Finding all Pareto optimal paths by simulating ripple relay race in multi-objective networks

Xiao-Bing Hu [a,b,c], Sheng-Hao Gu [a], Chi Zhang [b,*], Gong-Peng Zhang [b], Ming-Kong Zhang [d], Mark S. Leeson [c]

[a] *CAUC-ENAC Joint Research Center of Applied Mathematics for Air Traffic Management, College of Safety Science and Engineering, Civil Aviation University of China, Tianjin, 300300, China*
[b] *Collaborative Innovation Center of eTourism, Beijing Union University, Beijing, China*
[c] *School of Engineering, University of Warwick, Coventry, CV4 7AL, UK*
[d] *Institute of Geographical Sciences, Henan Academy of Sciences, Zhengzhou, Henan, 450000, China*

Keywords:
Path optimization
Ripple-spreading algorithm
Multi-objective optimization
Complete Pareto front

A B S T R A C T

This paper proposes a novel nature-inspired method, so-called ripple-spreading algorithm (RSA) for multi-objective path optimization problem (MOPOP). Unlike most existing methods mainly capable of finding partial or approximated Pareto front, this paper focuses on calculating the complete Pareto front. This is achieved by taking advantage of the optimality principle in natural ripple-spreading phenomenon. Basically, the proposed RSA carries out a one-off ripple relay race in the route network, and then the complete Pareto front will be identified with guaranteed optimality by backtracking those Pareto non-dominated ripples (PNDRs) which reached the destination node. Theoretical analyses and comprehensive experiments show that all complete Pareto fronts of a one-to-all MOPOP can also be found in just a single run of ripple relay race, and the reported method can be further extended to calculate all Pareto optimal paths in dynamical networks, which have rarely been touched by existing MOPOP methods. Since many real-world application problems can be converted into MOPOP, the reported method has a great potential of applications.

INTRODUCTION

Inspirations from the nature have successful helped researchers to develop many effective multi-agent-based evolutionary computation methods [1],[2]. Ripple-spreading models and algorithms are relatively new in the family of nature-inspired multi-agent-based methods, but they have shown potentials in studying some problems (e.g., sequencing aircraft [3], modeling complex networks [4], simulating epidemic dynamics [5], finding the first shortest path in static [6] and dynamical environments [7], calculating the *k* shortest paths without [8] and with time-windows [9]), and they suggest a possibility of new stream of optimization methods, distinguished from both classical deterministic method stream (e.g., A* algorithm [10], Dijkstra's algorithm [11] and label correcting algorithm [12]) and stochastic method stream (e.g., genetic algorithm [13], particle swarm optimization [14], ant colony optimization [15] and firefly optimization [16]). Usually, deterministic methods have optimality guarantee but are not flexible in modification, while stochastic methods often output non-optimal solutions but are easy to modify for various applications. Interestingly, as a multi-agent-based method, ripple-spreading algorithm (RSA) is optimal, flexible and fast, and it exhibits merits of both method streams while avoids their demerits [6]-[7]. This paper aims to make RSA a more mature and more distinguishing method, which will further enrich the family of nature-inspired methods. To this end, the RSA reported in this paper, different from all previously published RSAs, which are single-objective optimization methods, is developed for the first time as a true multi-objective optimization method. Furthermore, this paper extends RSA to find complete Pareto front with theoretical optimality guarantee for dynamical multi-objective path optimization problem, which has never been achieved by any existing methods except [17].

Many theoretical and application researches often involve path optimization [1],[2],[18]. In reality, path optimization often has to consider some different optimization objectives simultaneously, for example, path length, traveling time, fuel consumption, service charge, and journey risk due to exposure to adverse events, so comes MOPOP. In general, MOPOP has not just a single optimal path, but a set of optimal paths, whose projections in the objective space compose so-called Pareto front. Therefore, finding Pareto front associated with all Pareto optimal paths is the ultimate goal of resolving a MOPOP. However, calculating complete Pareto front is not an easy task for multi-objective optimization problem (MOOP) [19],[20]. In particular, it has even rarely been studied to identify all Pareto optimal paths in time-varying route networks, despite of the fact that dynamical MOPOP is the most realistic when compared with other kinds of path optimization problems. Therefore, resolving dynamical MOPOP will be a main concern of this paper.

Pareto front is a core concept for resolving MOOPs, and it is associated with a set of Pareto optimal solutions [20],[21]. However, most existing methods for MOOPs focus on finding just a partial or approximated Pareto front, and there seriously lacks effective method to guarantee the complete Pareto front. Basically, there are four kinds of MOOP methods, and they are based on aggregate objective function (AOF), constrained objective function (COF), Pareto-compliant ranking (PCR), and decomposition evolutionary algorithms (MOEA/D). An AOF method actually resolves a single-objective optimization problem (SOOP) which combines all of the original objectives in an MOOP to construct a single aggregate objective function [22]. Various techniques for combining original objectives were reported, for instance, linear summing based on weights was the most popular [23], while purpose-designed nonlinear combining functions were sometimes used, such as weighted exponential sum [24], weighted min-max method [25], and weighted product method [26]. A COF method optimizes only one single objective, and treats all other objectives as extra constraints [27]. Different constraint strategies, such as normal boundary intersection [28] and normalized normal constraint [29] were introduced, and how to achieve an even representation of Pareto front was often a major concern of COF methods [30, 31]. A PCR method ranks and evolves a population of candidate solutions in order to find multiple Pareto non-dominated solutions. Pareto-compliant ranking was first introduced into genetic algorithm [32], and then improved by elitist technique [33], reference-point based technique [34], dynamic population size and adaptive local archives [35], Pareto archived evolution strategy [36] and objective reduction method [37]. The MOEA/D methods are also population-based, but there is a big difference from the PCR methods. In PCR, each individual of a population represents a single solution to the MOOP, while in MOEA/D, thanks to problem-decomposition technologies, each individual of a population represents a family of solutions with similar features to the MOOP, and this often makes MOEA/D more efficient than PCR [38],[39].

There is a common problem faced by existing MOOP methods, i.e., they usually just find partial or approximated Pareto fronts, and have no guarantee of complete Pareto front [27]. There are some benchmark test problems with full information about complete Pareto front of continuous MOOPs [40], but few results are available on the quality of approximated Pareto front for discrete MOOPs [20],[41]. As for dynamical MOOP, AOF and COF methods are rarely used, and mainly PCR and MOEA/D methods can approximate some Pareto optimal solutions [42],[43], but complete Pareto front is never mentioned for dynamical MOOP.

Reference [44] reported a deterministic method that can, theoretically and practically, guarantee the finding of complete Pareto front for discrete MOOPs. The basic idea is: find the *k* best single-objective optimal solutions in term of each of the N_{Obj} objectives in an MOOP, then, all Pareto non-dominated solutions in those $N_{\mathrm{Obj}} \times k$ single-objective optimal solutions will reveal the complete Pareto front of the original MOOP. To find the *k* best single-objective optimal solutions, the method of [44] employs a single-objective ripple-spreading algorithm (RSA).

For multi-objective path optimization problem (MOPOP), some researchers have already proven that it is possible to find the complete Pareto front by, for instance, a label-correcting method based on a technique of extending one label at a certain node according to all arcs out of that node [45], a two-phase method based on parametric network simplex algorithm and *k* shortest paths algorithm [46], a label-setting algorithm based on a dimensionality reduction technique [47], a Dijkstra-like method based on extreme supported solutions [48], and an exact recursive method based on implicit enumeration that aggressively prunes dominated solutions [49]. Generally, these methods can be viewed as extensions of single-objective Dijkstra's algorithm and A* algorithm, and they mainly studied static bi-objective problems. As shown in [6] and [7], RSA, as a newly reported, nature-inspired, multi-agent-based but deterministic algorithm, exhibits some advantages against traditional Dijkstra's algorithm and A* algorithm, particularly when it comes to resolve some complicated path optimization problems. For example, when route networks with time windows or dynamical routing environments are concerned, RSA has demonstrated a good modification flexibility and computational efficiency [7],[8]. This paper, with finding complete Pareto front as the focus, is first interested in exploring the possibility of applying RSA to target MOPOP with static route networks, and then, it goes on to extend to MOPOP with dynamical route networks, which are more complicated and have rarely been touched in those MOPOP literatures such as [45]-[49]. Actually, RSA is the only method that has been illustrated so far to find complete Pareto front for dynamical MOPOP [17]. In terms of studying dynamical MOPOP, this paper extends the preliminary results of [17] from one-to-one problem to one-to-all problem with more experimental results.

It should be emphasized that the basic idea of the method in [44] for MOOPs has nothing to do with the ripple-spreading optimization principle as identified in [6], and the RSA used in [44] itself has nothing to do with MOOPs. In other words, the method of [44] needs an algorithm to calculate the *k* best single-objective optimal solutions, and the RSA happens to have such a capability, i.e., the RSA used in [44] is itself just for single-objective optimization problems (SOOPs). For an MOOP with N_{Obj} objectives, the RSA in [44] needs to be run for at least N_{Obj} times, each time for one of the N_{Obj} objectives. Actually, the RSA employed by the method of [44] can be replaced by any method that is capable of calculating the *k* best single-objective optimal solutions, and after the replacement, the method of [44] can still resolve MOOPs. In other words, the MOOP method of [44] does not rely on the ripple-spreading optimization principle of [6].

Differently, the MOOP method proposed in this paper thoroughly relies on the ripple-spreading optimization principle of [6]. The RSA used in [44] is a single-objective method, while the RSA reported in this paper is truly a multi-objective method. To find the complete Pareto front for MOPOP, the RSA in [44] has to run for at least N_{Obj} times, while the RSA in this paper only needs to be run for just once. Actually, a single run of the new RSA can even guarantee to find all complete Pareto fronts of one-to-all MOPOP. To apply the method of [44] to a one-to-all MOPOP with N_{N} nodes, then one needs to run the method of [44] for (N_{N}-1) times, which means to run the RSA in [44] for at least $(N_{\mathrm{N}}\text{-}1) \times N_{\mathrm{Obj}}$ times. Differently, the new RSA proposed in this paper only needs to be run once to tackle the same one-to-all MOPOP. Besides, the RSA reported in this paper exhibits a unique capability far beyond the method of [44] as well as most other existing methods for MOOPs, and that is to find complete Pareto front with optimality guarantee in dynamical routing environments. Actually, complete Pareto front of MOPOP in dynamical route networks has seldom been discussed elsewhere.

