Calculating Complete and Exact Pareto Front for Multiobjective Optimization: A New Deterministic Approach for Discrete Problems

Xiao-Bing Hu, Ming Wang, and Ezequiel Di Paolo

***Abstract*—Searching the Pareto front for multiobjective optimization problems usually involves the use of a population-based search algorithm or of a deterministic method with a set of different single aggregate objective functions. The results are, in fact, only approximations of the real Pareto front. In this paper, we propose a new deterministic approach capable of fully determining the real Pareto front for those discrete problems for which it is possible to construct optimization algorithms to find the k best solutions to each of the single-objective problems. To this end, two theoretical conditions are given to guarantee the finding of the actual Pareto front rather than its approximation. Then, a general methodology for designing a deterministic search procedure is proposed. A case study is conducted, where by following the general methodology, a ripple-spreading algorithm is designed to calculate the complete exact Pareto front for multiobjective route optimization. When compared with traditional Pareto front search methods, the obvious advantage of the proposed approach is its unique capability of finding the complete Pareto front. This is illustrated by the simulation results in terms of both solution quality and computational efficiency.**

***Index Terms*—Multiobjective optimization, Pareto front, ripple-spreading algorithm, route optimization.**

I. Introduction

PARETO optimality originates from the concept of Pareto efficiency, which was proposed to study economic efficiency and income distribution [1]. In economics, a Pareto improvement, given an initial allocation of goods among a set of individuals, is defined as a change to a different allocation that makes at least one individual better off without making any other individual worse off, and an allocation is called “Pareto efficient” or “Pareto optimal” when no further Pareto improvements can be made [2]. This concept has been widely applied to multiobjective optimization, which aims to simultaneously optimize two or more often conflicting objectives subject to certain constraints [2], [3]. Multiobjective optimization has a very wide set of realistic applications including, to name a few, product and process design, finance, aircraft design, the oil and gas industry, and automobile design. In multiobjective optimization problems, decisions have to be made in order to achieve the best tradeoffs between two or more conflicting objectives. For example, maximizing profit and minimizing the cost of a product, maximizing performance and minimizing fuel consumption of a vehicle, and minimizing weight while maximizing the strength of a particular component are some pairs of conflicting objectives which may be considered in multiobjective optimization problems [4].

As is well known, it is, in general, impossible to identify a single solution that simultaneously optimizes each objective for nontrivial multiobjective problems. In such problems, improving an objective often makes other objectives suffer as a result. Therefore, Pareto optimality becomes a crucial concept for solving multiobjective problems. A tentative solution is called nondominated Pareto optimal or Pareto efficient if it cannot be eliminated from consideration by replacing it with another solution which improves one objective without worsening another one [2], [3]. Therefore, the goal of multiobjective optimization is to find such nondominated solutions and quantify the tradeoffs in satisfying the different objectives. All the Pareto-optimal solutions compose the Pareto-optimal set, and the projection of Pareto-optimal set in the objective space is called the Pareto front.

Most optimization methods are originally proposed to target single-objective optimization problems. As a first step, it is intuitive to try to extend such methods to multiobjective optimization problems by simply constructing a single aggregate objective function (AOF) which combines all of the objectives. A well-known example of AOF is the weighted linear sum of the objectives, but nonlinear AOFs are also often employed [2], [3]. However, the weighted sum method, like any method of selecting a single solution as preferable to all others, is essentially subjective, in that a decision maker needs to supply the weights. Moreover, this approach may prove difficult to implement if the Pareto front is not globally convex and/or the objective function

Manuscript received April 6, 2012; revised July 28, 2012; accepted October 2, 2012. Date of publication November 12, 2012; date of current version May 10, 2013. This work was supported in part by the China “973” Project under Grant 2012CB955404, by the Open Fund KF-2009-01 from the State Key Laboratory of Earth Surface Processes and Resource Ecology, and by the Seventh Framework Programme (FP7) of the European Union under Grant PIOF-GA-2011-299725. This paper was recommended by Editor E. Santos, Jr.

X.-B. Hu is with the State Key Laboratory of Earth Surface Processes and Resource Ecology, Beijing Normal University, Beijing 100875, China, and also with the School of Engineering, University of Warwick, Coventry CV4 7AL, U.K. (e-mail: dr_xiaobinghu@hotmail.co.uk).

M. Wang is with the State Key Laboratory of Earth Surface Processes and Resource Ecology, Beijing Normal University, Beijing 100875, China (e-mail: wangming@bnu.edu.cn).

E. Di Paolo is with the Ikerbasque, Basque Science Foundation, Centre for Research on Life, Mind and Society, University of the Basque Country, 20080 San Sebastian, Spain, and also with the Centre for Computational Neuroscience and Robotics, Department of Informatics, University of Sussex, Brighton BN1 9RH, U.K. (e-mail: ezequiel_dipaolo@ehu.es).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TSMCB.2012.2223756

to be minimized is not globally concave. Even if the Pareto front is globally convex, there is no theoretical guarantee that a given set of AOFs can find the complete Pareto front; in other words, some Pareto-optimal solutions might not be identified by a finite set of AOFs. Methods using AOFs largely focus on how to choose a proper set of AOFs such that the resulting solutions may distribute on the Pareto front as evenly as possible [5]–[9].

The objective way of characterizing multiobjective problems is to develop a Pareto-compliant ranking method, which, by favoring nondominated solutions, is capable of identifying multiple Pareto-optimal candidate solutions. In essence, a method following this practice needs to be able to generate and operate on a pool of candidate solutions—this is why population-based evolutionary approaches, such as genetic algorithm (GA), particle swarm optimization, and ant colony optimization, are currently very popular in the study of multiobjective optimization problems [10]–[22]. In these methods, no weight is required to predefine the relative importance of objectives. An initial population of candidate solutions is generated mainly in a stochastic manner, and then, a certain Pareto-compliant ranking method is employed to calculate the rank of each candidate solution. Based on the rank, the population is evolved for many generations until a given termination criterion is satisfied. Then, the last population of candidate solutions is considered to correspond to the Pareto front, and any solution in the final generation can be chosen for implementation depending on the decision maker's preferences. It should be noted that, due to the stochastic nature of multiobjective evolutionary approaches, what they output is actually an approximation of the Pareto front; in other words, it is likely that there is no Pareto-optimal solution at all contained in the final generation [10], [12]. This is different from methods using AOFs: as long as the optimization method employed can guarantee optimality, a given AOF will definitely result in a Pareto-optimal solution [9].

In short, most existing methods can only produce an incomplete or approximate Pareto front [5], [10], [23], [24]. This paper attempts to answer a fundamental question in the study of multiobjective optimization: Is it possible to calculate the complete exact Pareto front? The importance of finding the complete exact Pareto front is obvious, e.g., it will provide decision makers with the most accurate information with respect to optimality, and it can improve the robustness of systems with the least possibility of losing Pareto optimality (i.e., when a Pareto-optimal solution becomes infeasible due to external changes, it would be possible to have the most choices of nearby Pareto-optimal solutions that are still feasible). This paper will attempt to answer this important question with a particular focus on discrete problems. As pointed out in [23], very few results are available on the quality of the approximation of the Pareto front for multiobjective discrete problems. The work reported in this paper shows that finding the complete exact Pareto front is actually possible, theoretically and practically, for some classes of discrete problems.

In the methodology proposed in this paper, the most vital precondition is the availability of a method that is capable of finding the global kth best solution for any given k in terms of a given single objective. Although most optimization algorithms only calculate the global first best solution, researchers have also developed some effective algorithms to find solutions from the first best to the kth best. For example, in route optimization, there are quite a few methods reported to find the k shortest paths [25]–[27]. It should be noted that these path optimization methods themselves cannot calculate the complete Pareto front (they were not designed for multiobjective optimization at all), but their capability of finding the k shortest paths makes it possible to realize the methodology proposed in this paper for calculating the complete Pareto front. In other words, based on the capability of finding the k best solutions and the methodology reported in this paper, determining the complete exact Pareto front becomes theoretically and practically possible for some discrete problems.

The remainder of this paper is organized as follows. Section II provides the theoretical underpinnings for explaining under what circumstances the Pareto front to a multiobjective optimization problem can be exactly calculated in a deterministic way. Section III describes the details regarding how to design a deterministic search procedure to calculate the exact Pareto front for discrete problems. An example is then given in Section IV by designing a novel ripple-spreading algorithm for multiobjective route optimization problems (MOROPs). This paper ends with conclusions and discussions on future work in Section V.

II. Conditions for Finding the Real Pareto Front

First of all, we need a general mathematical formulation of multiobjective optimization problem

$$\min_{x} \left[g_1(x), g_2(x), \ldots, g_{N_{\mathrm{Obj}}}(x)\right]^{\mathrm{T}} \tag{1}$$

subject to

$$h_I(x) \le 0 \tag{2}$$

$$h_E(x) = 0 \tag{3}$$

$$x \in \Omega_X \tag{4}$$

where g_i is the ith objective function of the total N_{Obj} objective functions, h_I and h_E are the inequality and equality constraints, respectively, and x is the vector of optimization or decision variables belonging to the set of Ω_X. A Pareto-optimal solution x^* to the aforementioned problem is such that there exists no x that makes

$$g_i(x) \le g_i(x^*) \qquad \text{for all } i = 1, \ldots, N_{\mathrm{Obj}} \tag{5}$$

$$g_j(x) < g_j(x^*), \text{ for at least one } j \in [1, \ldots, N_{\mathrm{Obj}}]. \tag{6}$$

The projection of such an x^* in the objective space, i.e., the point $[g_1(x^*), g_2(x^*), \ldots, g_{N_{\mathrm{Obj}}}(x^*)]$, is called a Pareto point. For the aforementioned problem, there is usually a set of Pareto-optimal solutions, and the projection of this set in the objective space is called the Pareto front. Fig. 1 shows two intuitive examples of Pareto front, where $N_{\mathrm{Obj}} = 2$, i.e., there are two objective functions. Fig. 1(a) shows the Pareto front of continuous problems with infinite Pareto points, while Fig. 1(b) shows that of discrete problems with finite Pareto points. In Fig. 1(a), the shadow area is the projection of the set Ω_X in

