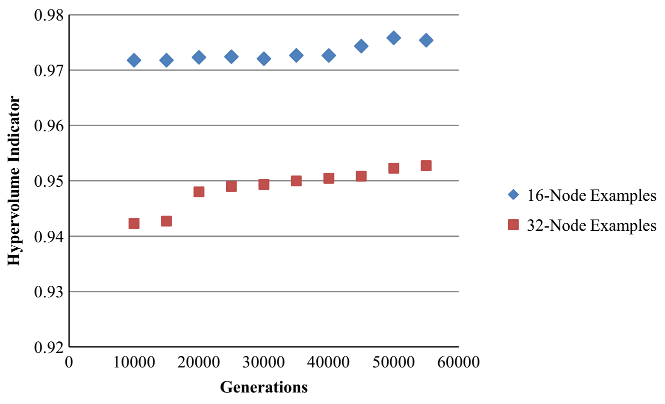

Fig. 3. Hypervolume indicator values for different number of generations.

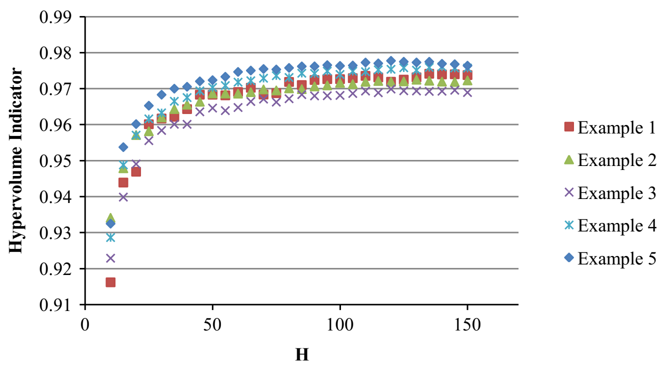

Fig. 4. Hypervolume indicator values for different sizes of nondominated set approximations.

For all replications, we use the two methods explained in Section 3.3 to find their nondominated sets. We set *ND* to 500. To decide on the number of points (H) to be generated on the L_q distance function for the overall nondominated set, we set H to values between 10 and 150, and compute the HV indicator values corresponding to each H value for five randomly selected instances. The results can be seen in Fig. 4.

As expected, with increasing number of nondominated points, the HV indicator values also increase. The results seem to stabilize around $H = 100$. The relation between H and HV may not be linear since all points found need not necessarily be in the nondominated set and some may turn out to be dominated. Therefore, more points do not always mean better HV indicator values. Considering the general trend between H and the HV indicator, we use $H = 100$ in approximating the nondominated set of the EA.

The first row in Table 1 indicates the percentages of HV of the true nondominated set explained by the results of the EA. The first three values correspond to the method, *finding EA's nondominated set exactly*, and the last three values correspond to the method, *approximating EA's nondominated set.*

When we find the nondominated sets of the EA exactly, we explain at least 98.0% of the true nondominated set. This value reduces to 96.6% when we approximate the nondominated sets. For both methods, the resulting nondominated sets are very close to the true nondominated set. We performed *paired-t test* to see if the two methods are statistically different. The first method always results in higher HV indicator values and the *paired-t test* resulted in the rejection of the hypothesis that the mean difference of the two methods is equal to 0. However, the HV indicator values corresponding to approximating the EA's nondominated set are also very high.

We next compare the CPU times of the ε-constraint method with that of EA. When calculating EA's duration, we also consider the time spent to generate EA's nondominated set. The results can be seen in the first row of Table 2. For the first and second methods, the average EA run times are less than 60% and 30% of that of the ε-constraint method, respectively. As expected, the difference grows substantially with problem size. The run times of the three methods turned out to be statistically different at 95% confidence level. At the expense of a slight sacrifice in the HV indicator value, the second method results in much shorter run durations than the first method.

Table 1
Hypervolume indicator values (16 and 32-node instances).

Problem instance	Finding EA's nondominated set exactly			Approximating EA's non-dominated set		
	Min	Average	Max	Min	Average	Max
16-node	0.980	0.988	0.996	0.966	0.975	0.989
32-node	0.945	0.957	0.969	0.930	0.947	0.965

4.2. 32-node BOTSP instances

Similar to the 16-node BOTSPs, we first generate the true nondominated sets of 32-node BOTSPs using the ε-constraint method. We then run the EA for 30,000 generations for all instances with crossover probability of 0.9, mutation probability of 0.1. We set the maximum number of crossover positions, c, to 20, and (M, s) to $(100,5)$. We replicate each test instance five times.