There are 5 parts in the remainder of this paper, in order to present a methodological tree of RSA for various MOPOPs, i.e., static one-to-one MOPOP, static one-to-all MOPOP, dynamical one-to-one MOPOP, and dynamical one-to-all MOPOP. Mathematical models are described in Section 2, the new RSA is developed in Section 3 and analyzed theoretically in Section 4, comprehensive experimental data are given in Section 5, and conclusions and future works are discussed in Section 6.

Problem Description of Static Multi-Objective Path Optimization Problems

Assume a static route network *G*(*V*,*E*) is composed of node set *V* and link set *E*. *V* has N_{N} different nodes including the source and the destination, and *E* has N_{L} links between nodes. This route network can be recorded as an $N_{\mathrm{N}} \times N_{\mathrm{N}}$ adjacent matrix *A*. The matrix entry $A(i,j)=1$, $i=1,\ldots,N_{\mathrm{N}}$ and $j=1,\ldots,N_{\mathrm{N}}$, defines a link from node *i* to node *j*. Otherwise, $A(i,j)=0$ means no link. Assume $A(i,i)=0$, i.e., no self-connecting link is

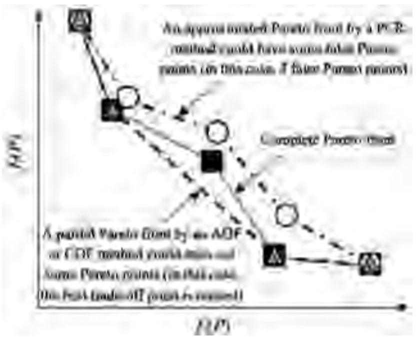


Fig. 1. Examples of partial, approximated and complete Pareto fronts.

allowed in this study. There are N_{Obj} costs, i.e., $C_k(i,j)$, $k=1,\ldots,N_{\text{Obj}}$, associated with each link $A(i,j)$, and $C_k(i,j)$ will be used to calculate the kth objective of a path. Here, the route network $G(V,E)$ is called static, because both $A(i,j)$ and $C_k(i,j)$ are fixed and will not change during optimization.

In a one-to-one static MOPOP, a pair of source and destination nodes, denoted as $[S,D]$, are specified, and then the goal is to find Pareto optimal paths from S to D. Suppose a candidate path is recorded as an integer vector whose element $P(i)=j$ means node j is the ith node in the path, $i=1,\ldots,N_{\text{P}}$, and $j=1,\ldots,N_{\text{N}}$, where N_{P} tells how many nodes, including the source and the destination, are included in the path. Obviously, $P(1)$ is the source and $P(N_{\text{P}})$ is the destination, i.e., $P(1)=S$, and $P(N_{\text{P}})=D$. Since no loop is allowed in this study, no node can appear in a path for more than once, i.e.,

$$P(i) \neq P(j), \text{ if } i \neq j, i = 1, \ldots, N_P, \text{ and } j = 1, \ldots, N_P. \tag{1}$$

For a given path P, its total cost from the source to the destination can be calculated in terms of each objective

$$f_k(P) = \sum_{i=1}^{N_P-1} C_k(P(i), P(i+1)), k = 1, \ldots, N_{\text{Obj}}, \tag{2}$$

where f_k is the kth objective of an MOPOP.

Then, a general mathematical formulation of one-to-one static MOPOP can be given as following:

$$\min_P \left[f_1(P), f_2(P), \ldots, f_{N_{Obj}}(P) \right]^T \tag{3}$$

subject to Eq.(1)

$$P \in \Omega_P, \tag{4}$$

where Ω_{P} is the set of all possible paths connecting S and D.

A Pareto-optimal path P^* to the above problem is such that there exists no P that makes

$$f_i(P) \leq f_i(P^*), \text{ for all } i = 1, .., N_{\text{Obj}}, \tag{5}$$

$$f_j(P) < f_j(P^*), \text{ for at least one } j \in \left[1, .., N_{\text{Obj}}\right]. \tag{6}$$

The N_{Obj} objective values of such a P^*, i.e., $[f_1(P^*), f_2(P^*), \ldots, f_{N_{Obj}}(P^*)]$, determine a Pareto point in the objective space. In general, there are more than one Pareto-optimal path for the above MOPOP, and their associated Pareto points in the objective space construct the Pareto front.

Many methods have been reported to find a partial or approximated Pareto front for the above MOPOP. Fig. 1 intuitively illustrates what a partial, an approximated and a complete Pareto front might look like in a bi-objective path optimization problem. Fig. 1 clearly implies that, when compared with the complete Pareto front, a partial or approximated Pareto front provides insufficient or even incorrect information to decision-makers.

Ripple-Spreading Algorithm for Static Problems

This section describes how to extend single-objective RSA to resolve multi-objective path optimization problems (not only one-to-one problem, but also one-to-all problem) in static route networks.

3.1. The basic idea of ripple-spreading algorithm

Reference [6], by mimicking the natural ripple-spreading phenomenon, developed ripple-spreading algorithm (RSA) for some single-objective path optimization problems (POPs), and proved the optimality of RSA based on the ripple-spreading optimization principle, i.e., a ripple always reaches the closest spatial point first because of its identical spreading speed in all directions. This very simple principle can be easily applied to accomplish path optimization problems (POPs) effectively, and several ripple-spreading algorithms (RSAs) were developed in [6] to resolve one-to-one, one-to-all, many-to-many the 1st shortest path problems, and one-to-one the k shortest paths problem. There are usually two kinds of path optimization methods, deterministic and stochastic. Deterministic methods are centralized, top-down, logic-rule-based search algorithm, and they can usually guarantee optimality and run fast, but lack flexibility of extending to various problems, because a logic rule is often purposed-designed for a specific problem. Stochastic methods are usually decentralized, bottom-up, agent-based simulation model, and they often output non-optimal solutions and run relatively slow due to their population-based nature, but they can be easily modified for various problems, because agent behavior can be flexibly defined according to problem characteristics. RSA is actually decentralized, bottom-up, agent-based simulation model, but by defining the ripple-spreading behavior of individual nodes, it is guaranteed that optimality will automatically and quickly emerge as a result of the collective performance of the model. In other words, RSA exhibits the merits of both classical deterministic methods and stochastic methods, and avoid their demerits for path optimization [6]-[8].

Basically, the procedure of RSA is simple: an initial ripple is generated at the source, and it spreads out at a specified speed; when it reaches an unvisited node, it triggers a new ripple, which will spread and trigger more ripples at other nodes; ripple spreading and triggering behaviors go on and on until the destination has been reached by a ripple; then the shortest path between the source and the destination can be determined by backtracking the ripple which has arrived at the destination first. Fig. 2 illustrates how the RSA in [6] finds the shortest path by generating and spreading ripples step by step. The procedure of RSA is likened to a ripple relay race. By defining ripple spreading and triggering behaviors according to the characteristics of a given problem, RSA can be modified and extended to various POPs [6].

It should be emphasized that single-objective POPs are the focus of RSA in existing literatures (e.g., the RSA employed in [44] can find just the single-objective k shortest paths). Because of the theoretical and practical importance of MOPOP, and also because of the complexity of MOPOP, RSA needs to be extended to MOPOP in order to prove its full value. Actually, this paper will extend RSA not only to those MOPOPs which have already been studied in existing literatures, but also to some MOPOPs that have rarely been tackled so far. To apply RSA to static MOPOPs, new ripple spreading and triggering behaviors need to be introduced for RSA with respect to the characteristics of static MOPOPs.

3.2. The design of new ripple-spreading algorithm for one-to-one static problem

Here, a pure RSA will be designed, no need of any other existing measures (such as constructing AOFs, applying COFs, conducting stochastic search, or calculating N_{obj} sets of the k best single-objective paths like in [44]), to find the complete Pareto front for the one-to-one static MOPOP defined in Section 2, with both theoretical and practical guarantee of optimality. Basically, to apply RSA to static MOPOP, new agent behavior

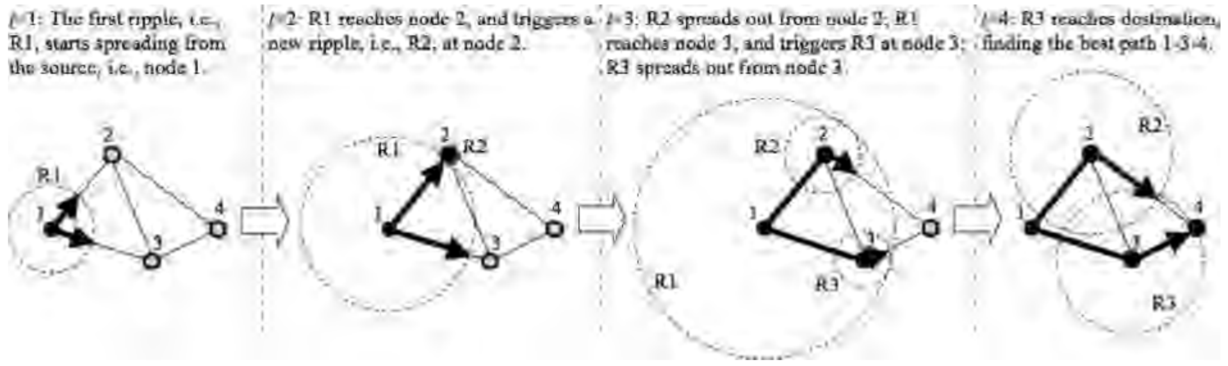

t=1: The first ripple, i.e., R1, starts spreading from the source, i.e., node 1.
t=2: R1 reaches node 2, and triggers a new ripple, i.e., R2, at node 2.
t=3: R2 spreads out from node 2; R1 reaches node 3, and triggers R3 at node 3; R3 spreads out from node 3.
t=4: R3 reaches destination, finding the best path 1-3-4.
R1
R2
R3
1
2
3
4

Fig. 2. How the RSA reported in [6] finds the best path.

and termination criteria need to be defined. If an incoming ripple is dominated by any existing Pareto non-dominated ripple (PNDR) of a node, then the incoming ripple will trigger no new ripple at the node; and the macro termination criteria becomes that there is no active ripples any more.