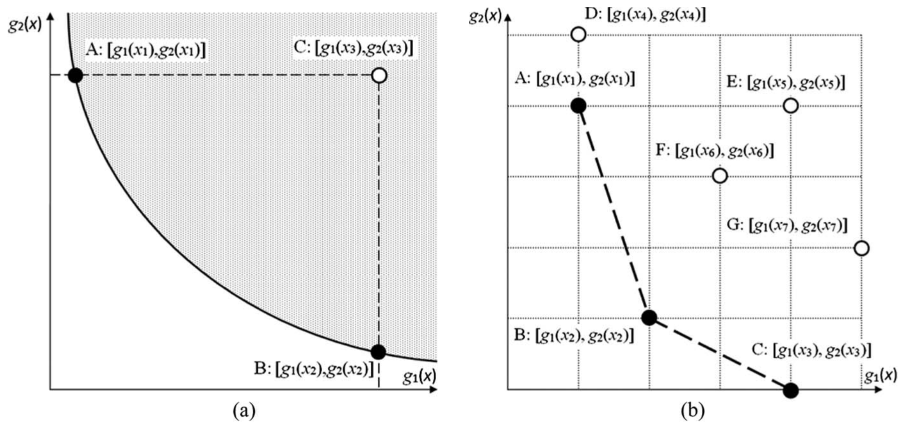


Fig. 1. Illustration of the Pareto front in objective space for continuous and discrete problems. (a) Continuous and infinite. (b) Discrete and finite.

the space of objective functions, the curve is the Pareto front, and the black points are Pareto points. Clearly, for x_3, any x whose projection is within the area circled by points A, B, and C will lead to better g_1 and g_2, while for any Pareto point (like A and B), there exists no x that satisfies (5) and (6). The Pareto points in Fig. 1(b) for discrete problems obviously have the same properties.

Based on the definition of Pareto front, we have the following statements for discrete-x-based multiobjective optimization problems.

Proposition 1: Suppose that we sort all discrete $x \in \Omega_X$ according to a certain objective function $g_j(x)$ and $x_{j,i}$ has the ith smallest g_j. For a given constant c, if there exists an index k that satisfies

$$g_j(x_{j,k}) \leq c < g_j(x_{j,k+1}) \tag{7}$$

then the number of Pareto points whose $g_j \leq c$ is no more than k, and all the associated x values are included in the set $[x_{j,1}, \ldots, x_{j,k}]$.

Proof: It is obvious that the number of Pareto points whose $g_j \leq c$ is no more than k, because all $x \in \Omega_X$ are sorted according to $g_j(x)$. Because all solutions are sorted, any x not belonging to the set $[x_{j,1}, \ldots, x_{j,k}]$ must have an index larger than k, which means $g_j(x) > c$.

Theorem 1: Suppose that we have a constant vector $[c_1, \ldots, c_{N_{\mathrm{Obj}}}]$, the element c_j is for objective function g_j, and after sorting all discrete $x \in \Omega_X$ according to each objective function g_j, we have k_j satisfying (7). If, for any $j = 1, \ldots, N_{\mathrm{Obj}}$

$$g_i\left(x_{j,k_j}\right) \leq g_i\left(x_{i,k_i}\right) \qquad \text{for all } i \neq j \tag{8}$$

then the total number of Pareto points is no more than

$$N_{\mathrm{PP}} \leq \sum_{j=1}^{N_{\mathrm{Obj}}} k_j \tag{9}$$

and all associated x values are included in the union set of $[x_{j,1}, \ldots, x_{j,k_j}]$, $j = 1, \ldots, N_{\mathrm{Obj}}$.

Proof: Assume that Theorem 1 is false; in other words, there is at least one Pareto point whose associated x^* is outside the union set of $[x_{j,1}, \ldots, x_{j,k_j}]$, $j = 1, \ldots, N_{\mathrm{Obj}}$. This means that $g_j(x^*) > c_j$, for all $j = 1, \ldots, N_{\mathrm{Obj}}$. Then, for a x_{i,k_i}, we have $g_j(x^*) > g_j(x_{i,k_i})$ for all $j = 1, \ldots, N_{\mathrm{Obj}}$ according to (7) and (8). This is, however, against the definition of Pareto point as given by (5) and (6). Therefore, Theorem 1 must be true.

Theorem 1 implies that, by sorting discrete x in terms of each single-objective function and then cross-checking the results sorted according to different objective functions, it is possible to work out a set of solutions which will cover all Pareto-optimal solutions. Condition (8) is used to judge whether k_j is large enough to guarantee the finding of such a cover set of solutions. Let us take Fig. 1(b) as an example, where there are seven solutions in total, i.e., x_1–x_7, corresponding to seven points in the objective space, i.e., point A to point G, respectively. If we sort x in an ascending order according to g_1, we have the sequence of x_4, x_1, x_2, x_6, x_5, x_3, and then x_7, while if we sort x in an ascending order according to g_2, we get the sequence of x_3, x_2, x_7, x_6, x_5, x_1, and then x_4. After sorting x, we choose $k_1 = 3$ and $k_2 = 2$ and have $x_{1,k_1} = x_2$ and $x_{2,k_2} = x_2$. Then, obviously, (8) holds for $k_1 = 3$ and $k_2 = 2$. The first three best solutions in terms of g_1 are x_4, x_1, and x_2, while the first two best solutions in terms of g_2 are x_3 and x_2. Therefore, Theorem 1 asserts that the complete Pareto front can be calculated by some or all of x_1, x_2, x_3, and x_4. The complete Pareto front in Fig. 1(b) is determined by x_1, x_2, and x_3, thus illustrating Theorem 1.

Proposition 1 and Theorem 1 can be extended into the case of continuous x value as follows.

Proposition 2: For a given objective function g_j and a constant c, suppose that there is a subset $\Omega_X(j, c)$ (maybe a single set, maybe a number of separated regions), such that, for all $x \in \Omega_X(j, c)$, $g_j(x) \leq c$, while for all $x \notin \Omega_X(j, c)$, $g_j(x) > c$. Then, for all Pareto points whose $g_j \leq c$, the associated x values belong to $\Omega_X(j, c)$.

Theorem 2: Suppose we have a constant vector $[c_1, \ldots, c_{N_{\mathrm{Obj}}}]$, the element c_j corresponding to objective function g_j and for each pair of c_j and g_j, there is a subset $\Omega_X(j, c_j)$, such that, for all $x \in \Omega_X(j, c_j)$, $g_j(x) \leq c_j$, while for all $x \notin \Omega_X(j, c_j)$, $g_j(x) > c_j$. Suppose that, for any $j = 1, \ldots, N_{\mathrm{Obj}}$

$$g_i(x) \leq c_j \qquad \text{for all } i \neq j \text{ and any } x \in \Omega_X(i, c_i) \tag{10}$$

then all Pareto points have their associated x values within the union set $\bigcup_{j=1}^{N_{\mathrm{Obj}}} \Omega_X(j, c_j)$.

One can easily derive the proofs of Proposition 2 and Theorem 2 by referring to those of Proposition 1 and Theorem 1. Basically, one can see that, theoretically, Theorem 1 and Theorem 2 give an envelope covering the complete exact Pareto front for discrete and continuous problems, respectively. Although a continuous problem usually has an infinite Pareto front, Theorem 2 might still give a complete cover as long as relevant c_j and $\Omega_X(j, c_j)$ can be identified. From an algorithm design point of view, Proposition 1 and Theorem 1 are more useful when the Pareto front is discrete and finite. In particular, most combinatorial optimization problems are based on discrete x values. As mentioned in [23], very few results are available on the quality of the approximation of the Pareto front for multiobjective discrete problems. In this paper, we will focus our study on discrete-x-value-based multiobjective optimization problems and propose a methodology that is capable of, rather than providing an approximation of the true Pareto front, determining the complete exact Pareto front.

It should be emphasized that the methodology explained in the next few sections is not suitable for continuous problems, particularly due to their uncountable set of solutions. Further effort is needed to develop more practicable theorems for continuous problems, and for this, Theorem 1 and Theorem 2 might be able to give a little inspiration. In future work, it will be worth investigating further whether Theorem 2 or an improved version might be applicable to some specific classes of continuous problems.

III. General Deterministic Methodology

Based on Proposition 1 and Theorem 1, in this section, we will propose a general deterministic procedure to calculate the complete exact Pareto front for discrete-x-value-based multiobjective optimization problems. We proceed along the followings steps.

Step 1) Design a problem-dependent deterministic algorithm that is capable of calculating any global kth best solution in terms of a single-objective function g_j, for any $j = 1, \ldots, N_{\mathrm{Obj}}$. Initialize two integers $k > 0$ and $\Delta k \geq 1$. Usually, we may set $\Delta k = 1$. Initialize the Pareto-front-associated x value set as $\Omega_{\mathrm{PFX}} = \emptyset$.

Step 2) Calculate the $k+1$ global best solutions in terms of each single-objective function g_j, and sort the solutions as $[x_{j,1}, \ldots, x_{j,k}, x_{j,k+1}]$ accordingly.

Step 3) If, for any $j = 1, \ldots, N_{\mathrm{Obj}}$

$$g_j(x_{j,k}) < g_j(x_{j,k+1}) \tag{11}$$

$$g_i(x_{j,k}) \leq g_i(x_{i,k}) \qquad \text{for all } i \neq j \tag{12}$$

then go to Step 4). Otherwise, increase k by Δk, i.e., $k = k + \Delta k$, and then, go to Step 2).

Step 4) Calculate the union set of $[x_{j,1}, \ldots, x_{j,k}]$, $j = 1, \ldots, N_{\mathrm{Obj}}$, denoting as Ω_{UX}.

Step 5) For any $x \in \Omega_{\mathrm{UX}}$, if there exists no $\breve{x} \in \Omega_{\mathrm{UX}}$ such that $g_i(\breve{x}) \leq g_i(x)$, for all $i = 1, \ldots, N_{\mathrm{Obj}}$, and $g_j(\breve{x}) < g_j(x)$, for at least one $j \in [1, \ldots, N_{\mathrm{Obj}}]$, then we know that the point $[g_1(x), \ldots, g_{N_{\mathrm{Obj}}}(x)]$ is a Pareto point. Therefore, add x into Ω_{PFX}, i.e., $\Omega_{\mathrm{PFX}} = \Omega_{\mathrm{PFX}} + \{x\}$.

The essential idea of the aforementioned procedure is to find a proper k that makes (11) and (12) hold. The calculation of the union set Ω_{UX} implies that different objective functions may identify common point(s) in the objective space. We have the following lemma to show that there is at least one common Pareto point shared by those identified by any two objective functions when k is properly valued.