We compare the HV indicator values of the EA results with the true nondominated sets in the second row of Table 1. For these instances, the EA explains 95.7% of the true nondominated set with the first method on the average. For the second method, we generate $H = 100$ points, and explain on the average 94.7% of the true nondominated set. For these instances, the EA's nondominated sets are close to the true nondominated sets. Similar to the 16-node problems, the first method always results in higher HV indicator values and the *paired-t test* resulted in the rejection of the hypothesis that the mean difference of the two HV indicator values of methods is equal to 0.

We compare the CPU times of the EA with the duration to derive the whole nondominated sets in Table 2. The run times of the first method are around 10% of that of the ε-constraint method, and this decreases to 1% for the second method. Naturally, the run times of the three methods are statistically different at 95% confidence level. As the problem size increases, solving EA becomes substantially more efficient in terms of run times in return to a slight sacrifice in the HV indicator values.

4.3. 64-node BOTSP instances

The largest instance we address in this study is 64-node BOTSPs. The generation of the whole true nondominated set of a single problem takes several days. Therefore, we find well-distributed subsets of the true nondominated sets of these problems using the ε-constraint method instead of generating the whole sets. To do this, we first find the objective values of the extreme nondominated points of the problem. We then divide the range of the values of the second objective to 99 intervals. Setting ε each time to interval ends, we generate at most 100 nondominated points. The optimization model may give the same result for a number of consecutive ε values, therefore, the maximum number of nondominated points found is 100. We use this subset of nondominated points to compare against the results of the EA.

We run the EA for 30,000 generations for all instances with crossover probability of 0.9, mutation probability of 0.1. We set the maximum number of crossover positions, c, to 40, and (M, s) to $(100,5)$. We replicate each test instance five times. To have a fair comparison with the true nondominated set, we generate $H = 100$ nondominated points for the results of the EA using the second method.

We give the HV indicator values of the EA results with the true nondominated sets in Table 3. The EA explains 89.2% of the true nondominated set on average. The CPU time comparison of the ε-constraint method with the duration of EA can be seen in Table 4. The CPU times of the EA are less than 3% of the CPU times for generating a subset of the true nondominated sets. The run times are statistically different for the two methods at 95% confidence level. Although we generate the same number of nondominated points in the ε-constraint method and EA, the CPU times are completely different due to the difference in the complexity of the models solved in obtaining each nondominated point. Solving a

Table 2
CPU times (seconds) for the ε-constraint method and EAs (16 and 32-node instances).

Problem instance	ε-constraint method	EA+finding EA's nondominated set exactly			EA+approximating EA's nondominated set		
		Min	Average	Max	Min	Average	Max
16-node	1947.02	949.011	1111.120	1328.219	504.476	512.041	525.313
32-node	55675.40	3704.681	4773.668	6676.083	575.856	588.322	599.729

Table 3
Hypervolume indicator values (64-node instances).

Approximating EA's nondominated set		
Min	Average	Max
0.876	0.892	0.907

Table 4
CPU times (seconds) for the ε-constraint method and EAs (64-node instances).

ε-constraint method	EA+approximating EA's nondominated set		
	Min	Average	Max
29102.632	779.179	794.555	820.598

Table 5
Hypervolume indicator values of a-EA and s-EA.

	Example	a-EA	s-EA (30,000 generations)	s-EA (100,000 generations)
16-node problems	Example 1	0.9730	0.7540	0.7644
	Example 2	0.9718	0.8315	0.8452
	Example 3	0.9764	0.8056	0.8085
	Example 4	0.9743	0.8147	0.8277
	Example 5	0.9658	0.7882	0.7990
	Average	**0.9723**	**0.7988**	**0.8090**
32-node problems	Example 1	0.9424	0.6746	0.6878
	Example 2	0.9625	0.7036	0.7336
	Example 3	0.9506	0.7118	0.7381
	Example 4	0.9488	0.6576	0.6662
	Example 5	0.9485	0.6784	0.6963
	Average	**0.9506**	**0.6852**	**0.7044**

TSP with constraints (1)–(6) is much harder compared to solving model (20)–(23). The EA runs in very short durations and explains around 90% of the HV of a subset of the true nondominated sets.

We have not tested larger instances than 64-nodes, since solving for the true nondominated sets of much larger instances is computationally prohibitive. On the other hand, the EA can easily solve much larger instances. Looking at the trend between 16, 32, and 64-node instances, the performance of the EA deteriorates only slightly with increasing problem size. Some deterioration is expected since we keep the population size constant while the true nondominated set grows much larger with the problem size. If much larger problem instances need to be solved with the EA, it might be advisable to somewhat increase the population size.

4.4. *Comparison with a straightforward method*

A straightforward EA (s-EA) for the generalized BOTSP can be constructed by representing solutions with both their tours and the efficient edges used between consecutive nodes. In this section, we explain the s-EA that we develop and compare its results with both the true nondominated set and our EA's approximation results (which we will refer to as a-EA).