Here is the definition of PNDR. Assume ripple r travels from S to node n through the path P_r. In terms of the ith objective, i=1,..,N_{Obj}, path P_r has a cost of $f_i(P_r)$. Assume by the time when ripple r arrives at node n, another m ripples, i.e., $r_1,\ldots,r_m$, have already got to node n. If the following conditions are not satisfied for every $k \in [1, \ldots, m]$,

$$f_i\left(P_{r_k}\right) \leq f_i\left(P_r\right), \textit{for all } i = 1, \ldots, N_{Obj} \tag{7}$$

$$f_j\left(P_{r_k}\right) < f_j\left(P_r\right), \textit{for at least one } j \in \left[1, \ldots, N_{Obj}\right] \tag{8}$$

then for node n, ripple r is a PNDR.

The pseudocode of RSA for static MOPOP based on a given pair of [S,D], i.e., one-to-one static MOPOP, is described as following. Let $S_{\text{R}}(r)$ denote the state of ripple r, and $S_{\text{R}}(r)$=0/1 means ripple r is inactive/active. $E(r)$ denotes the epicenter of ripple r, i.e., ripple r is generated at node $E(r)$. $R(r)$ records the radius of ripple r. $T(r)$ records which ripple triggers ripple r. $\Omega_{\text{PNDR}}(n)$ is the set of Pareto non-dominated ripples of node n. For a ripple $r \epsilon \Omega_{\text{PNDR}}(n)$, in terms of the ith objective, i=1,…,N_{Obj}, ripple r has a cost of $F_i(r,n)$ to travel from S to node n. Table 1 gives a list of all main variable symbols for reader reference purposes.

Step 1: Assuming the hth objective has the maximal value of min($C_h(i,j)$)/max($C_h(i,j)$) among all of the N_{Obj} objectives, then, for the sake of both optimality and computational efficiency [6], set the ripple spreading speed as

$$v = \min\left(C_h(i,j)\right),\ i = 1, \ldots, N_N, j = 1, \ldots, N_N \tag{9}$$

Step 2: Initialize $\Omega_{\text{PNDR}}(n)$=Φ for every node $1 \leq n \leq N_{\text{N}}$. Initialize the first ripple at the source S, and set the current number of ripples as N_{R}=1. Set $E(N_{\text{R}})$=S, $R(N_{\text{R}})$=0, $S_{\text{R}}(N_{\text{R}})$=1, and $T(N_{\text{R}})$=0 (which means the first ripple is self-triggered). Let $\Omega_{\text{PNDR}}(S)$={N_{R}}, and set $F_i(N_{\text{R}},S)$=0, i=1,…,N_{Obj}. Set t=0.

Step 3: If $S_{\text{R}}(r)$=0 for every r=1,...,N_{R}, i.e., if there is no active ripple any more, or if for any $S_{\text{R}}(r)$=1, $1 \leq r \leq N_{\text{R}}$, the costs of the so-far path of ripple r are not Pareto non-dominated by those existing PNDRs on the destination D, then go to Step 7; Otherwise, let t=t+1, and go to Step 4.

Step 4: For each active ripple r, i.e., if $S_{\text{R}}(r)$=1, $1 \leq r \leq N_{\text{R}}$, update the radius of ripple r by v, i.e.,

$$R(r) = R(r) + v. \tag{10}$$

Step 5: For any node n, if there is at least one $1 \leq r \leq N_{\text{R}}$ that has $S_{\text{R}}(r)$=1, $A(E(\text{r}),n)$=1 and $R(r) \geq C_h(E(r),n)$, i.e., ripple r is a new incoming ripple to node n, then, based on all objectives, conduct comparison between all new incoming ripples to node n (if there are more than 1 new incoming ripple), and also conduct comparison between the new incoming ripples to node n and the existing PNDRs on node n, in order to find out such new incoming ripples which are Pareto non-dominated by any other new incoming ripples to node n, or by any existing PNDR on node n. For each of such new incoming ripples to node n, say, ripple r is such a new incoming ripple to node n, then, ripple r becomes a new PNDR of node n, so, update $\Omega_{\text{PNDR}}(n)$=$\Omega_{\text{PNDR}}(n)$+{r}, and set $F_i(r,n)$, i=1,…,N_{Obj}, according to the costs of the path travelled by ripple r from S to node n; and ripple r as PNDR will trigger a new ripple at node n: Let N_{R}=N_{R}+1; set $E(N_{\text{R}})$=n, $T(N_{\text{R}})$=r, $R(N_{\text{R}})$=$R(r)$-$C_h(E(r),n)$; and if $n \neq D$, set $S_{\text{R}}(N_{\text{R}})$=1, otherwise, set $S_{\text{R}}(N_{\text{R}})$=0, i.e., a new ripple triggered at the destination node is inactive.

Step 6: For any active ripple r, i.e., $S_{\text{R}}(r)$=1, if for any node n that has $A(E(r),n)$=1, the following condition holds

$$R(r) \geq C_h(E(r), n), \tag{11}$$

then set $S_{\text{R}}(r)$=0, i.e., ripple r becomes inactive. Go to Step 3.

Step 7: For every PNDR on the destination node, i.e., for every $r \epsilon \Omega_{\text{PNDR}}(D)$, $1 \leq r \leq N_{\text{R}}$, [$F_1(r,D)$,…, $F_{N\text{Obj}}(r,D)$] is a Pareto point, and the path travelled by ripple r from S to D is the associated Pareto optimal solution. In this way, the complete Pareto front and all Pareto optimal paths are determined by all of those PNDRs on D.

In the above pseudocode of RSA, one can see that ripples spread on a network with the hth objective as link cost, as both ripple-spreading speed and ripple stop condition are defined based on the hth objective, i.e., see Eq.(9) and Eq.(11). However, during the ripple relay race, whether a ripple is PNDR or not is judged based on all the i=1,…,N_{Obj} objectives (see Step 2 and Step 5).

Step 4 to Step 6 are the core of RSA, and they define ripple spreading, triggering and deactivating behaviors, respectively. Step 3 tells when the ripple relay race should be stopped, i.e., when no active ripple exists, the ripple relay race is terminated to go to Step 7, where the complete Pareto front will be identified.

One might notice that in Step 5, when updating the set of Pareto non-dominated ripples (PNDRs), it only needs to add in new PNDRs, and never needs to delete any existing PNDRs from set $\Omega_{\text{PNDR}}(n)$. This is rather different from most evolutionary computation methods (such as genetic algorithm, ant colony optimization and particle swarm optimization), where those non-dominated solutions found in a generation could turn out dominated by some solutions found in the next generation. Fortunately, this situation will not arise in RSA, thanks to the ripple spreading optimization principle as revealed in [6]. In other words, once a ripple r is qualified to be added into $\Omega_{\text{PNDR}}(n)$, it will never be dominated by any ripple which arrives at node n after ripple r (see Lemma 1 in Section 4.1 for theoretical proof). Therefore, compared with other evolutionary computation methods, no need of deleting any so-far found PNDRs implies a great searching efficiency of RSA.

Fig. 3 shows the fundamental difference between the procedures of the method in [44] and the new RSA proposed here for MOPOP. From

Table 1
List of main variable symbols and acronyms

f_k	The kth objective in MOPOP; $f_k(P)$ is the kth objective of path P
f_{REED}	Function of routing environment evolving dynamics
v	Ripple spreading speed; v should be satisfy Condition (9), so that RSA can guarantee to find complete Pareto front for a MOPOP
A	Adjacent matrix, $A(i,j)$=1 means a link from node i to node j
$A_{z\|0}$	Dynamical adjacent matrix for time z predicted at time 0
C	Cost matrix, $C_k(i,j)$ is associated with link $A(i,j)$, and is used to calculate the kth objective of a path
$C_{z\|0}$	Dynamical cost matrix for time z predicted at time 0
D	Destination node
$E(r)$	Epicenter of ripple r
$F_i(r,n)$	Cost of the path travelled by ripple r from S to node n in terms of the ith objective
N_{ATU}	Number of simulation time units a ripple takes to go through a link
N_{L}	Number of all links in a route network
N_{N}	Number of all nodes in a route network
N_{Obj}	Number of objectives in MOPOP
N_{R}	Number of ripples ever generated so far
N_{P}	Number of nodes, including S and D, included in path P
N_{PP}	Number of Pareto points in a given MOPOP
P	Path, $P(i)$=j means node j is the ith node in path P
P^*	Pareto-optimal path; P^* must satisfy Conditions (5) and (6)
$R(r)$	Radius of ripple r; $R(r)$ increases by v each time as ripple r spreads
S	Source node
$S_{\text{R}}(r)$	State of ripple r; $S_{\text{R}}(r)$=0/1 means ripple r is inactive/active.
$T(r)$	Ripple that triggers ripple r
Ω_{P}	Set of all possible paths connecting S and D
$\Omega_{\text{PNDR}}(n)$	Set of Pareto non-dominated ripples (PNDRs) on node n
AOF	Aggregate objective function
CEPO	Co-evolutionary path optimization
COF	Constrained objective function
MOEA/D	Decomposition based evolutionary algorithms
MOOP	Multi-objective optimization problem
MOPOP	Multi-objective path optimization problem
OPRO	Online path re-optimization
PCR	Pareto-compliant ranking
PNDR	Pareto non-dominated ripples
POP	Path optimization problem
RSA	Ripple-spreading algorithm
SOOP	Single-objective optimization problem
SOPOP	Single-objective path optimization problem
TDPO	Time-dependent path optimization