Lemma: Suppose that a k makes (11) and (12) hold true and $[x_{j,1}, \ldots, x_{j,k}]$ denotes the k best solutions in terms of objective function g_j, $j = 1, \ldots, N_{\mathrm{Obj}}$. Then, for any i and j, $i \neq j$, $i \in [1, \ldots, N_{\mathrm{Obj}}]$, and $j \in [1, \ldots, N_{\mathrm{Obj}}]$, there always exist $1 \leq m_i \leq k$ and $1 \leq m_j \leq k$, such that $x_{i,m_i} = x_{j,m_j}$, and the point $[g_1(x_{i,m_i}), \ldots, g_{N_{\mathrm{Obj}}}(x_{i,m_i})]$ is a Pareto point.

Proof: Because $[x_{i,1}, \ldots, x_{i,k}]$ denotes the k best solutions in terms of objective function g_i, then, according to (11) and (12), obviously, at least, we have $x_{j,k} \in [x_{i,1}, \ldots, x_{i,k}]$. In other words, let $m_j = k$, and then, we always have that x_{j,m_j}, i.e., $x_{j,k}$, belongs to the set $[x_{i,1}, \ldots, x_{i,k}]$ for any $i \neq j$, $i \in [1, \ldots, N_{\mathrm{Obj}}]$.

Whether a value of k is proper largely depends on the landscape of the problem solution space. Fig. 2 shows two examples, where it is assumed there are six candidate solutions, i.e., x_1–x_6, and two objective functions, i.e., g_1 and g_2, are considered. In Fig. 2(a), both g_1 and g_2 change rather randomly as the solution changes from x_1 to x_6. When $k = 2$, we have x_5 and x_2 as the first two best solutions according to g_1 and x_4 and x_2 as the first two best solutions according to g_2. Here, x_4 and x_2 actually have the same value of g_2, and we treat x_2 as the second best solution in terms of g_2, because x_2 has a smaller value of g_1, which could be more likely to satisfy (12). Then, we check whether (11) and (12) hold for g_1 and g_2 when $k = 2$, respectively. They do hold when $k = 2$. Therefore, the complete Pareto front should be determined by some or all of x_2, x_4, and x_5. After the further check in Step 5), the complete Pareto front is revealed as $\{[g_1(x_5), g_2(x_5)], [g_1(x_2), g_2(x_2)]\}$. There is a common Pareto point $[g_1(x_2), g_2(x_2)]$ identified by the two different objective functions. In Fig. 2(b), g_1 and g_2 follow completely opposite monotonous trends as the solution changes from x_1 to x_6. If we still use $k = 2$, then x_1 and x_2 are the first two best solutions according to g_1, and x_6 and x_5 are the first two best solutions according to g_2. From Fig. 2(b), one can see clearly that (12) is not satisfied when $k = 2$, and no common Pareto point is identified either. When we increase the value to $k = 4$, then both (11) and (12) hold, and four Pareto points are identified by g_1, i.e., associated with x_1–x_4, while four Pareto points are identified by g_2, i.e., associated with x_3–x_6. In this case, the complete Pareto front is composed of six Pareto points associated with x_1–x_6, respectively, and there are two common Pareto points $[g_1(x_3), g_2(x_3)]$ and $[g_1(x_4), g_2(x_4)]$ identified by the two objective functions. With a landscape of solution space similar to that of Fig. 2(b), k usually has to be given a large value in order to make (11) and (12) hold. For instance, in

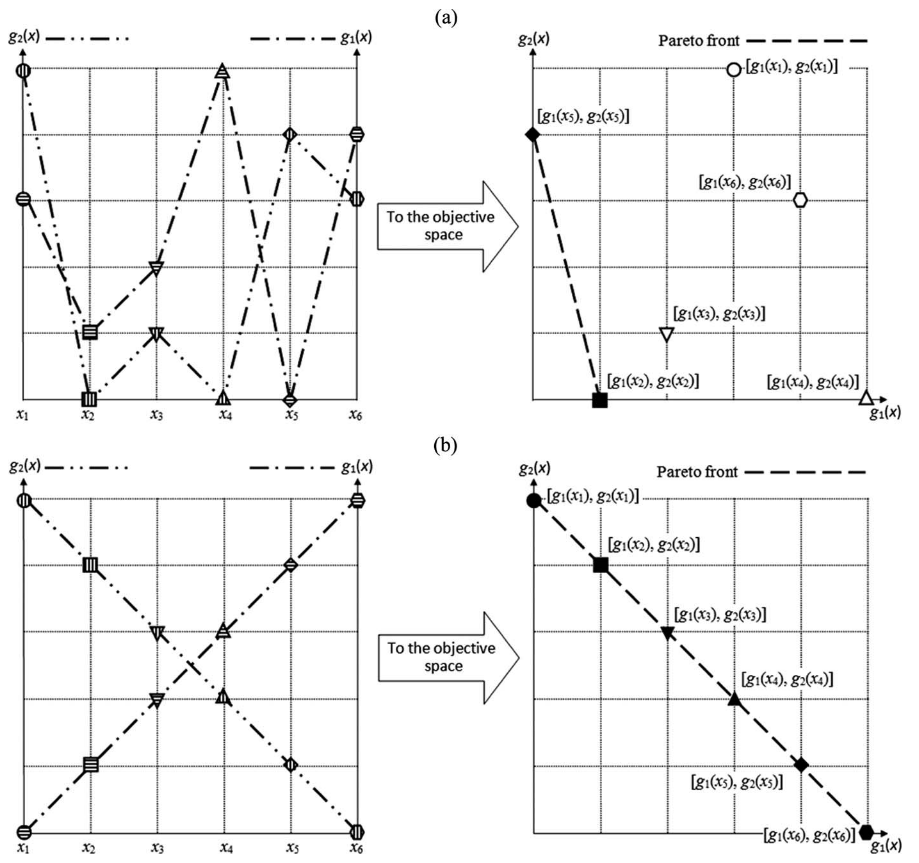


Fig. 2. Choosing a proper k to determine the Pareto front. (a) $k = 2$, two Pareto points, and one common Pareto point identified by two objective functions. (b) $k = 4$, six Pareto points, and two common Pareto points identified by two objective functions.

the case of Fig. 2(b), $k = 4$ is quite large when compared with the solution space size of six.

When the aforementioned procedure is compared with Theorem 1, one can see that no constant vector $[c_1, \ldots, c_{N_{\mathrm{Obj}}}]$ is used, but instead, an adjustable parameter k is shared by all objective functions. This is for the convenience of algorithm design. However, using the same k for all objective functions is likely to make it difficult for (11) and (12) to be satisfied. In other words, Steps 2) and 3) will have to be repeated for many times before a suitable k is found. Such a k is often large, which often implies a huge computational cost. Therefore, we can improve the aforementioned procedure by replacing k with an integer vector $[k_1, \ldots, k_{N_{\mathrm{Obj}}}]$, where k_j, $j = 1, \ldots, N_{\mathrm{Obj}}$, is exclusively for the objective function g_j. Then, we need to modify Steps 2) and 3) in the following way.

Step 2) For any newly initialized or updated k_j, calculate the $k_j + 1$ global best solutions in terms of the single-objective function g_j, and sort the solutions as $[x_{j,1}, \ldots, x_{j,k_j}, x_{j,k_j+1}]$ accordingly.

Step 3) If, for any $j = 1, \ldots, N_{\mathrm{Obj}}$

$$g_j(x_{j,k_j}) < g_j(x_{j,k_j+1}) \tag{13}$$

$$g_i(x_{j,k_j}) \leq g_i(x_{i,k_i}) \qquad \text{for all } i \neq j \tag{14}$$

then go to Step 4). Otherwise, fix k_j for any j that has (13) and (14) both satisfied, and increase k_j by Δk for the j that has (14) satisfied for the most i values. Go to Step 2).

It is easy to derive that, once (13) and (14) are satisfied under a certain k_j, they will always hold for this k_j, no matter how other k_i, $i \neq j$, increases. Therefore, the modified Step 3) only increases the k_j that has the most potential to get (13) and (14) satisfied for the associated j. This may significantly improve the computational efficiency. Usually, the Pareto-front-associated solutions are just a tiny fraction of the entire solution space, and therefore, (13) and (14) may often be satisfied under reasonably small k_j for $j = 1, \ldots, N_{\mathrm{Obj}}$. However, in the worst scenario where each solution in the solution space leads to its own unique Pareto point, k_j could be as large as up to about the half of the solution space size, in order to guarantee the completion of exact Pareto front, just like that shown in Fig. 2(b). Of course, in such a case, no other method can guarantee the completion without searching the entire solution space either.

The landscape or the distribution pattern of the projections of all solutions in the objective space largely determines the searching efficiency of the proposed procedure, as well as the possibility of improving the proposed procedure further. Fig. 3 shows whether an area in the objective space will be explored by the proposed procedure. Suppose that the curve A–B–C defines the complete Pareto front in Fig. 3. Then, we know for sure that there is no need for the proposed procedure to explore Zone I at all. Depending on the final $[k_1, \ldots, k_{N_{\mathrm{Obj}}}]$, Zone II could be explored. For a given $[k_1, \ldots, k_{N_{\mathrm{Obj}}}]$ which makes (13) and (14) hold, the proposed procedure actually only explores Zone III and Zone IV before finding the complete Pareto front. If most solutions to a problem are projected into Zone I,

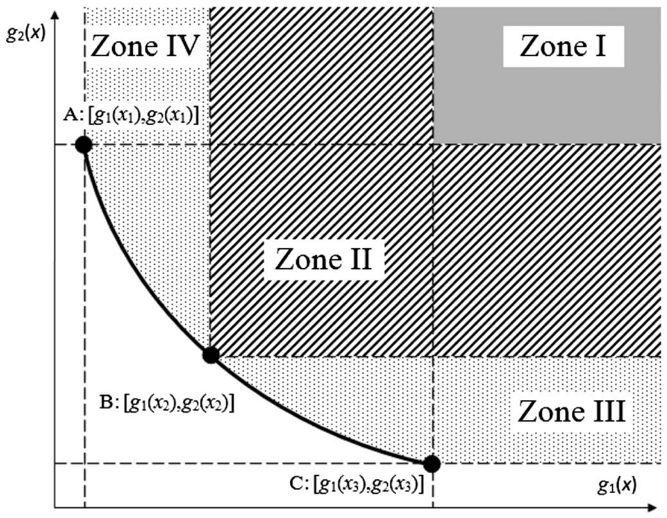

$g_2(x)$
Zone IV
Zone I
A: $[g_1(x_1), g_2(x_1)]$
Zone II
B: $[g_1(x_2), g_2(x_2)]$
Zone III
C: $[g_1(x_3), g_2(x_3)]$
$g_1(x)$

Fig. 3. Illustration of areas to explore for Pareto front.

which is often the case, then the proposed procedure will exhibit a high searching efficiency. If we can find a $[k_1, \ldots, k_{N_{\mathrm{Obj}}}]$ to minimize the total number of points in Zone III and Zone IV, then we can maximize the searching efficiency. This is no doubt worthy of further investigation in future study.