Since the generalized BOTSP is new in the literature, we developed a meaningful representation for the solutions and made modifications of various operators to make them suitable for the problem setting in s-EA. As explained above, each solution's representation contains information of both the tour and the efficient edges used between the consecutive nodes of the tour. In the crossover operator, the offspring gets c node positions from its parent along with the efficient edges used, and the remaining positions and efficient edges are taken from the other parent. In mutation, two nodes' positions are exchanged, and the efficient edges used between those nodes and their subsequent nodes are randomly assigned.

The s-EA is also implemented using NSGA-II. We use the same parameters for the s-EA to have a comparable computational difficulty as our EA. Specifically, we set the number of generations to 30,000, crossover probability to 0.9, mutation probability to 0.1, and the maximum number of crossover positions, c, to 10 for 16 nodes, and 20 for 32 nodes. Since we use 100 points to approximate our EA's nondominated set, we also set the population size to 100 in s-EA.

Table 5 shows the percentages of the HV of the true nondominated set explained by a-EA and by s-EA. As can be seen, a-EA explains a much larger percentage of the HV compared to that of s-EA. We also ran s-EA for 100,000 generations to see if the results would improve with longer runs. The improvement was negligible compared to more than three-fold increase in the run length. The results of s-EA for the 32-node instances are much worse as can be seen from the table.

We also present the solution sets of s-EA, a-EA, and the true nondominated set for one of the instances of 16-node and 32-node problems in Figs. 5 and 6, respectively. The results for the other instances are similar. Both the diversity and convergence of the population members of s-EA to the true nondominated set is poor. On the other hand, a-EA shows good diversity and convergence to the true nondominated set.

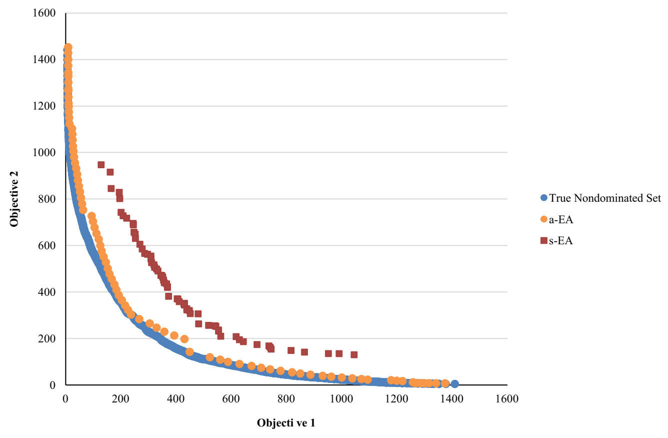

Fig. 5. The true nondominated set and the solutions of s-EA and a-EA (16-node instance).

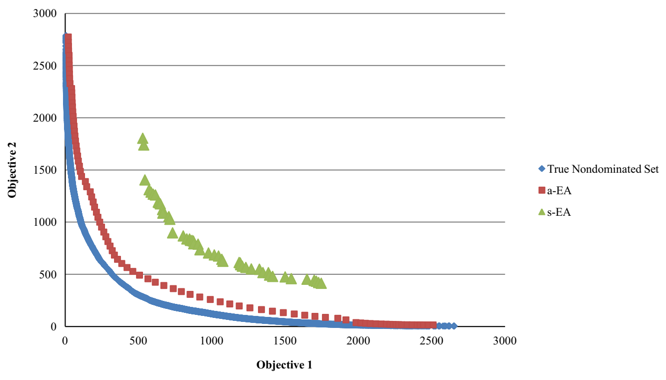

Fig. 6. The true nondominated set and the solutions of s-EA and a-EA (32-node instance).

5. Conclusions

We developed mechanisms that can be incorporated into any EA to approximate the nondominated points of the generalized BOTSPs. In this problem, there are two sets of decisions to be made: the order of visit to nodes and the edge to use between the connected nodes. Our algorithm results in tours that are potentially efficient by approximating the efficient edges of each tour. After determining a number of potentially efficient tours, we generate their overall nondominated sets. For this, we develop both an exact and an approximation method.

This approach reduces the problem size considerably. For the approximation of the overall nondominated set, the proportion of tours to be considered decreases substantially. Furthermore, there is no need to calculate all efficient edges between the nodes. If two nodes are not connected in any of the efficient tours, then we do not need to find the efficient edges between these nodes. For our algorithm, finding only three efficient edges is sufficient for each of the node pairs.