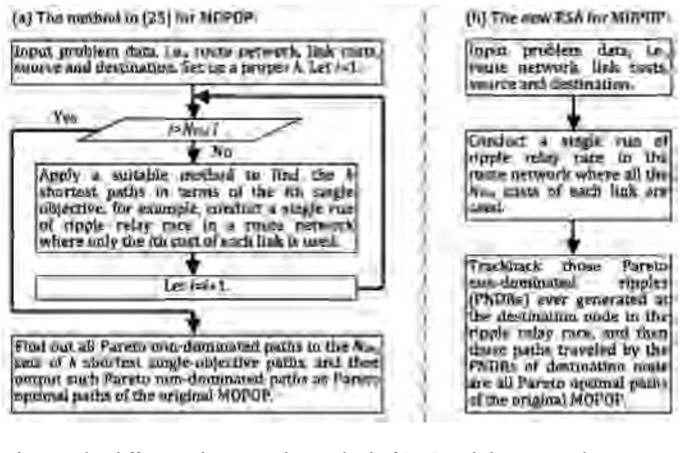
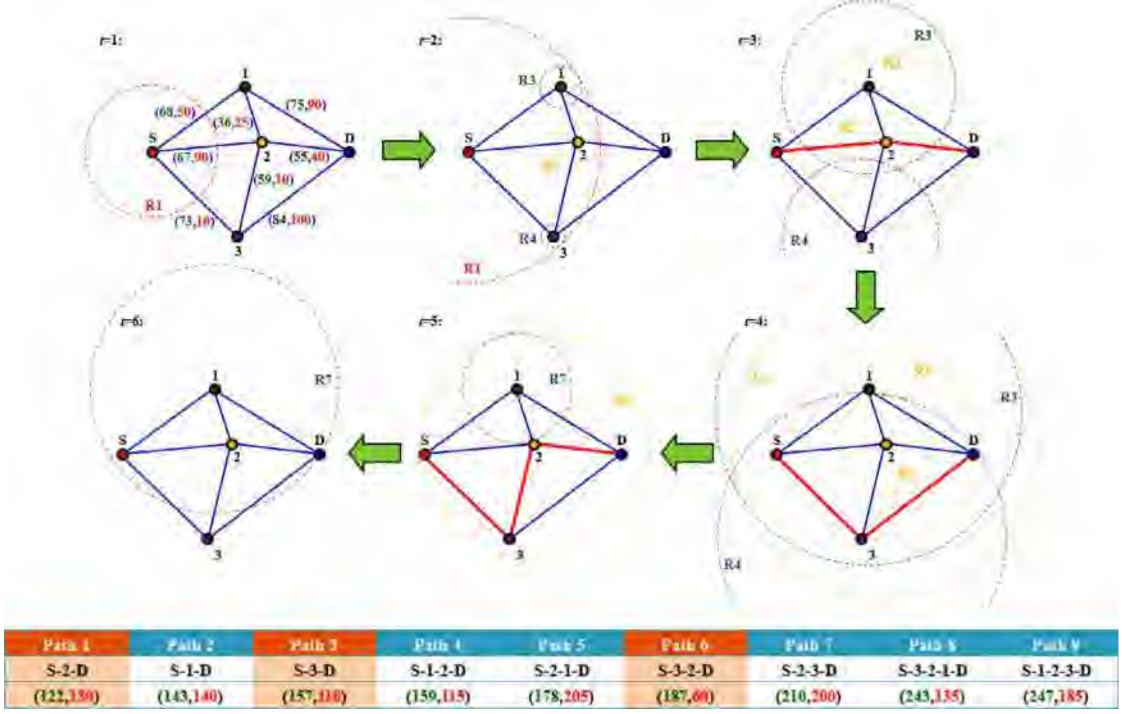

(a) The method in [25] for MOPOP
Input problem data, i.e., route network, link costs, source and destination. Set up a proper k. Let i=1.
Yes
i>N_Obj?
No
Apply a suitable method to find the k shortest paths in terms of the ith single objective, for example, conduct a single run of ripple relay race in a route network where only the ith cost of each link is used.
Let i=i+1.
Find out all Pareto non-dominated paths to the N_Obj sets of k shortest single-objective paths and then output such Pareto non-dominated paths as Pareto optimal paths of the original MOPOP.
(b) The new RSA for MOPOP
Input problem data, i.e., route network, link costs, source and destination.
Conduct a single run of ripple relay race in the route network where all the N_Obj costs of each link are used.
Trackback those Pareto non-dominated ripples (PNDRs) ever generated at the destination node in the ripple relay race, and then those paths traveled by the PNDRs of destination node are all Pareto optimal paths of the original MOPOP.

Fig. 3. The difference between the method of [44] and the reported new RSA.

Fig. 3 one can see clearly that (i) the method of [44] does not necessarily rely on RSA, and any algorithm capable of finding the k shortest single-objective paths can be used as an inner optimizer within the method of [44]; (ii) if RSA is employed by the method of [44], RSA needs to be run for at least N_{Obj} times; (iii) the new method reported here is a pure RSA, and it takes only a single run of ripple relay race to identify the complete Pareto front.

Fig. 4 gives an illustration about how the above RSA finds the complete Pareto front for an MOPOP. The MOPOP in Fig. 4 is a bi-objective path optimization problem, $C_1(i,j)$ for objective 1, i.e., the green number in bracket, is physical distance between nodes, and $C_2(i,j)$ for objective 2, i.e., the red number in bracket, is random weight. Pareto optimal paths are highlighted as bold red line, and the data of Pareto optimal paths are highlighted with orange background. Other colors and styles of node, link and data are only for the sake of good illustrating effect. The ripple relay race of RSA starts with ripple R1 at the source S. During time 2, R1 reaches nodes 2, 1 and 3 in turn. Since nodes 2, 1 and 3 have no Pareto non-dominated ripple (PNDR) yet before R1 arrives, R1 becomes the first PNDR at them and triggers new ripples R2 at node 2, R3 at node 1, and R4 at node 3. R1 then becomes inactive as it has reached all the ends of its links. R2 reaches the destination D at time 3, becomes the first NPDR at D, and identifies route S-2-D as a Pareto optimal path. R3 reaches node 2 during time 3. By comparing the objective values of R3 when R3 reaches node 2 with those of R1 when R1 reaches node 2, it is clear that R3 is not dominated by R1, so, R3 becomes the second PNDR at node 2, and then triggers a new ripple R5 at node 2. Although R2 reaches node 1 during time 3, R2 is however dominated by R1, the first NPDR at node 1, and therefore fails to trigger any new ripple at node 1. During time 4, R3 reaches D first, but fails to identify any new Pareto optimal path as R3 is dominated by R2, the first PNDR at D; then, R4 follows R3 to reach D, becomes the second PNDR at D, and identifies path S-3-D as another Pareto optimal path; R5 closely follows R4 to reach D, but it is dominated by R4; R4 also reaches node 2, and since R4 is not dominated by any previous PNDR at node 2 (i.e., R1 and R3), it becomes the third PNDR at node 2 and triggers ripple R6 at node 2; R2 reaches node 3, but it is dominated by the first PNDR at node 3, i.e., R1z. During time 5, R6 reaches D and identifies path S-3-2-D as a new Pareto optimal path as R6 is not dominated by any previous PNDR at D (i.e., R2 and R4); R6 also reaches node 1, becomes the second PNDR at



Path 1	Path 2	Path 3	Path 4	Path 5	Path 6	Path 7	Path 8	Path 9
S-2-D	S-1-D	S-3-D	S-1-2-D	S-2-1-D	S-3-2-D	S-2-3-D	S-3-2-1-D	S-1-2-3-D
(122,130)	(143,140)	(157,110)	(159,115)	(178,205)	(187,60)	(210,200)	(243,135)	(247,185)

Fig. 4. An illustration about how the complete Pareto front is found in a single run of ripple relay race of RSA for one-to-one static MOPOP.

node 1 and triggers R7 at node 1. At time 6, R7 reaches D, but fails to identify any new Pareto optimal path, as it is dominated by all previous PNDRs at D. After time 6, there is no active ripple, so the ripple relay race terminates. Then, the complete Pareto front is determined by those paths identified by all PNDRs at D, i.e., R2, R4 and R6.

3.3. *Ripple-spreading algorithm for one-to-all static problem*

The above RSA is developed for one-to-one static MOPOP. Fortunately, it is very easy and highly straightforward to modify the RSA in order to apply to one-to-all static MOPOP, where all complete Pareto fronts for a given S to each of other (N_N-1) nodes in the network need to be found. To resolve a one-to-all static MOPOP, there is no need to repeat the above one-to-one RSA for (N_N-1) times, and it is enough to conduct a one-off ripple relay race where micro agent behaviors are slightly modified, i.e., there is no destination and therefore a new ripple is always triggered as active (whilst for one-to-one static MOPOP, a ripple triggered at D is always inactive). Due to limited space, here the details of the RSA for one-to-all static MOPOP are skipped, and only a simple illustration is given in Fig. 5 to show how a single run of ripple relay race can identify all complete Pareto fronts in a one-to-all bi-objective problem, i.e., those complete Pareto fronts with nodes 1 and 2 as source and destination, with nodes 1 and 3 as source and destination, and with nodes 1 and 4 as source and destination, respectively.

In Fig. 5, the first ripple, i.e., R1, starts spreading out from the source, i.e., node 1 at time t=1. By time t=2, R1 has got to nodes 2 and 3. There is no PNDR at node 2 or 3 by time t=2, so, the first PNDR of both nodes is R1, and R1 identifies path 1-2 as a Pareto optimal path from node 1 to node 2, and path 1-3 as a Pareto optimal path from node 1 to node 3. R1 also triggers new ripple R2 at node 2 and new ripple R3 at node 3. Then, R1 becomes inactive (or dead) because it has reached all nodes that link to its epicenter node. At time t=3, R2 and R3 keep spreading. By time t=4, R2 has reached both node 3 and node 4. At node 3, R2 is dominated by R1, i.e., the existing PNDR of node 3, so, R2 triggers no new ripple at node 3. At node 4, R2 becomes the first PNDR of node 4, so, it identifies path 1-2-4 as a Pareto optimal path from node 1 to node 4. R2 also triggers R4 at node 4. By time t=4, R3 has got to nodes 4 and 2. At node 4, no existing PNDR dominates R3, so, R3 becomes the second PNDR of node 4, and identifies another Pareto optimal path from node 1 to node 4, i.e., path 1-3-4. R3 also triggers R5. At node 2, no existing PNDR dominates R3, so, R3 identifies path 1-3-2 as another Pareto optimal path between nodes 1 and 2. R3 also triggers R6 at node 2. After time t=4, both R2 and R3 have become inactive. At time t=5, R4, R5 and R6 keep spreading. By time t=6, R4 has got to nodes 2 and 3, however, R4 is dominated at either node 2 or 3. By time t=6, R5 has also reached both nodes 2 and 3, however, R5 is also dominated at either node 2 or 3. By time t=6, R6 has reached node 4, non-dominated by any existing PNDRs of node 4, so, R6 is the third PNDR at node 4, and identifies the third Pareto optimal path from node 1 to node 4, i.e., path 1-3-2-4. After time t=6, all ripples have become inactive, so, the ripple relay race stops, and all complete Pareto fronts are identified by those PNDRs of nodes 2, 3 and 4, respectively.