The most difficult part to realize the aforementioned procedure is Step 1). This is because of the following.

1) For many problems, it is very difficult, if not impossible, to develop a time-efficient deterministic algorithm to find the k global optimal solutions.
2) Usually, a deterministic algorithm is designed only to search the global first best solution, and some special complex modifications must be made in order to extend such an algorithm to calculate the k best solutions (for instance, to find the k shortest paths, a method was reported in [27] which needs to apply A^* algorithms and to reconstruct and transform the route network repeatedly).
3) Population-based evolutionary methods can hardly help here, because, due to their stochastic nature, they cannot even guarantee the finding of the first best solution.

When such a deterministic algorithm as required in Step 1) is not available, it is often possible to design an effective algorithm, deterministic or stochastic, to calculate suboptimal solutions in terms of a single-objective function. Although such an algorithm cannot tell whether a solution is optimal or the kth optimal, the proposed general procedure mentioned earlier can at least work out an approximation of the Pareto front, just like existing methods do.

IV. Case Study

In this section, we examine a case study to demonstrate the practicability and the effectiveness of the proposed methodology for determining the complete exact Pareto front for discrete problems and also to verify the theoretical results given in the aforementioned sections.

A. MOROP

The case study chosen here is a MOROP, which has a broad application background [28]. To be able to better visualize and verify the completion of the Pareto front, we simply consider two conflicting objectives: One is the distance, and the other is related to the inverse of the distance. A mathematical description of this MOROP is given as follows.

Assuming that a route network $G(V, E)$ is composed of node set V and connection set E. V has N_N different nodes including the source and the destination, and then, this route network can be recorded as an $N_N \times N_N$ adjacent matrix A. The matrix entry $A(i, j) = 1$, where $i = 1, \ldots, N_N$ and $j = 1, \ldots, N_N$, defines a connection, i.e., a direct route from node i to node j. Otherwise, $A(i, j) = 0$ means no direct route. We assume that $A(i, i) = 0$, i.e., no self-connecting route is allowed. In this paper, there are two values associated with each connection $A(i, j)$: One is the distance between node i to node j, denoted as $D(i, j)$, and the other is an artificial cost defined as

$$C(i, j) = D_{\mathrm{Max}} \times D_{\mathrm{Min}} / D(i, j) \tag{15}$$

where D_{Max} and D_{Min} are the maximal and minimal connection lengths in the route network, respectively. Suppose that a candidate route is recorded as an integer vector whose element $R(i) = j$ means that node j is the ith node along the route, $i = 1, \ldots, N_L$, and $j = 1, \ldots, N_N$, where N_L tells how many nodes, including the source and the destination nodes, are included in the route. Obviously, $R(1)$ is the source, and $R(N_L)$ is the destination. In this paper, for the sake of simplicity, we assume that no node can appear in a route more than once, which means that no loop is allowed in a route. The mathematical form of this assumption is the following constraint:

$$R(i) \neq R(j), \text{if } i \neq j, i = 1, \ldots, N_L, \text{and } j = 1, \ldots, N_L. \tag{16}$$

Then, for a given route, we can calculate its distance and artificial cost from the source to the destination

$$g_1(R) = \sum_{i=1}^{N_L - 1} D(R(i), R(i+1)) \tag{17}$$

$$g_2(R) = \sum_{i=1}^{N_L - 1} C(R(i), R(i+1)) \tag{18}$$

which are used as two objective functions in the MOROP. From the definition of the artificial cost $C(i, j)$ in (15), one can easily see that g_1 and g_2 are conflicting with each other. For a single connection, (15) shows that the maximal/minimal distance is associated with the minimal/maximal artificial cost. However, it should be noted that the maximal/minimal g_1 is not necessarily associated with the minimal/maximal g_2.

The MOROP is then mathematically formulated as the following minimization problem:

$$\min_{R \in \Omega} g_1(R) \tag{19}$$

$$\min_{R \in \Omega} g_2(R) \tag{20}$$

subject to (16) and

$$A(R(i), R(i+1)) = 1, \qquad \text{for } i = 1, \ldots, N_L - 1 \tag{21}$$

where Ω is the set of all feasible routes connecting the source and the destination.

Population-based evolutionary algorithms such as nondominated sorting GA (NSGA) [13] and the improved version NSGA-II [14] can directly be applied to approximate the Pareto front with no need to modify the aforementioned minimization problem. AOF-based methods require integrating g_1 and g_2 into a single-objective function, i.e., AOF. In this paper, we use a weight to construct a linear AOF

$$g_{\mathrm{AOF}}(R) = \alpha \times g_1(R) + (1-\alpha) \times g_2(R) \tag{22}$$

where $0 \leq \alpha \leq 1$ is the weight which determines the importance of g_1 and g_2 in the AOF. To use the AOF to resolve the MOROP defined by (19)–(21), we need to predetermine a set of different values for α, denoted as $[\alpha_1, \ldots, \alpha_{N_W}]$, where N_W is the number of α values in the set. Each α value determines a different AOF and possibly leads to a different Pareto point, but much more likely, most α values result in a same Pareto point. By repeatedly minimizing $g_{\mathrm{AOF}}(R)$, each time with a different weight in $[\alpha_1, \ldots, \alpha_{N_W}]$, one may find some Pareto points, but there is no guarantee of finding the complete exact Pareto front. One cannot even estimate, by either population-based evolutionary algorithms or AOF-based methods, how many Pareto points there might be in the complete exact Pareto front.

B. Ripple-Spreading Algorithm to Find the Complete Pareto Front for the MOROP

As emphasized in Sections I–III, the key ingredient to determine the complete exact Pareto front is to design a method that is capable of calculating the general kth best solution. There are already some algorithms reported to find the k shortest paths (e.g., those in [25]–[27]), which can be integrated into the methodology proposed in Section III, in order to find the complete exact Pareto front for the MOROP. Here, we will develop a brand-new algorithm to find the k shortest paths, in order to encourage the development of algorithms with the same capability, so that the proposed methodology for determining the true Pareto front will become more practicable.

Like that in [29], the new algorithm is inspired by the natural ripple-spreading phenomenon along a quiet water surface. There is a very simple optimization principle in the natural ripple-spreading phenomenon: A ripple travels at the same speed in all directions, so it reaches spatial points in order according to their distances to the ripple epicenter, i.e., the closest spatial point is the first to be reached by the ripple, and the kth closest spatial point is the kth to be reached. Now, we let a ripple start from the source and travel only along connections between nodes in a route network. Once the ripple reaches a new node, it will trigger a new ripple with that node as the center, and the new ripple will also travel only along connections and trigger ripples at farther away nodes. This is like a ripple relay race from the source to the destination. Since all ripples travel at the same speed, it is obvious that the ripple that reaches the destination first has traveled along the first shortest route and the ripple that reaches the destination in the kth place has traveled along the kth shortest route. As will be shown later, in the simulation of this ripple relay race, we do not need to calculate any route length from the source to any intermediate node, either the first shortest or the kth shortest. All we need to do is to simply simulate the spreading and activation of ripples, i.e., whether a ripple reaches the end node of a connection, and which ripple is triggered by which ripple. As ripples reach the destination one by one, we can then easily know the kth shortest route by simply backtracking the ripple in the kth place to reach the destination. Obviously, the length of a connection can be defined as anything rather than physical distance, and then, the ripple relay race can identify the general kth best route in terms of any single-objective function. The detailed ripple-spreading algorithm to find the general kth best route in terms of a given objective function is given as follows. For the sake of simplicity but without losing generality, we assume that the source is node 1 and the destination is node N_N.

Step 1) Set up the length of each connection in the route network according to a given objective function. For instance, when g_1 in (17) is the concern, set the length of connection $A(i,j)$ as $L(i,j) = D(i,j)$, while when g_2 in (18) is concerned, set $L(i,j) = C(i,j)$. Set up N_{NBR} to indicate how many best routes need to be found. Set the time instant as $t = 0$. Set the number of ripples that have, so far, reached the destination by time instant t as $N_{\mathrm{RRD}}(t) = 0$.

Step 2) Let the ripple-spreading speed be a constant $0 < s \leq L_{\mathrm{Min}}$, where L_{Min} is the minimal connection length in the route network. Set the current number of ripples as $N_R = 1$. Initialize the first ripple: The epicenter of ripple 1 is the source, i.e., Epicenter(1) $= 1$; the radius of ripple 1 is zero, i.e., Radius(1) $= 0$; and no ripple triggers ripple 1, i.e., Trigger(1) $= 0$. Initialize a node set that ripple 1 needs to visit, denoted as $\Omega_{N2V}(1)$, and it includes the end nodes of those connections which start from the source.

Step 3) Do, while $N_{\mathrm{RRD}}(t) < N_{\mathrm{NBR}}$, the following.

Step 3.1) Let $t = t + 1$.

Step 3.2) For any ripple i, $i = 1, \ldots, N_R$, if its node set to visit is not empty, i.e., $\Omega_{N2V}(i) \neq \varnothing$, let Radius($i$) = Radius($i$) + s. For any node j belonging to $\Omega_{N2V}(i)$, i.e., $j \in \Omega_{N2V}(i)$, if it has just been reached by ripple i, i.e., Radius(i) $\geq L$(Epicenter(i), j), then remove node j from $\Omega_{N2V}(i)$, i.e., $\Omega_{N2V}(i) = \Omega_{N2V}(i) - \{j\}$. Let a new ripple be triggered by ripple i at node j, so $N_R = N_R + 1$, and initialize the new ripple: Epicenter(N_R) $= j$, Trigger(N_R) $= i$, and Radius(N_R) = Radius(i) $- L$(Epicenter(i), j); initialize the node set to visit $\Omega_{N2V}(N_R)$ according to $A(j,k) = 1$, $k = 1, \ldots, N_N$, but excluding all epicenters whose ripples ancestor the new ripple. If node j is the destination, let $N_{\mathrm{RRD}}(t) = N_{\mathrm{RRD}}(t-1) + 1$, and set $\Omega_{N2V}(N_R) = \varnothing$.