We demonstrate that the algorithm works well in approximating the nondominated sets on three different sets of problems; 16-node, 32-node, and 64-node generalized BOTSPs, for a typical set of parameter settings in NSGA-II. For the 16 and 32-node problems, we compare the approximate nondominated sets of the EA with the true nondominated sets. For the 16-node problems, with the exact generation of the EAs nondominated set, we explain 98.8% of the true nondominated set in less than two-thirds of the duration for generating the true nondominated set. When we approximate the nondominated set of the EA, we explain 97.5% of the nondominated set in one-fourth of the duration for generating the true nondominated set. The HV indicator values are around 95% for 32-node problems, and EA run times are around 1% of the duration for generating the true nondominated set. For 64-node problems, even though the exact nondominated sets could not be obtained, we explain a subset of the nondominated set. The EA runs in 3% of the CPU required for generating the nondominated set, and explains 90% of the nondominated set. Overall, the area of the nondominated region for both the true nondominated sets and the EAs nondominated sets are very close. Our EA is general for any number of nodes and its computational burden increases only slightly with the number of nodes. It is applicable to any BOTSP and promising especially for large problem instances for which a good approximation of its nondominated set is needed.

References

[1] Chankong V, Haimes YY. Multiobjective decision making: theory and methodology.New York: Elsevier-North-Holland; 1983.
[2] Deb K. Multi-objective optimization using evolutionary algorithms.Chichester: Wiley; 2001.
[3] Deb K, Pratap A, Agarwal S, Meyarivan T. A fast and elitist multiobjective genetic algorithm: NSGA–II. IEEE Trans Evolut Comput 2002;6(2):182–97.
[4] Ehrgott M. Multi objective optimization.Berlin: Springer; 2000.
[5] Gandibleux X, Beugnies F, Randriamasy S. Martins' algorithm revisited for multi-objective shortest path problems with a maxmin cost function. Q J Oper Res 2006;4(1):47–59.
[6] Jaszkiewicz A, Zielniewicz P. Pareto memetic algorithm with path reedgeing for bi-objective traveling salesperson problem. Eur J Oper Res 2009;193(3):885–90.
[7] Ke L, Zhang Q, Battiti R. Hybridization of decomposition and local search for multiobjective optimization. IEEE Trans Cybern 2014;44(10):1808–20.
[8] Köksalan M. A heuristic approach to bicriteria scheduling. Nav Res Logist 1999;46:777–89.
[9] Köksalan M, Lokman B. Approximating the nondominated frontiers of multi-objective combinatorial optimization problems. Nav Res Logist 2009;56:191–8.
[10] Köksalan M, Soylu B. Bicriteria p-hub location problem and evolutionary algorithms. Informs J Comput 2010;22(4):1–15.
[11] Larranaga P, Kuijpers CMH, Murga RH, Inza I, Dizdarevic S. Genetic algorithms for the travelling salesman problem: a review of representations and operators. Artif Intell Rev 1999;13:129–70.
[12] Lust T, Jaszkiewicz A. Speed-up techniques for solving large-scale biobjective TSP. Comput OR 2010;37(3):521–33.
[13] Lust T, Teghem J. The multiobjective traveling salesman problem: a survey and a new approach. Adv Multi-Object Nat Inspir Comput Stud Comput Intell 2010;272:119–41.
[14] Mooney P, Winstanley A. An evolutionary algorithm for multicriteria path optimization problems. Int J Geogr Inf Sci 2006;20(4):401–23.
[15] Özpeynirci Ö, Köksalan M. Multiobjective traveling salesperson problem on halin graphs. Eur J Oper Res 2009;196(155–161):155–61.
[16] Özpeynirci Ö, Köksalan M. Pyramidal tours and multiple objectives. J Glob Optim 2010;48(4):569–82.
[17] Paquete L, Stützle TA. Two-phase local search for the biobjective traveling salesman problem. In: Proceedings of the 2nd international conference on evolutionary multi-criterion optimization (EMO 2003), LNCS 2632. Springer; 2003. p. 479–493.
[18] Raith A, Ehrgott M. A comparison of solution strategies for biobjective shortest path problems. Comput Oper Res 2009;36(4):1299–331.
[19] Steuer RE. Multiple criteria optimization: theory, computation, and application.New York: Wiley; 1986.
[20] Tezcaner D, Köksalan M. An interactive algorithm for multi-objective route planning. J Optim Theory Appl 2011;150(2):379–94.
[21] Tezcaner Öztürk D, Köksalan M. An interactive approach for biobjective integer programs under quasiconvex preference functions. Ann Oper Res 2016. http://dx.doi.org/10.1007/s10479-016-2149-9.
[22] Zitzler E, Thiele L. Multiobjective optimization using evolutionary algorithms – a comparative case study. In: Agoston E. Eiben, Thomas Back, Marc Schoenauer, Hans-Paul Schwefel, editors. Proceedings of fifth international conference on parallel problem solving from nature (PPSN-V). Berlin, Germany: Springer; 1998. p. 292–301.