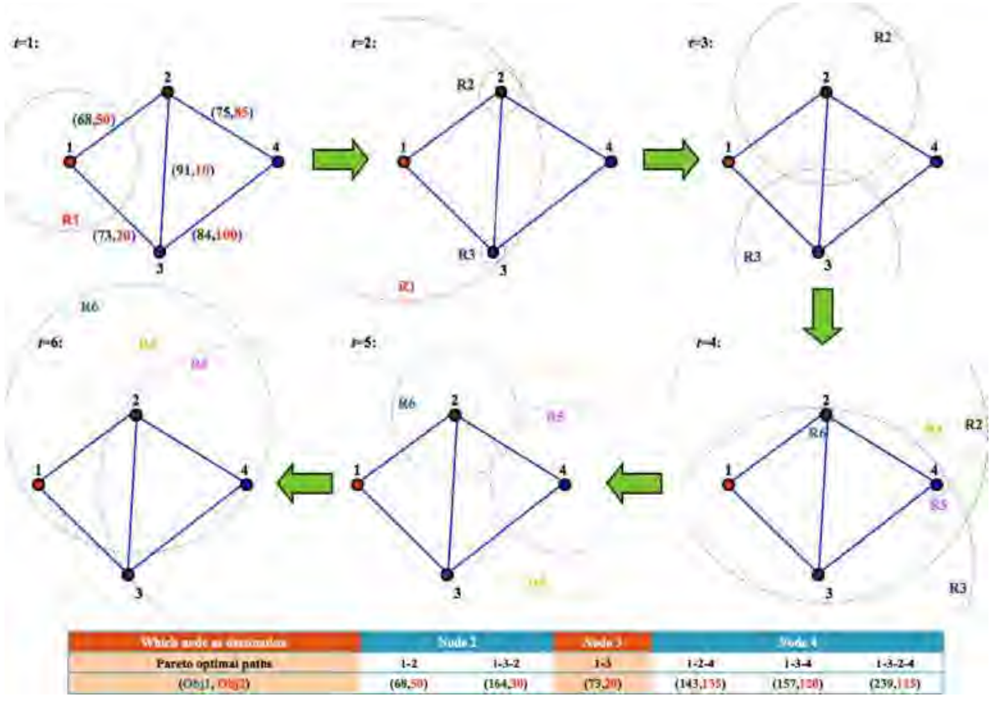
Further Analyses

This section first conducts theoretical analyses about the optimality and computational efficiency of the reported RSA for static MOPOPs, and then goes on to extend RSA from static MOPOPs to dynamical MOPOPs.

4.1. *Optimality of ripple-spreading algorithm for static multi-objective path optimization problems*

Lemma 1: Once ripple r is added as a PNDR at node n at time t, then, it will never be dominated by any ripple that reaches node n at time t+z, z>0.

Proof: Assume ripple r reaches node n at time t_1. According to Step 5 in the RSA pseudocode of Section 3, one has $t_1 \le t$. When ripple r reaches node n, it has a value of $v \times t_1$ for the hth objective function according to Eq.(9). For a ripple that reaches node n at time t_2=t+z, z>0, it has a value of $v \times t_2$ for the hth objective function. Since $t_1 \le t < t+z = t_2$, one knows for sure that the ripple arriving at time t_2 has a larger value for the hth objective function than ripple r has, and therefore, it cannot dominate



Which node as destination	Node 2		Node 3	Node 4		
Pareto optimal paths	1-2	1-3-2	1-3	1-2-4	1-3-4	1-3-2-4
(Obj1, Obj2)	(68,50)	(164,30)	(73,20)	(143,135)	(157,120)	(239,115)

Fig. 5. An illustration of ripple relay race of RSA for one-to-all static MOPOP.

ripple r according to the definition of Pareto optimality by Eq.(5) and Eq.(6).

Lemma 1 guarantees that Step 5 in the RSA pseudocode of Section 3 will never miss out any Pareto non-dominated ripples.

Lemma 2: Suppose a ripple reaches a node. If this ripple is dominated by any previous PNDR at the node, then any path that connects S and D and covers the path this ripple has travelled through is not Pareto optimal.

Proof: Let P_1 be the path travelled through by this ripple, including N_{L1} nodes, and P_2 the path by the PNDR which dominates this ripple, including N_{L2} nodes. Then, one has $P_1(1)=P_2(1)=S$, and $P_1(N_{L1})=P_2(N_{L2})$. According to the definition of Pareto dominance, one knows that (i) $f_k(P_1)\geq f_k(P_2)$ for all $k=1,\ldots,N_{\mathrm{Obj}}$, and (ii) there exists an $h\in[1,\ldots,N_{Obj}]$ which makes $f_h(P_1)>f_h(P_2)$.

A path covering P_1 and connecting S and D can be formulated as $P_4=P_1+P_3$. Then, a new path $P_5=P_2+P_3$ can be constructed, which obviously also connects S and D. Since all objective functions satisfy condition (2) in this study, one has that (i) $f_k(P_4)=f_k(P_1+P_3)\geq f_k(P_2+P_3)=f_k(P_5)$ for all $k=1,\ldots,N_{\mathrm{Obj}}$, and (ii) $f_h(P_4)=f_h(P_1+P_3)>f_h(P_2+P_3)=f_h(P_5)$. This means P_5 dominates P_4. Therefore, any path covering P_1 is not Pareto optimal to connect S and D.

Lemma 3: Suppose a path P^* is Pareto optimal connecting S and D. Then, after the ripple relay race terminates, there is a ripple triggered at D that has traveled through this path P^*.

Proof: Assume Lemma 3 is false, which means there must be a node in P^*, say, the ith node $P^*(i)$, $1<i\leq N_{\mathrm{L}}^*$ (N_{L}^* is the number of nodes in P^*), that has eliminated the ripple which has reached $P^*(i)$ by travelling through nodes $P^*(1),\ldots,P^*(i\text{-}1)$. Then, according to Lemma 2, P^* is not Pareto optimal as it covers the path $[P^*(1),\ldots,P^*(i\text{-}1)]$. So, Lemma 3 must be true.

Lemma 2 guarantees that (i) all those ripples which are eliminated by PNDRs at intermediate nodes are of no use for identifying any Pareto optimal path connecting S and D, which can lead to computational efficiency, and (ii) all those ripples which are triggered at D are associated with Pareto optimal paths, which is the necessary condition of optimality. Lemma 3 guarantees that, when the ripple relay race terminates, every Pareto optimal path connecting S and D will have an associated ripple triggered at D, which is the sufficient condition of optimality. Based on Lemmas 2 and 3, one has the following theorem for the optimality of the proposed RSA.

Theorem 1: In the proposed RSA for static MOPOPs, when the ripple relay race terminates, the complete Pareto front is just determined by all of those ripples which have been triggered at D.

4.2. *Complexity of ripple-spreading algorithm for static multi-objective path optimization problems*

Theorem 2: Suppose, for a given network with N_{N} nodes, N_{L} links and a specified source S, each of other (N_{N}-1) nodes has N_{PP} Pareto points on average in its associated complete Pareto front. Then, the computational complexity of RSA for one-to-all N_{Obj}-objective POP is about $O(N_{\mathrm{Obj}}\times N_{\mathrm{L}}\times N_{\mathrm{PP}}^2)$.

Proof: Based on Theorem 1, one can deduce that in the ripple relay race on average, each node (other than S) will have N_{PP} PNDRs and therefore will be triggered to generate N_{PP} new ripples. There are mainly three kinds of basic computational steps in RSA: increase the radius of a ripple by the constant ripple spreading speed, check if a ripple has reached the end of a link, and check if an incoming ripple is dominated by any previous PNDR at a node. The first kind is an addition operation, the second kind is a comparison operation, and the third kind includes calculating N_{Obj} so-far objective values for a ripple and comparing each value with up to N_{PP} previous PNDRs at a node. On average, each node has $2\times N_{\mathrm{L}}/N_{\mathrm{N}}$ links. Suppose on average, it takes N_{ATU} simulation time units for a ripple to travel through a link. Then, before a ripple becomes inactive, it may on average need $(2\times N_{\mathrm{ATU}}+N_{\mathrm{Obj}}+N_{\mathrm{Obj}}\times N_{\mathrm{PP}})\times 2\times N_{\mathrm{L}}/N_{\mathrm{N}}$ operations. Since the source S will generate only one ripple and each of other (N_{N}-1) nodes will generate N_{PP} ripples on average, one then has that, before all ripples become inactive (when the ripple relay race terminates), it will take about $(1+(N_{\mathrm{N}}\text{-}1)\times N_{\mathrm{PP}})\times(2\times N_{\mathrm{ATU}}+N_{\mathrm{Obj}}+N_{\mathrm{Obj}}\times N_{\mathrm{PP}})\times 2\times N_{\mathrm{L}}/N_{\mathrm{N}}$ operations. Usually, $N_{\mathrm{PP}}>>N_{\mathrm{Obj}}$ and also $N_{\mathrm{Obj}}\times N_{\mathrm{PP}}>>2\times N_{\mathrm{ATU}}$, so, the computational complexity can be assessed as $O(N_{\mathrm{Obj}}\times N_{\mathrm{L}}\times N_{\mathrm{PP}}^2)$.

If the method of [44] is applied to resolve the same one-to-all MOPOP, it needs to set up a proper k to satisfy certain theoretical conditions and then run the single-objective RSA in [44] for N_{Obj} times. The problem is that it is difficult to pre-know which k is proper for a given MOPOP, so in practice, one needs to test k one by one by increasing k from 1 to a value that satisfies those theoretical conditions in [44], and for each k value, one needs to run the single-objective RSA in [44] for N_{Obj} times, and then check the Pareto dominating relationships between N_{Obj} sets of single-objective k shortest paths (the number of paths in the N_{Obj} sets of single-objective k shortest paths is at least N_{PP}, but usually much larger than N_{PP}). The complexity of the single-objective RSA of [44] for a single problem of k shortest paths is about $O(k \times N_{\text{L}} \times N_{\text{ATU}})$ [8]. Therefore, the complexity of method [44] for one-to-all MOPOP can be roughly assessed as $O(k^2 \times N_{\text{Obj}} \times N_{\text{L}} \times N_{\text{ATU}} \times N_{\text{PP}}^2)$, which is clearly much higher than the complexity of the RSA reported in this paper.