Step 4) If $N_{\text{RRD}}(t) = N_{\text{NBR}}$, go to Step 5). Otherwise, choose $N_{\text{NBR}} - N_{\text{RRD}}(t-1)$ ripples from the ripples which has just been triggered at the destination node during the last time instant t, and these chosen ripples should have the largest values of Radius(i), which means that they are triggered by those ripples that are in the first $N_{\text{NBR}} - N_{\text{RRD}}(t-1)$ places to reach the destination during the last time instant t. These $N_{\text{NBR}} - N_{\text{RRD}}(t-1)$ ripples which have reached the destination during time instant t plus those $N_{\text{RRD}}(t-1)$ ripples which have reached the destination by time instant $(t-1)$ will be used in Step 5).

Step 5) Recover the first N_{NBR} best routes by backtracking the ripples which are in the first N_{NBR} places to reach the destination. Suppose that ripple i is among the first N_{NBR} ripples to reach the destination. Then, first, we construct a backward route (i.e., from the destination from the source) associated with ripple i by the following three substeps.

Step 5.1) Let the current ripple be ripple i, i.e., $n_{\text{CR}} = i$. Let the current node in the backward route $\overleftarrow{R}$ be considered as the mth node of route $\overleftarrow{R}$, i.e., $\overleftarrow{R}(m)$, initialize $m = 1$, and let $\overleftarrow{R}(1) = N_N$, i.e., the destination.

Step 5.2) Repeat until the source is reached, i.e., until Epicenter$(n_{\text{CR}}) = 1$

$$m = m + 1, \quad n_{\text{CR}} = \text{Trigger}(n_{\text{CR}}) \tag{23}$$

$$\overleftarrow{R}(m) = \text{Epicenter}(n_{\text{CR}}). \tag{24}$$

Step 5.3) Assuming that the backward route $\overleftarrow{R}$ is composed of N_L nodes, then recover the normal route R (i.e., from the source to the destination) associated with ripple i as follows:

$$R(m) = \overleftarrow{R}(N_L + 1 - m), \qquad m = 1, \ldots, N_{\text{L}}. \tag{25}$$

Put simply, Steps 1) and 2) in the aforementioned ripple relay race process are there to set up the route network data and the initial ripple-spreading data, respectively. Step 3) is the core of the ripple relay race process, particularly Step 3.2), which basically checks and records which active ripple has reached which new nodes at each time. If a new ripple is to be triggered at a new node, Step 3.2) also needs to calculate the initial radius of the new ripple by simply deducting the connection length of the associated two nodes from the radius of the stimulating ripple. Sometimes, by the last time instant of the ripple relay race, the destination may have been reached by more than N_{NBR} ripples in total. In this case, Step 4) makes sure that only those ripples which have arrived at the destination in the first N_{NBR} places will be used in Step 5) to recover the N_{NBR} best routes.

Fig. 4 shows an intuitive illustration to show how the general kth shortest route is identified by the ripple relay race. In Fig. 4, only those ripples with $\Omega_{N2V}(i) \neq \varnothing$ are plotted. Obviously, the solution space has four different routes from node 1 to node

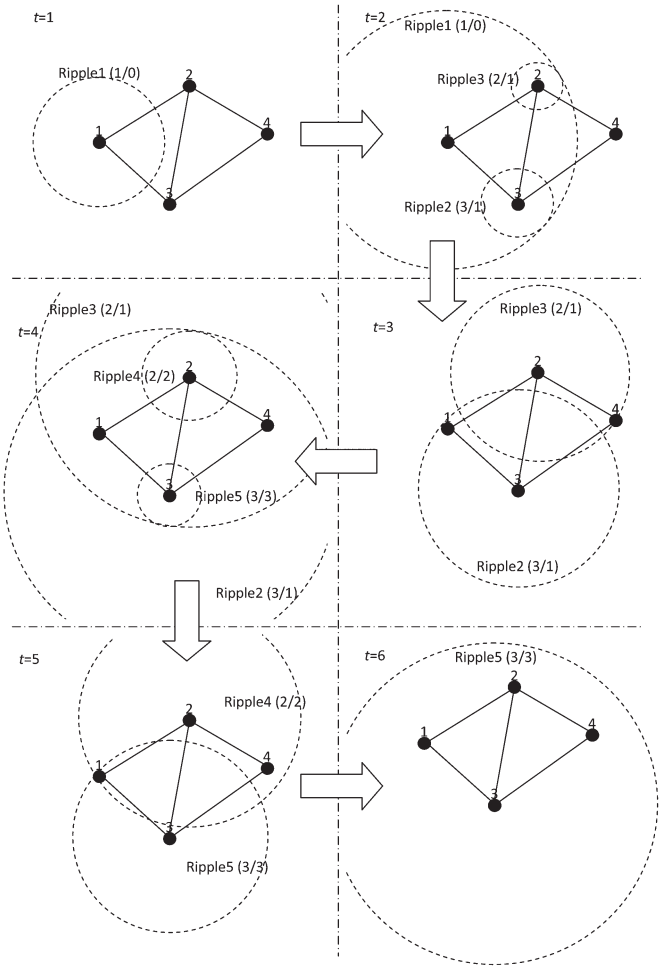


Fig. 4. Illustration of ripple relay race to identify the general kth shortest route.

4 in Fig. 4: $1 \to 2 \to 4$, $1 \to 3 \to 4$, $1 \to 2 \to 3 \to 4$, and $1 \to 3 \to 2 \to 4$. At time $t = 1$, only one ripple, i.e., ripple 1, starts from node 1. During time instant $t = 2$, ripple 1 reaches node 3 and then node 2 and triggers ripple 2 at node 3 and ripple 3 at node 2. Since the connection between node 1 and node 2 has a length different from that of the connection between node 1 and node 3, as a result, according to Step 3.2), ripple 2 and ripple 3 have different initial ripple radiuses, which are shown in the second step (i.e., $t = 2$) of Fig. 4. Ripple 3 reaches the destination first at time $t = 3$, identifying the first shortest route $1 \to 2 \to 4$. Ripple 2 reaches the destination at time $t = 4$, identifying the second shortest route $1 \to 3 \to 4$. During time instant $t = 4$, ripple 2 also triggers ripple 4 at node 2, and ripple 3 triggers ripple 5 at node 3. At time $t = 5$, ripple 4 reaches the destination, discovering the third shortest route $1 \to 3 \to 2 \to 4$. At time $t = 6$, ripple 5 reaches the destination, finding the worst route $1 \to 2 \to 3 \to 4$. Now, it should be clear that the aforementioned ripple-spreading algorithm is capable of calculating the general kth best route in terms of a given objective. By integrating this ripple-spreading algorithm into the first step of the methodology proposed in Section III, we can get an effective method to determine the complete exact Pareto front for the MOROP.

The ripple-spreading algorithm intuitively looks like it is able to guarantee optimality; however, we may raise the question of whether it is theoretically correct—i.e., is the kth ripple to reach the destination theoretically associated with the kth shortest route? It is straightforward to prove the theoretical optimality of the ripple-spreading algorithm. Assume that there are N_R ripples that reach the destination and ripple k travels along the kth shortest route to the destination. Since all ripples travel at the same speed, ripple k must reach the destination in the kth place. Otherwise, it is easy to mathematically prove that the route it travels along is not the kth shortest route, which is against the assumption.

In Step 2) of the method proposed in Section III, it is generally required to recalculate the $k+1$ best solutions if the previous k value does not fulfill (11) and (12). However, in the case of MOROP, this is not necessary, and only the $(k+1)$th shortest route needs to be calculated, because the k shortest routes remain the same as calculated before. With the ripple-spreading algorithm, after the k shortest routes are found, we simply freeze/pause (not terminate) the ripple-spreading process (simulation). Then, we check if we are lucky to have found the complete Pareto front. If not, we simply unfreeze/restart the ripple-spreading process (simulation) from the point where it is frozen/paused, in order to search for the $(k+1)$th shortest route.

The scalability of the method proposed in Section III mainly depends on that of the built-in optimizer for finding the k best solutions, e.g., in this case study, the computational efficiency of the ripple-spreading algorithm. Traditional route optimization algorithms employ certain artificial/heuristic search rules/logics (such as breadth-first search, depth-first search, and best first search) and often need to keep calculating the distance from the source to intermediate nodes and/or estimating the remaining distance from intermediate nodes to the destination. Existing algorithms for the kth shortest path problem usually employ similar searching techniques but have an even worse computational efficiency, because, in order to find the $(j+1)$th best solution, $1 \leq j < k$, they need to keep reconstructing route network based on the information of the first j best solutions [25]–[27]. Differently, the proposed ripple-spreading algorithm needs neither the aforementioned traditional search techniques nor the reconstruction of route network but simply mimicking the natural ripple-spreading phenomenon. Once a ripple reaches the destination in the kth place, the underlying optimization principle of the natural ripple-spreading phenomenon can automatically tell which route is the kth best route.

This gives the ripple-spreading algorithm a good scalability potential. The computational load of the ripple-spreading algorithm mainly comes from simulating the ripple relay race, and its complexity can be roughly estimated by the total number of all ripples ever generated/triggered. For the sake of simplicity, suppose that all connections are of unit length, each node connects to N_C other nodes, and the kth shortest route is composed of N_L nodes. Then, by the time when the ripple associated with the kth shortest route reaches the destination, there have been roughly $(N_C-1)^{N_L-2}$ ripples generated. As discussed in [29], the computational efficiency may be improved by adopting time-varying ripple-spreading speed as well as introducing multiple initial ripples. However, ripple-spreading algorithm is not the major focus of this paper. Readers may refer to [29] for more detailed theoretical analyses on the design and performance of general ripple-spreading algorithm. Please note that the method reported in [29] is only for finding the first best solution for single-objective optimization problems. This paper keeps the design of the ripple-spreading algorithm as simple as possible, in order to better highlight the proposed method for determining the complete Pareto front.