In path optimization, the space complexity of a method is often analyzed for single-objective one-to-one problem, and it usually implies how many nodes and links in a route network will be explored before the method finds a solution. However, for complex path optimization such as the k shortest paths problem, MOPOP and one-to-all problem, this kind of space complexity analysis is not very popular or useful, because all nodes and links in the route network will usually be explored before the algorithm can find solutions. Another possible definition of space complexity in path optimization is: the ratio of the number of explored paths against all possible paths between source and destination. It is difficult to assess the number of either explored paths or all possible paths, particularly in complex path optimization. So, this kind of space complexity analysis is rare and might be worth future investigation.

Besides the above theoretical complex analysis, the practical computational efficiency of RSA may be influenced by some algorithm parameters, such as ripple-spreading speed v. The influence of such algorithm parameters is a common issue to all RSAs, no matter RSA is designed for MOPOP or not. For more discussions about RSA parameters, readers may refer to some previous publications on RSA [6]-[7].

4.3. Extending ripple-spreading algorithm to dynamical multi-objective path optimization problems

One might argue that the RSA reported in Section 3 is just another method to a well-resolved problem, because there have already been a few methods that can calculate the complete Pareto front to the static MOPOP of Section 2 (e.g., see References [44]-[49]). In order to better verify the novelty and contribution of this paper, here the RSA is extended to a much more complicated MOPOP, i.e., calculating the complete Pareto front in dynamical routing environments, which is called dynamical MOPOP in this paper. To our best knowledge, those methods in [44]-[49] are all focused on static MOPOP, no attempt has ever been reported to extend those methods of [44]-[49] to dynamical MOPOP. Actually in [17], RSA has been preliminarily tested to identify complete Pareto front for one-to-one dynamical MOPOP, and here it will be further extended to one-to-all dynamical MOPOP.

Although theoretical researches like to use static problems, real-world applications often have to take into account some time-varying factors [50],[51]. Taking dynamical MOPOP as an example, a time-varying network means nodes and/or links may temporarily become inaccessible (i.e., $A(i,j)$ can change between 1 and 0), and the costs of links may change from time to time (i.e., $C_k(i,j)$ can change as time goes on). It is usually the case that static optimal solutions are not suitable to a dynamical problem.

The following is the mathematical description of dynamical MOPOP. Assuming at the initial time t=0, it is predicted that there will be a direct link between nodes i and j at time z, then $A_{z|0}(i,j)$=1. It is also predicted at time t=0 that the link between nodes i and j at time z has costs $C_{k,z|0}(i,j)$, k=1,…,N_{Obj}. In this paper, it is assumed that travelling time (or a similar index) is the 1st objective function, i.e., it will take time $C_{1,z|0}(i,j)$ to pass link $A_{z|0}(i,j)$. Based on the time-varying network defined by $A_{z|0}$ and $C_{k,z|0}$, the following minimization problem well defines dynamical MOPOP:

$$\min_{P\in\Omega_P} \left[f_1(P, N_P), f_2(P, N_P), ..., f_{N_{Obj}}(P, N_P)\right], \tag{12}$$

subject to

$$A_{z|0}(P(i), P(i+1)) = 1,\ z = z_i, ..., z_{i+1}, \tag{13}$$

$$f_k(P, 1) = 0, k = 1, \dots, N_{\text{Obj}}, \tag{14}$$

$$f_1(P, i+1) = \max(f_1(P, i), z_i|0) + C_{1,z_i|0}(P(i), P(i+1)), i = 1, \dots, N_P - 1, \tag{15}$$

$$f_k(P, i+1) = \sum_{j=1}^{i} C_{k,z_j|0}(P(j), P(j+1)), k = 2, \dots, N_{\text{Obj}}, \tag{16}$$

where P, N_{P}, N_{Obj} and Ω_{P} have the same definitions as in Section 2, $f_k(P,i)$ is the cost to travel along P to get to the ith node of P in the time-varying network defined by $A_{z|0}$ and $C_{k,z|0}$, and assuming after node $P(i)$ is reached by traveling along P, the earliest time for $P(i)$ to become passible is z_i.

According to Eq.(13) to Eq.(16), one can see that both accessibility and costs of a link in path P rely on what time to pass the link. This is completely different from those MOPOPs in References [44] and [45]-[49]. To predict time-varying $A_{z|0}$ and $C_{k,z|0}$, this paper assumes a routing environment evolving dynamics (REED) is pre-given as follows:

$$\left[A_{z+1|0}, C_{1,z+1|0}, ..., C_{N_{Obj},z+1|0}\right] = f_{REED}\left(A_{z|0}, C_{1,z|0}, ..., C_{N_{Obj},z|0}\right), z \geq 0, \tag{17}$$

subject to $A_{0|0}$=A_0 and $C_{k,0|0}$=$C_{k,0}$, k=1,…,N_{Obj}, where f_{REED} is the function of routing environment evolving dynamics, A_0 is the adjacent matrix measured (not predicted) at the initial time t=0, and $C_{i,0}$ is the measured cost matrix in terms of the ith objective at the initial time t=0, i=1,…,N_{Obj}.

It should be emphasized that f_1 in Eq.(15) is calculated rather differently from f_k in Eq.(16), k=2,…,N_{Obj}, because as traveling time, f_1 relies on both link cost $C_{1,z|0}$ and node accessible time with $A_{z|0}(i,j)$=1. If a node is reached during its inaccessible time by a ripple, then waiting behavior is necessary for the ripple in front of the node, and f_1 must include such waiting time. In terms of another N_{Obj}-1 objective functions, there is no additional cost due to waiting behavior in front of a node, and f_k is just the sum-up of relevant $C_{k,z|0}$, k=2,…,N_{Obj}. For example, the service charge of highway segment relies on whether it is peak time or off-peak time, and traffic jam time usually cannot be converted to service charge.

Apparently, according to Eq.(13), Eq.(15) and Eq.(16), the dynamical MOPOP defined by Eq.(12) needs to resolved by observing those changes in not only link accessibility but also link costs as described by Eq.(17).

Actually, single-objective path optimization problem (SOPOP) in dynamical routing environment is often discussed by researchers. Here, only deterministic methods are reviewed, because calculating complete Pareto front demands an optimality guarantee, which stochastic methods usually lack. There are mainly two categories of methods reported to address dynamical SOPOP: one category is online path re-optimization (OPRO) [52],[53], and the other is time-dependent path optimization (TDPO) [54]-[56]. As illustrated in Fig. 6, OPRO cannot ensure to find the theoretical optimal path in a given dynamical network in terms of a single objective, let alone dynamical MOPOP. TDPO does not need online re-optimization, and for a given dynamical routing environment, it takes only a single offline run to make the resulted actual traveling trajectory optimal with a theoretical guarantee, which comes from a static time-expanded hypergraph recording all the information of N_{TUinP} pairs

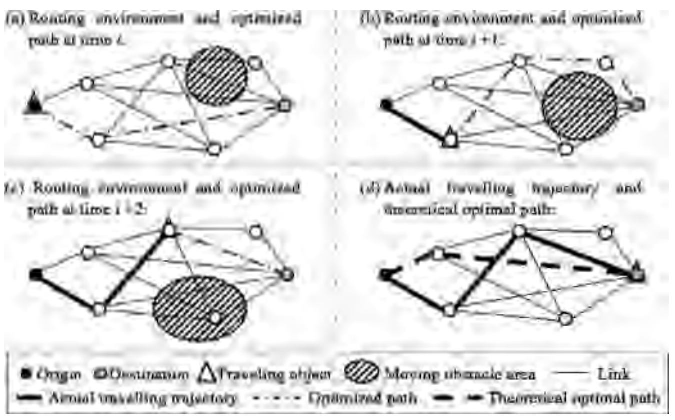

(a) Routing environment and optimized path at time t.
(b) Routing environment and optimized path at time t + 1.
(c) Routing environment and optimized path at time t + 2.
(d) Actual travelling trajectory and theoretical optimal path.
Origin
Destination
Traveling object
Moving obstacle area
Link
Actual travelling trajectory
Optimized path
Theoretical optimal path

Fig. 6. Online path re-optimization (OPRO) cannot find the actual optimal travelling trajectory.

of $[A_{z|0},C_{z|0}]$, where N_{TUinP} is the number of time units taken to travel between a given pair of source and destination nodes along an optimal or promising path. However, dealing with a time-expanded hypergraph is generally complicated and memory-expensive, and TDPO methods often have a serious scalability issue even for single-objective problem. Therefore, there has rarely been any attempt ever reported to extend TDPO methods to dynamical MOPOP.

Rather different from both OPRO and TDPO, a novel co-evolutionary path optimization (CEPO) method has recently been reported to address dynamical SOPOP [7]. Basically, CEPO does not need either online re-optimization, or time-expanded hypergraph, and a single offline run of CEPO method based on a network of original size can output actual optimal travelling trajectory with a theoretical guarantee.

Reference [7] particularly modified RSA to realize CEPO. Basically, not only those changes in link accessibility and costs but also the ripple relay race of RSA happens based on the same simulated time unit. So, the changing of route network can be seamlessly integrated into the ripple spreading procedure of RSA. This seamless integration of ripple-relay race and changes in link accessibility and costs provides the foundation to ensure RSA to find optimal actual traveling trajectory in a given dynamical route network [7].

The RSA reported in [7] can only target dynamical SOPOP, and here necessary modifications will be introduced, so that RSA can be applied to dynamical MOPOP. Due to limited space, here pseudocode details will be skipped, leaving focus on major modifications. As mentioned in [6], RSA is actually a decentralized, multi-agent based simulation model, and by simply modifying node micro behaviors, RSA can be flexibly extended to various POPs. Based on the RSA reported for dynamical SOPOP in [7], the following two key modifications are introduced in order to enable RSA to resolve dynamical MOPOP. In the new RSA, more than 1 ripple may be generated at a node, while differently, each node can have up to 1 ripple in the RSA of [7]. Basically, when a ripple reaches a node, if no existing PNDR of the node can dominate the ripple, then, the ripple will trigger a new ripple at the node, regardless how many ripples the node have generated. The termination criteria of RSA are also modified. In the new RSA, whether the ripple relay race should be terminated only depends on whether there exists any active ripple, regardless how many ripples have reached the destination, while in the RSA of [7], once a ripple has arrived at the destination, the ripple relay race should be terminated immediately.