C. Simulation Results

Now, let us conduct some experimental analyses of the reported method by comparing it with population-based evolutionary algorithms and AOF-based methods for multiobjective optimization. We choose the most acknowledged NSGA-II in [14] as a representative of evolutionary algorithms. To apply NSGA-II to route optimization, we use the integer vector R, which is introduced in Section IV-A to record a candidate route, to construct chromosomes of GA. The techniques of mutation and elitism are used, but there is no crossover, because it may cause serious feasibility problems in route optimization. How to design effective and efficient crossover operators for combinatorial problems like route optimization has long been an interesting research topic [30], but it is not the concern of this study. When mutating a route, we randomly choose a pair of nodes which are not neighbors to each other in the old route but have direct connection in the route network, and then, we remove all the intermediate nodes between these two nodes to generate a mutated new route. The mutation probability is 0.3, i.e., 30% of the population will be randomly chosen for mutation. For each route, we calculate its g_1 value and g_2 value, and then, we use the crowding-distance computing and Pareto-ranking methods in [14] to rank all chromosomes in a generation. We always copy the best chromosome directly to the next generation (elitism). To retain population diversity, we renew by reinitializing the 30% of the population that have the lowest rankings. When initializing a chromosome, there is a one-third chance to choose the next node randomly, a one-third chance to choose the closest nodes, and another one-third chance to choose the farthest node. In the experiment, NSGA-II has a population size of 50 and evolves 200 generations. In the AOF-based method used in this experiment, denoted as AOF hereafter, we use the Dijkstra's algorithm, which is one of the best deterministic algorithms for single-objective route optimization [31]–[33], to minimize an AOF. To better understand the performance of the proposed deterministic method, we develop two versions with different algorithms for the k shortest paths problem as built-in optimizer. One version uses the well-known Yen's algorithm [25], and the other version employs the ripple-spreading algorithm described in Section IV-B, and they are denoted as DM-Yen and DM-RSA hereafter, respectively.

In the experiment, the network model in [34] is modified by allowing no link rewinding operation, in order to randomly generate route networks. First, we randomly disturb the locations of N_N nodes which are originally evenly distributed in a rectangular area defined by $[-1000\ 1000\ -1000\ 1000]$; we randomly choose a node to connect it to its closest neighboring

TABLE I
AVERAGE EXPERIMENTAL RESULTS

		NSGA-II		AOF	DM-Yen	DM-RS
		Average mean	Average SD			
N_N=25 ($N_{\mathrm{ETNR}}\approx$ 26244)	CT(sec.)	38.31	4.92	**0.41**	1.32	0.59
	N_{PPF}	4.06	0.94	2.75	**5.14**	**5.14**
	N_{SEE}	10000.00	(N/A)	50.00	**33.12**	**33.12**
	R_{FCPF}	0.29	(N/A)	0.05	**1.00**	**1.00**
N_N=36 ($N_{\mathrm{ETNR}}\approx$ 236196)	CT(sec.)	69.92	10.05	**0.78**	11.06	3.57
	N_{PPF}	4.38	1.24	3.21	**7.61**	**7.61**
	N_{SEE}	10000.00	(N/A)	72.00	**36.80**	**36.80**
	R_{FCPF}	0.24	(N/A)	0.04	**1.00**	**1.00**
N_N=49 ($N_{\mathrm{ETNR}}\approx$ 2125764)	CT(sec.)	115.72	19.77	**1.85**	94.31	29.76
	N_{PPF}	3.92	1.03	3.46	**9.12**	**9.12**
	N_{SEE}	10000.00	(N/A)	98.00	**54.86**	**54.86**
	R_{FCPF}	0.12	(N/A)	0.02	**1.00**	**1.00**

nodes and make sure every node has no more than four connections. Then, $D(i,j)$ is set as the physical straight line distance between node i and node j, and $C(i,j)$ is set according to (15). In the experiment, N_N has three values, i.e., 25, 36, and 49, and for each N_N, 100 route networks are generated.

For each route network, the NSGA-II, the AOF, and the two versions of the proposed method, i.e., DM-Yen and DM-RSA, are applied to search the Pareto front. In each test (i.e., for each route network), 20 runs of NSGA-II are conducted in order to average the influence of the stochastic nature of NSGA-II. In each test, AOF executes the Dijkstra's algorithm for $N_W = 2N_N$ times and each time one of N_W α values evenly distributed within [0 1] is used to integrate g_1 and g_2 into a single AOF. In each test, DM-Yen and DM-RSA are both executed twice, one time with g_1 as objective, and the other time with g_2 as objective. All methods are coded, and all tests are conducted in a Matlab environment on a personal computer with a 2.6-GHz CPU, 2-GB memory, and Windows XP operating system.

For each N_N, the average results of 100 tests are given in Table I, where N_{ETNR} is the estimated total number of candidate routes from the source to the destination, CT stands for computational time, N_{PPF} means the number of Pareto points found by a method, N_{SEE} is the number of solutions ever explored by a method, and R_{FCPF} is the rate of a method for finding the complete Pareto front, i.e., in how many of the 100 route networks of a given N_N that the method has found the complete Pareto front. N_{ETNR} is estimated as follows. In every test, the source is always the node at the left bottom, and the destination is the node at the right top. Therefore, a route with minimal nodes is roughly composed of $\sqrt{2N_N}$ nodes, while a long route could be composed of all the N_N nodes. Here, we conservatively assume that an average route is composed of $2\sqrt{N_N}$ nodes. Since each node has four connections, the source have four choices. For any node other than the source and the destination in a route, it may have at most three choices as its leading node is fixed. Therefore, given that an average route has $2\sqrt{N_N}$ nodes, the total number of routes can be roughly estimated as $N_{\mathrm{ETNR}} = 4 \times 3^{2\sqrt{N_N}-2}$. Since NSGA-II is a stochastic algorithm by nature, for a given route network, we have a mean value and standard deviation (SD) for CT and N_{PPF} based on 20 runs. Then, based on the 100 route networks of a given N_N, we have an average mean value and average SD in Table I. For AOF, DM-Yen, and DM-RSA, there is no such average mean or SD data, because they are all deterministic methods. From Table I, one can have the following observations.

1) All outputs of the AOF-based method are Pareto points, but the least Pareto points are found by the AOF-based method. The NSGA-II finds more Pareto points than the AOF-based method. The two versions of the proposed method, i.e., DM-Yen and DM-RSA, find the most Pareto points (on average, about 100% more than either the AOF-based method or the NSGA-II). In the experiment, there are always k_1 and k_2 found to satisfy (13) and (14), which means that the proposed method always finds the complete exact Pareto front.
2) Regarding CT, the AOF-based method is the most efficient, the proposed method is the second most time efficient, and the NSGA-II is the slowest method. However, it should be noted that the CT consumed by the proposed method sours up significantly as N_N increases. Therefore, the scalability of the proposed method needs to be improved. The NSGA-II is also confronted with the same scalability problem.
3) When comparing the number of solutions ever explored by a method, one can see that, due to its stochastic nature, the NSGA-II has to search an enormous number of candidate solutions to find a few Pareto points. The AOF-based method uses many different α values to construct AOFs, but most AOFs lead to a few same Pareto points. The proposed method searches the fewest candidate solutions to identify the complete exact Pareto front. If one has a look at the value of $N_{\mathrm{PPF}}/N_{\mathrm{SEE}}$, it is further clear that the proposed method has the best searching efficiency, while the NSGA-II has the worst.
4) When we look at R_{FCPF}, i.e., the rate of a method to find the complete Pareto front, it is clear that, in most times, neither NSGA-II nor AOF can find the complete Pareto front. AOF has the smallest R_{FCPF}, because only when the Pareto front is globally convex does AOF stand a chance to find it. The adopted Pareto-ranking technique gives NSGA-II a better chance to find the complete Pareto front, but this is not always guaranteed due to its stochastic nature. As the Pareto front becomes more complicated in general with increasing N_N, R_{FCPF} shows that it gets more difficult for NSGA-II to find the complete Pareto front. Only the proposed method, either DM-Yen or DM-RSA, can always guarantee to find the complete Pareto front, i.e., $R_{\mathrm{FCPF}} = 1$.
5) From Table I, one can see that the two versions of the proposed method, i.e., DM-Yen and DM-RSA, have exactly the same performance except different CTs. This proves that, as long as the proposed method has a built-in optimizer that is capable of finding the k best solutions in terms of each of the single-objective functions, the finding of the compete Pareto front is then guaranteed. Different built-in optimizers only lead to different computational efficiencies. The CT data in Table I imply that the reported ripple-spreading algorithm is more

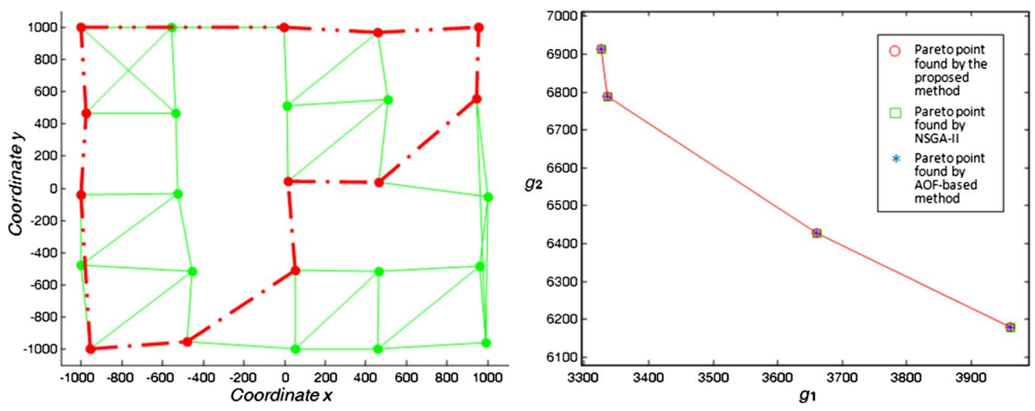


Fig. 5. Example 1 of solutions by different methods ($N_N = 25$).

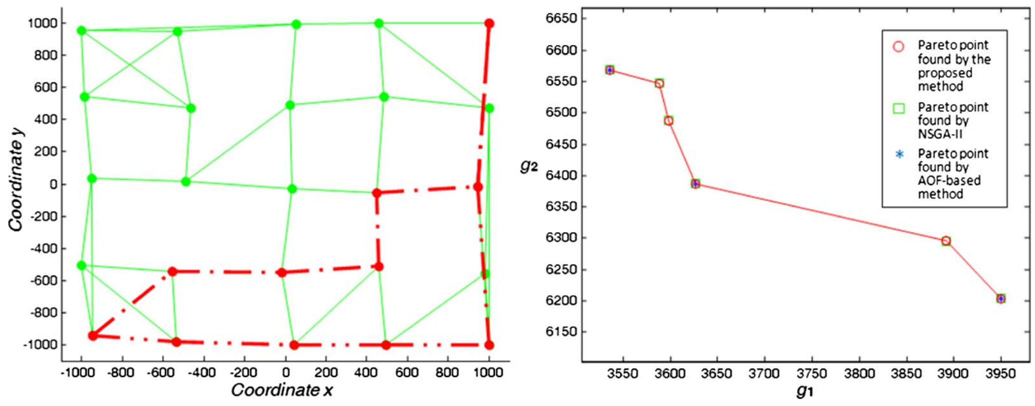


Fig. 6. Example 2 of solutions by different methods ($N_N = 25$).

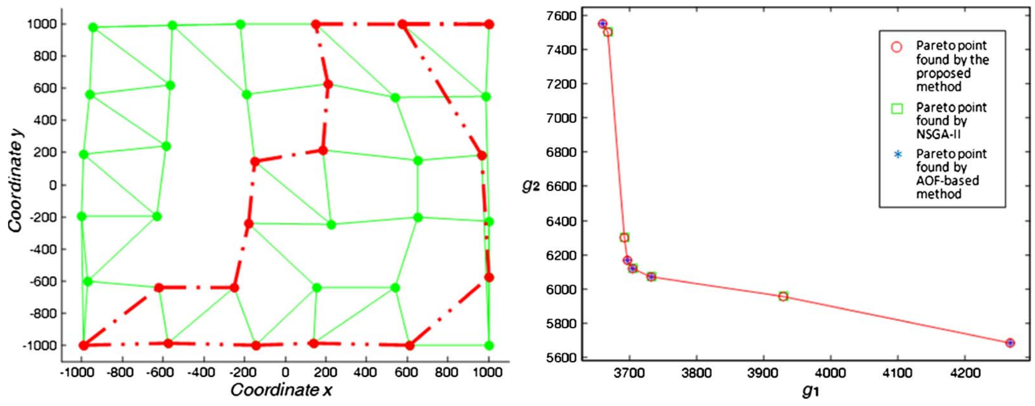


Fig. 7. Example 3 of solutions by different methods ($N_N = 36$).

computationally efficient than the Yen's algorithm in [25] for the k shortest paths problem, largely because, as already discussed in Section IV-B, the Yen's algorithm has to keep reconstructing route networks, while the ripple-spreading algorithm is completely free of this computational load.

Figs. 5–10 show some examples of optimal routes (in the left-side subgraph) and identified Pareto fronts (in the right-side subgraph). The dash-and-dot line in each of Figs. 5–10 is the first best route in terms of g_1, which corresponds to the most left Pareto point in the associated right-side subgraph. The dash-and-double-dot line in each of Figs. 5–10 is the first best route in terms of g_2, which leads to the lowest Pareto point in the associated right-side subgraph. As defined in Section IV-A, the objective g_1 is actually physical distance, while the artificial cost g_2 is related to the inverse of the physical distance of the connections. As a result of these two conflicting objectives, the best routes in terms of g_2 are much longer than those in terms of g_1, as shown in the left-side subgraphs of Figs. 5–10. However, it should be emphasized that, according to the definition given in (15) and (18), the best route in terms of the artificial cost is not necessarily the physically longest route. Each of the right-side subgraphs of Figs. 5–10 plots the Pareto front of the route network given in the corresponding left-side subgraph. A Pareto front identified by the proposed method is plotted as a curve connecting all real Pareto points. Such a curve of Pareto front can

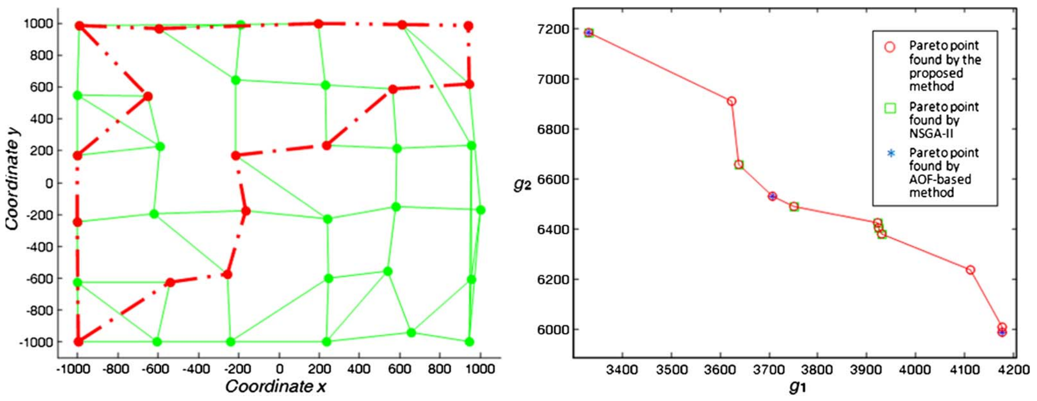


Fig. 8. Example 4 of solutions by different methods $(N_N = 36)$.

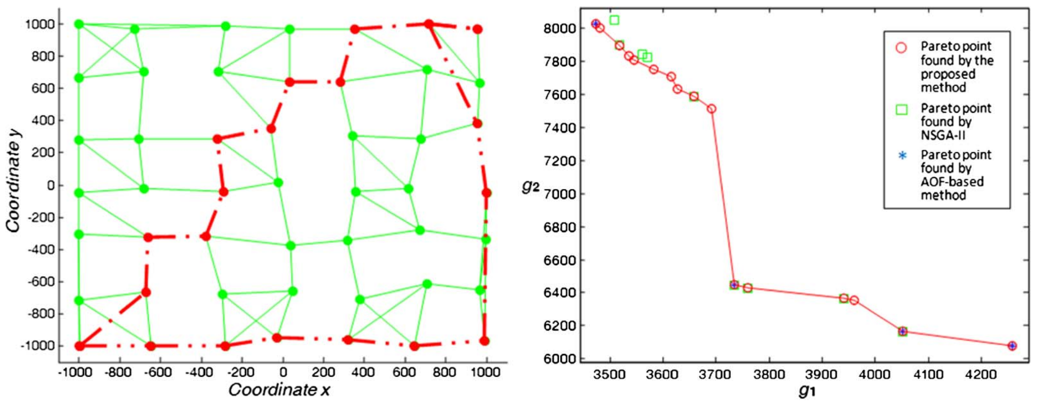


Fig. 9. Example 5 of solutions by different methods $(N_N = 49)$.

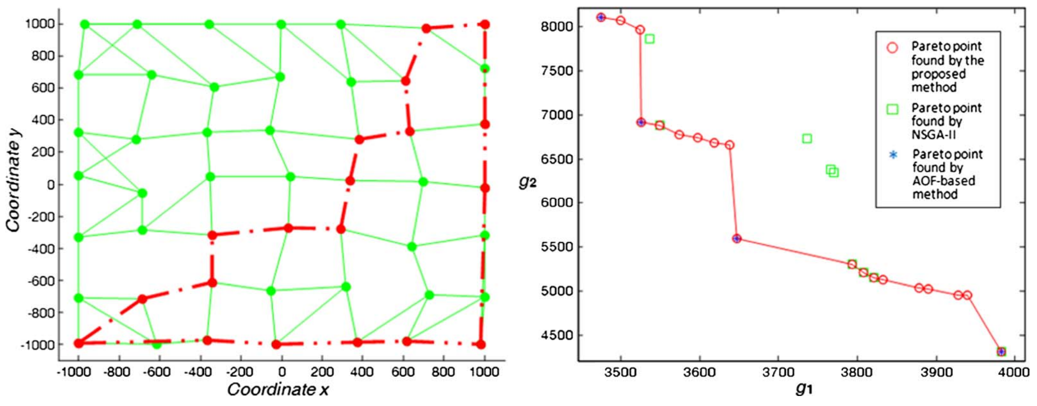


Fig. 10. Example 6 of solutions by different methods $(N_N = 49)$.

intuitively illustrate the definition of Pareto optimality: When we move from one Pareto point to another Pareto point, it always happens that one objective increases while the other decreases.

From the examples in Figs. 5–10, one may have more insights into the capabilities of different methods. If the Pareto front is simple, such as that in Fig. 5, all three methods can find the complete exact Pareto front. The AOF-based method is basically good at finding Pareto points which form a convex envelope covering the complete exact Pareto front but cannot find nonconvex Pareto points. For example, the Pareto front in Fig. 6 is also simple, and the NSGA-II can find the complete exact Pareto front as the proposed method does, but the AOF-based method fails to do so because there are three nonconvex Pareto points. The NSGA-II is also effective in finding or approximating Pareto points if the Pareto front is a relative smooth curve, such as that shown in Figs. 7 and 8. However, if the Pareto front is in a complex zigzag shape, e.g., Figs. 9 and 10, then the NSGA-II often struggles to find Pareto points.

In Figs. 5–10, the solutions found by either the NSGA-II or the AOF-based method never provide any Pareto point that is not included by the Pareto front identified by the proposed method. We checked the results of every test and found that this is always the case in all tests. This may intuitively demonstrate that the proposed method is capable of finding the complete exact Pareto front, although the theoretical proof given in Section II should already be sufficient.

It should be noted that the techniques used in the NSGA-II and the AOF-based method are rather basic and there are many new developments reported regarding how to improve the performances of the NSGA-II and the AOF-based method. Therefore, further comparison with such new developments is worthy of investigation, particularly in specific real-world applications. However, this is not really important in this study, because the message here is as follows: Either NSGA-II or the AOF-based method can only, in theory, generally provide an approximation of the true Pareto front, even though their performances might be improved by certain new techniques in a particular application; in contrast, the proposed method is theoretically capable of finding the true Pareto front.