Compared with the RSA for static MOPOP of Section 3, basically, the modified RSA for dynamical MOPOP has three major differences. First, the route network, i.e., the $[A_{z|0},C_{1,z|0},\ldots,C_{N\mathrm{Obj},z|0}]$ pair needs to be updated according to a given routing environmental dynamics at each simulated time unit during a single run of ripple relay race. Second, a ripple has to wait at a node if the node or a link ahead is inaccessible. Third, one might recall that in the RSA for static MOPOP of Section 3, updating the set of PNDRs does not need to delete any existing PNDR, while differently, in the RSA for dynamical MOPOP, there is such a need due to the introduction of waiting behavior. Because of waiting at a node, an earlier arrived PNDR may have exactly the same traveling time cost as a later arrived ripple. It is likely the later arrived ripple has better $f_2,\ldots,f_{N\mathrm{Obj}}$ values than the earlier arrived PNDR. So, the earlier arrived PNDR is dominated by the later arrived ripple, and therefore should be deleted. The flowchart of RSA for dynamical MOPOP is given in Fig. 7.

Similar to the way described in Sub-section 3.3, the RSA for one-to-one dynamical MOPOP can also be easily extended to one-to-all dynamical MOPOP. The optimality of RSA for dynamical MOPOP can be deduced by referring to the optimality proof of [7] and Theorem 1 in Sub-section 4.1. If the computational time for upgrading routing environmental information is not considered (e.g., because an external meteorological model can tell which part of route network will become inaccessible at what time under adverse weathers), then the computational complexity of RSA for dynamical MOPOP can be assessed as $O(N_{\mathrm{Obj}} \times N_{\mathrm{L}} \times N_{\mathrm{PP}}{}^2)$, exactly the same as that for static MOPOP.

Experimental Results

In this section, 4 sets of different experiments are conducted, in order to domenstrate the effectiveness of the reported RSA. In experiments, all methods used were coded by the same group of researchers, in the same hardware and software environments, and tested on the same sets of problems.

5.1. Results on one-to-one static bi-objective path optimization problems

Section 4 gives the theoretical proof about the optimality of the proposed RSA for static MOPOPs. Here experimental proofs will also be given. To this end, the new RSA is compared with the method of [44] and the Dijkstra-like method of [48], both of which are also capable of finding the complete Pareto front of static MOPOPs. Hereafter, the method of [44] is denoted as SOK, because it is based on single-objective k shortest paths, and the method of [48] as DL. For a better demonstrative effect, the new RSA is also compared with methods of AOF, NSGA-II and MOEA/D, which have no theoretical guarantee of finding complete Pareto front.

Although NSGA-II and MOEA/D are not the focus of this paper, it is still necessary to roughly explain some of their designs. In either NSGA-II or MOEA/D, crossover is not adopted, because crossing two path vectors can easily produce infeasible paths by causing a node to appear twice in a same path. Mutation is relatively straightforward, i.e., randomly select two non-successive nodes in a path, and if there is a link between them in the network, then delete those intermediate nodes between them in the path and make them successive. In a new generation of paths, some comes from mutating randomly selected paths in the previous generation, some are randomly produced by certain path initialization techniques, and some are copied from elite paths in the previous generation. In NSGA-II, which path will be chosen for mutation or elite copy and which path will be updated by initialization techniques largely depend on the Pareto-compliant ranking of each path among its generation. MOEA/D can be viewed as to simultaneously evolve solutions to a set of subproblems, which result from the decomposition of the original MOPOP. In this study, each subproblem is actually a single-objective path optimization problem defined by a weight sum of original objectives, and each subproblem is associated with a randomly given set of weights in order to evaluate the fitness of those paths to that subproblem. Mutation, initialization and elite copy in MOEA/D are conducted based on such scalar fitness values within each subproblem. Therefore, Pareto-compliant ranking is not necessary for MOEA/D, in other words, the computational burden of Pareto-compliant ranking is avoided in MOEA/D. Some preliminary experiments on adjusting pure GA parameters are conducted in order to set up NSGA-II and MOEA/D. Based on

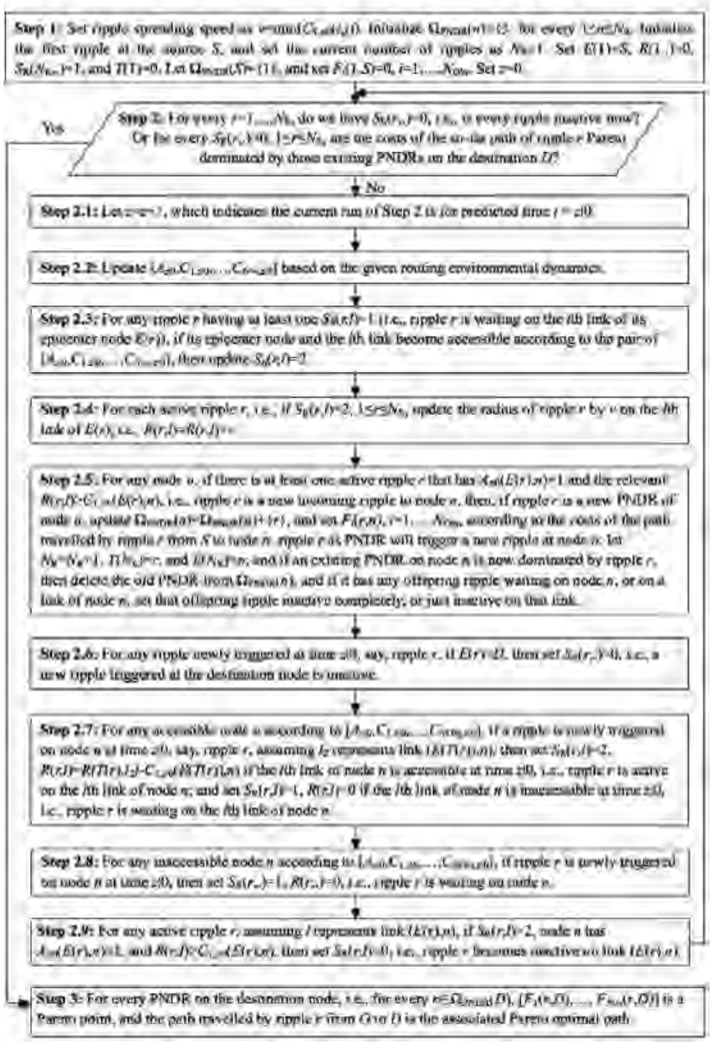

Step 1: Set ripple spreading speed as v=min(C_{1,t0}(i,j)). Initialize Ω_PNDR(n)=∅ for every 1≤n≤N_N. Initialize the first ripple at the source S, and set the current number of ripples as N_R=1. Set E(1)=S, R(1,.)=0, S_R(N_R,.)=1, and T(1)=0. Let Ω_PNDR(S)={1}, and set F_i(1,S)=0, i=1,...,N_Obj. Set z=0.
Yes
Step 2: For every r=1,...,N_R, do we have S_R(r,.)=0, i.e., is every ripple inactive now? Or for every S_R(r,.)≠0, 1≤r≤N_R, are the costs of the so-far path of ripple r Pareto dominated by those existing PNDRs on the destination D?
No
Step 2.1: Let z=z+1, which indicates the current run of Step 2 is for predicted time t = z∂.
Step 2.2: Update [A_{z∂},C_{1,z∂},...,C_{N_Obj,z∂}] based on the given routing environmental dynamics.
Step 2.3: For any ripple r having at least one S_R(r,l)=1 (i.e., ripple r is waiting on the lth link of its epicenter node E(r)), if its epicenter node and the lth link become accessible according to the pair of [A_{z∂},C_{1,z∂},...,C_{N_Obj,z∂}], then update S_R(r,l)=2.
Step 2.4: For each active ripple r, i.e., if S_R(r,l)=2, 1≤r≤N_R, update the radius of ripple r by v on the lth link of E(r), i.e., R(r,l)=R(r,l)+v.
Step 2.5: For any node n, if there is at least one active ripple r that has A_{z∂}(E(r),n)=1 and the relevant R(r,l)≥C_{1,z∂}(E(r),n), i.e., ripple r is a new incoming ripple to node n, then, if ripple r is a new PNDR of node n, update Ω_PNDR(n)=Ω_PNDR(n)+{r}, and set F_i(r,n), i=1,...,N_Obj, according to the costs of the path travelled by ripple r from S to node n. ripple r as PNDR will trigger a new ripple at node n: let N_R=N_R+1, T(N_R)=r, and E(N_R)=n; and if an existing PNDR on node n is now dominated by ripple r, then delete the old PNDR from Ω_PNDR(n), and if it has any offspring ripple waiting on node n, or on a link of node n, set that offspring ripple inactive completely, or just inactive on that link.
Step 2.6: For any ripple newly triggered at time z∂, say, ripple r, if E(r)=D, then set S_R(r,.)=0, i.e., a new ripple triggered at the destination node is inactive.
Step 2.7: For any accessible node n according to [A_{z∂},C_{1,z∂},...,C_{N_Obj,z∂}], if a ripple is newly triggered on node n at time z∂, say, ripple r, assuming l_T represents link (E(T(r)),n), then set S_R(r,l)=2, R(r,l)=R(T(r),l_T)-C_{1,z∂}(E(T(r)),n) if the lth link of node n is accessible at time z∂, i.e., ripple r is active on the lth link of node n; and set S_R(r,l)=1, R(r,l)=0 if the lth link of node n is inaccessible at time z∂, i.e., ripple r is waiting on the lth link of node n.
Step 2.8: For any inaccessible node n according to [A_{z∂},C_{1,z∂},...,C_{N_Obj,z∂}], if ripple r is newly triggered on node n at time z∂, then set S_R(r,.)=1, R(r,.)=0, i.e., ripple r is waiting on node n.
Step 2.9: For any active ripple r, assuming l represents link (E(r),n), if S_R(r,l)=2, node n has A_{z∂}(E(r),n)=1, and R(r,l)≥C_{1,z∂}(E(r),n), then set S_R(r,l)=0, i.e., ripple r becomes inactive on link (E(r),n).
Step 3: For every PNDR on the destination node, i.e., for every r∈Ω_PNDR(D), [F_1(r,D),..., F_{N_Obj}(r,D)] is a Pareto point, and the path travelled by ripple r from O to D is the associated Pareto optimal path.