V. Conclusion and Future Work

Multiobjective optimization aims at finding the Pareto front. However, no existing methods, neither population-based evolutionary algorithms nor AOF-based methods, can guarantee the determination of the complete exact Pareto front. This paper has proposed for the first time a methodology that can theoretically guarantee finding the complete exact Pareto front for discrete problems. To this end, some new theoretical results for multiobjective optimization are developed. The crucial part of the proposed methodology is to design an optimization method capable of calculating the general kth best solution in terms of a given single objective. To demonstrate the practicability of the proposed methodology, a novel algorithm inspired by the natural ripple-spreading phenomenon is developed to calculate the kth shortest route for single-objective route optimization. With this ripple-spreading algorithm, the proposed methodology can then be applied to identify the complete exact Pareto front for multiobjective route optimization. The effectiveness and efficiency of the proposed method are illustrated in a comparative experiment.

It should be noted that very little work has so far been done regarding how to determine the complete exact Pareto front for multiobjective optimization. This paper attempts to shed some light in this direction theoretically and experimentally. Further extensive efforts are needed to make more progress. For example, it will be important to conduct more theoretical and experimental analyses on searching efficiency, extend the reported ripple-spreading algorithm to other problems besides route optimization, develop other algorithms capable of calculating the general kth best solution, improve the scalability of the proposed methodology, and investigate the possibility of calculating the complete Pareto front for continuous problems.

Acknowledgment

The authors would like to thank the anonymous reviewers, whose comments were of great help to improve the quality of this paper.

References

[1] N. Barr, *Economics of the Welfare State*. New York: Oxford Univ. Press, 2004.

[2] R. E. Steuer, *Multiple Criteria Optimization: Theory, Computations, and Application*. Hoboken, NJ: Wiley, 1986.

[3] Y. Sawaragi, H. Nakayama, and T. Tanino, *Theory of Multiobjective Optimization (Vol. 176 of Mathematics in Science and Engineering)*. Orlando, FL: Academic, 1985.

[4] L. Raamesh and G. V. Uma, "Data mining based optimization of test cases to enhance reliability of the testing," in *Advances in Computing and Information Technology*, D. W. Wyld, M. Wozniak, N. Chaki, N. Meghanathan, and D. Nagamalai, Eds. New York: Springer-Verlag, 2011, pp. 89–98.

[5] I. Das and J. E. Dennis, "Normal-boundary intersection: A new method for generating the Pareto surface in nonlinear multicriteria optimization problems," *SIAM J. Optim.*, vol. 8, no. 3, pp. 631–657, Aug. 1998.

[6] I. Das and J. Dennis, "Normal-Boundary Intersection: An Alternate Method for Generating Pareto Optimal Points in Multicriteria Optimization Problems," NASA Langley Res. Centr., Hampton, VA, NASA Contractor Rep. 201616, ICASE Rep. 96-62, 1996.

[7] A. Messac, A. Ismail-Yahaya, and C. A. Mattson, "The normalized normal constraint method for generating the Pareto front," *Struct. Multidiscipl. Optim.*, vol. 25, no. 2, pp. 86–98, Jul. 2003.

[8] A. Messac and C. A. Mattson, "Normal constraint method with guarantee of even representation of complete Pareto front," *AIAA J.*, vol. 42, no. 10, pp. 2101–2111, Oct. 2004.

[9] T. Erfani and S. V. Utyuzhnikov, "Directed search domain: A method for even generation of Pareto front in multiobjective optimization," *Eng. Optim.*, vol. 43, no. 5, pp. 467–484, May 2011.

[10] A. Konak, D. W. Coit, and A. E. Smith, "Multi-objective optimization using genetic algorithms: A tutorial," *Reliab. Eng. Syst. Safety*, vol. 91, no. 9, pp. 992–1007, Sep. 2006.

[11] M. Delgado, M. P. Cuellar, and M. C. Pegalajar, "Multiobjective hybrid optimization and training of recurrent neural networks," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 38, no. 2, pp. 381–403, Apr. 2008.

[12] D. F. Jones, S. K. Mirrazavi, and M. Tamiz, "Multiobjective metaheuristics: An overview of the current state-of-the-art," *Eur. J. Oper. Res.*, vol. 137, no. 1, pp. 1–9, Feb. 2002.

[13] N. Srinivas and K. Deb, "Multiobjective optimization using nondominated sorting in genetic algorithms," *Evol. Comput.*, vol. 2, no. 3, pp. 221–248, 1994.

[14] K. Deb, "A fast and elitist multiobjective genetic algorithm: NSGA-II," *IEEE Trans. Evol. Comput.*, vol. 6, no. 2, pp. 182–197, Apr. 2002.

[15] W. F. Leong and G. G. Yen, "PSO-based multiobjective optimization with dynamic population size and adaptive local archives," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 38, no. 5, pp. 1270–1293, Oct. 2008.

[16] J. D. Knowles and D. W. Corne, "Approximating the nondominated front using the Pareto archived evolution strategy," *Evol. Comput.*, vol. 8, no. 2, pp. 149–172, Jun. 2000.

[17] S. Bandyopadhyay, S. K. Pal, and B. Aruna, "Multiobjective GAs, quantitative indices, and pattern classification," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 34, no. 5, pp. 2088–2099, Oct. 2004.

[18] X. J. Lei and Z. K. Shi, "Overview of multi-objective optimization methods," *J. Syst. Eng. Electron.*, vol. 15, no. 2, pp. 142–146, Jun. 2004.

[19] K. Deb, *Multi-Objective Optimization Using Evolutionary Algorithms*. New York: Wiley, 2001.

[20] D. A. Van Veldhuizen and G. B. Lamont, "Multiobjective evolutionary algorithms: Analyzing the state-of-the-art," *Evol. Comput.*, vol. 8, no. 2, pp. 125–147, Jun. 2000.

[21] X. F. Zou, Y. Chen, M. Z. Liu, and L. S. Kang, "A new evolutionary algorithm for solving many-objective optimization problems," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 38, no. 5, pp. 1402–1412, Oct. 2008.

[22] E. Zitzler, K. Deb, and L. Thiele, "Comparison of multiobjective evolutionary algorithms: Empirical results," *Evol. Comput.*, vol. 8, no. 2, pp. 173–195, Jun. 2000.

[23] J. Figueira, S. Greco, and M. Ehrgottc, *Multiple Criteria Decision Analysis: State of the Art Surveys*. New York: Springer-Verlag, 2005.

[24] D. Craft, T. Halabi, H. Shih, and T. Bortfeld, "Approximating convex Pareto surfaces in multiobjective radiotherapy planning," *Med. Phys.*, vol. 33, no. 9, pp. 3399–3407, Sep. 2006.

[25] J. Y. Yen, "Finding the k shortest loopless paths in a network," *Manage. Sci.*, vol. 17, no. 11, pp. 712–716, Jul. 1971.

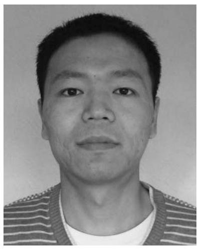
[26] D. Eppstein, "Finding the k shortest paths," *SIAM J. Optim.*, vol. 28, no. 2, pp. 652–673, Apr. 1998.

[27] H. Aljazzar and S. Leue, "K^*: A heuristic search algorithm for finding the k shortest paths," *Artif. Intell.*, vol. 175, no. 18, pp. 2129–2154, Dec. 2011.

[28] S. L. C. Pun-Cheng, "An interactive Web-based public transport enquiry system with real-time optimal route computation," *IEEE Trans. Intell. Transp. Syst.*, vol. 13, no. 2, pp. 983–988, Jun. 2012.

[29] X. B. Hu, M. Wang, M. S. Leeson, E. L. Hines, and E. Di Paolo, "A ripple-spreading algorithm for route optimization," in *Proc. IEEE SSCI*, Singapore, Apr. 15–19, 2013.

[30] S. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach,* 3rd ed. Englewood Cliffs, NJ: Prentice-Hall, 2010.
[31] E. W. Dijkstra, "A note on two problems in connexion with graph," *Numer. Math.*, vol. 1, no. 1, pp. 269–271, Dec. 1959.
[32] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, "Section 24.3: Dijkstra's algorithm," in *Introduction to Algorithms*, 2nd ed. Cambridge, MA: MIT Press, 2001, pp. 595–601.
[33] M. Sniedovich, "Section 15.5: Dijkstra's algorithm," in *Dynamic Programming: Foundations and Principles*. New York: Taylor & Francis, 2010, pp. 441–448.
[34] D. J. Watts and S. H. Strogatz, "Collective dynamics of 'small-world' networks," *Nature*, vol. 393, no. 6684, pp. 440–442, Jun. 1998.



Xiao-Bing Hu received the B.S. degree in aviation electronic engineering from the Civil Aviation Institute of China, Tianjin, China, in 1998, the M.S. degree in automatic control engineering from Nanjing University of Aeronautics and Astronautics, Nanjing, China, in 2001, and the Ph.D. degree in aeronautical and automotive engineering from Loughborough University, Loughborough, U.K., in 2005.

He is currently an Experienced Marie Curie Fellow with the State Key Laboratory of Earth Surface Processes and Resource Ecology, Beijing Normal University, Beijing, China, and also with the School of Engineering, University of Warwick, Coventry, U.K. His major fields of research include integrated risk governance, complex networks, artificial intelligence, and air traffic management.

Ming Wang received the B.S. degree in civil engineering from Tsinghua University, Beijing, China, in 2000 and the M.S. and Ph.D. degrees in structural engineering from the University of Maryland, College Park, in 2005 and 2006, respectively.

He is currently an Associate Professor with the State Key Laboratory of Earth Surface Processes and Resource Ecology, Beijing Normal University, Beijing. His major fields of research include risk modeling and simulation, integrated risk governance, and complex networks.

Ezequiel Di Paolo received the M.Sc. degree in nuclear engineering from the Instituto Balseiro, Bariloche, Argentina, and the Ph.D. degree in computer science and artificial intelligence from the University of Sussex, Brighton, U.K.

He is currently a Research Professor with Ikerbasque, Basque Science Foundation, Centre for Research on Life, Mind and Society, University of the Basque Country, San Sebastian, Spain. He is also a Visiting Senior Researcher with the Centre for Computational Neuroscience and Robotics, Department of Informatics, University of Sussex. His research interests include adaptive behavior in natural and artificial systems, biological modeling, evolutionary robotics, and enactive cognitive science.