Fig. 7. Flowchart of RSA for dynamical MOPOP.

such preliminary experiments, the mutation probability, the random initialization probability, the elitist probability, the population size and the number of evolving generations for both NSGA-II and MOEA/D in this sub-section are set up as 0.4, 0.5, 0.1, 400 and 400, respectively.

Here, the test setup of [44] is borrowed. Firstly, randomly disturb the locations of N_N nodes which are originally evenly distributed in a rectangular area defined by [-1000 1000 -1000 1000]; randomly choose a node to connect it to its closest neighboring nodes, and make sure every node has no more than 4 connections; then $C_1(i,j)$ is set as the result of dividing the physical straight line distance between nodes i and j by the ripple spreading speed; set $C_2(i,j)$ in the same way as in Reference [44], i.e.,

$$C_2(i,j) = \max(C_1(.,.)) \times \min(C_1(.,.))/C_1(i,j). \tag{18}$$

N_N is used to adjust the problem scale in the test, in this sub-section, N_N has three values, i.e., 25, 36 and 49, and for each N_N, 100 route networks are generated. For more details of the test setup, readers may refer to [44].

The main experimental results are given in Table 2, and Fig. 8 plots the results of a test case with N_N=25. In Table 2, CT is the average computational time (in second), N_{PPF} tells how many Pareto points a method has found, and R_{FCPF} gives the rate of a method for finding the complete Pareto front (i.e., divide the number of route networks where the method has found the complete Pareto front by the total number of route networks the method has been tested under a given N_N). Please note that, for each route network, either NSGA-II or MEA/D is run for 50 times; AOF is also run for 50 times, each time with a randomly generated set of weights for aggregate objective function; SOK, DL and the new RSA are run only once, respectively, as they can guarantee the finding of complete Pareto front (in other words, for a given route network, no matter how many times SOK, DL or the new RSA is run, the output is the same). Therefore, for NSGA-II, MOEA/D and AOF, Table 2 lists the average values of mean and SS (mean square error) for CT, N_{PPF} and R_{FCPF} (because of 50 runs for each of the 100 route networks), while for SOK, DL and the new RSA, just average CT, N_{PPF} and R_{FCPF} based on the 100 route networks. In Fig. 8, the first 9 subplots show all Pareto optimal paths of the test case; and the last subplot gives the Pareto fronts calculated by the 6 different methods. From Table 2 and Fig. 8, one can see that:

- SOK, DL and the new RSA proposed in this paper have exactly the same N_{PPF} and R_{FCPF} values in all test cases. As both SOK and DL have optimality guarantee of finding the complete Pareto front for static MOPOPs, Table 2 can serve as experimental proofs about the optimality of the new RSA.
- In terms of CT, both DL and the new RSA exhibit much better computational efficiencies than SOK, which will be further proven in the experiment of next subsection. Roughly speaking, DL is slower than the new RSA, and this is consistent with relevant theoretical analyses. According to [48], the computational

Table 2
Average Comparative Results of Bi-Objective POP

		NSGA-II		MOEA/D		AOF		SOK	DL	New RSA
		Mean	SS	Mean	SS	Mean	SS			
N_N=25	CT	36.29	4.93	32.02	4.76	**0.39**	0.02	0.54	0.49	0.47
	N_{PPF}	5.13	2.11	5.28	1.96	2.81	0.38	**7.67**	**7.67**	**7.67**
	R_{FCPF}	0.31	0.12	0.33	0.09	0.06	0.01	**1.00**	**1.00**	**1.00**
N_N=36	CT	58.70	8.18	51.78	10.27	**0.71**	0.05	4.63	1.16	0.73
	N_{PPF}	5.83	2.64	6.63	2.49	4.03	0.63	**10.09**	**10.09**	**10.09**
	R_{FCPF}	0.22	0.07	0.26	0.05	0.04	0.01	**1.00**	**1.00**	**1.00**
N_N=49	CT	107.38	14.23	94.75	16.20	**1.69**	0.17	33.15	2.97	1.85
	N_{PPF}	5.27	2.87	6.51	3.05	4.58	0.69	**15.44**	**15.44**	**15.44**
	R_{FCPF}	0.08	0.03	0.11	0.03	0.01	0.00	**1.00**	**1.00**	**1.00**

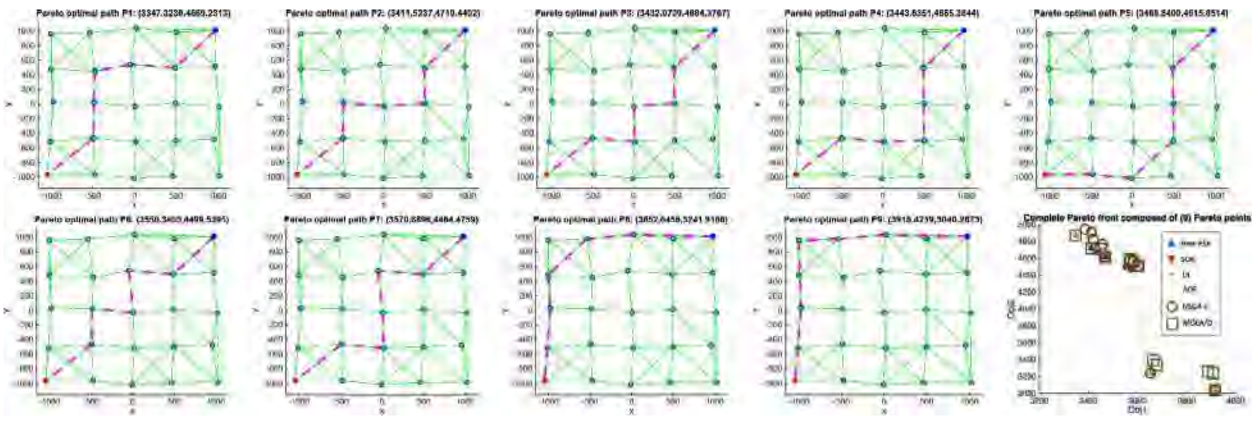


Fig. 8. Example of solutions by different methods in a test case of static MOPOP (NN=25).

complexity of DL is about O(N_{PP} × N_N × (N_L+N_N × logN_N)), i.e., O(N_{PP} × N_N × N_L+N_{PP} × N_N^2 × logN_N), while the new RSA O(N_{Obj} × N_L × N_{PP}^2). In this experiment, the value of the former is larger than the value of the latter, so, the new RSA runs faster than DL. Actually, the new RSA has similar CTs to the AOF method, which is the fastest.

- Table 2 also clearly demonstrates that either NSGA-II, MOEA/D or AOF often fails to find the complete Pareto fronts. In terms of N_{PPF}, the AOF method is the worst of all, as it cannot find those Pareto points in non-convex parts of a Pareto front. The reason why NSGA-II and MOEA/D often fail to find the complete Pareto front is because of the stochastic nature of NSGA-II and MOEA/D, which means there is no theoretical guarantee of optimality for even a single solution, let alone complete Pareto front. MOEA/D often runs faster and finds a little more Pareto points than NSGA-II. This is largely because the problem decomposition technique of MOEA/D leads to a better searching efficiency. However, in terms of CT, both NSGA-II and MOEA/D, due to their population-based evolutionary processes, are much worse than the other four methods.
- Because of the random features in NSGA-II, MOEA/D and AOF, the values of SS are not zero, which means for a given route network, different runs of either NSGA-II, MOEA/D or AOF may output different results. In terms of SS, AOF is more stable than NSGA-II and MOEA/D, this is probably because the only random feature of AOF is the random generation of weights, while there are much more random features in NSGA-II and MOEA/D.
- Please note that the performances of AOF, NSGA-II and MOEA/D are influenced not only by problem scale, but also by the values of some algorithm parameters. For example, if more sets of weights are used to combine N_{Obj} objectives in AOF, and a larger population and more generations to evolve in NSGA-II and MOEA/D, they might be able to find more Pareto points (however, AOF still cannot find Pareto points on non-convex parts of Pareto front, and either NSGA-II or MOEA/D will still output some false Pareto points), at the cost of increasing CT significantly. Differently, the performances of SOK, DL and the reported RSA here are much more stable, i.e., they can always find complete Pareto front, and their CTs are mainly dependent of problem characteristics, such as problem scale and network topology.
- It should be noted that, as mentioned in Section 1, a goal of this study is to further mature and enrich RSA as a third stream method distinguished from both classical deterministic methods (the first stream) and stochastic methods (the second stream). Therefore, the experiment here is mainly used to demonstrate the merits and demerits of three method streams (rather than to claim advantages against certain specific algorithms). If further adjust/modify AOF, NSGA-II and MOEA/D, or replace them with more recent methods, the experimental results of stream 1 and stream 2 might be improved, but the conclusion will not change that stream 1 is not flexible (e.g., AOF cannot find any Pareto points on non-convex parts of Pareto front) and stream 2 cannot guarantee optimality (e.g., either NSGA-II or MOEA/D often outputs false Pareto points), while RSA, representing stream 3, can always find all Pareto points and never output any false Pareto points.

5.2. Test on one-to-one static multi-objective path optimization problems with NObj>2 objectives

One might argue that the problem scale in the experiment of previous subsection is small, i.e., all test cases only have N_{Obj}=2 objectives, the largest networks have just N_N=49 nodes and N_L<200 links, and the most complicated Pareto front has just about 30 Pareto points. Therefore, one may ask: how about testing on some MOPOPs with more objectives, more nodes, and more Pareto points? Usually, the larger the values of N_{Obj}, N_N, N_L and N_{PP}, the more complicated the MOPOP is. Basically, with large settings of N_{Obj}, N_N, N_L and N_{PP}, if still use the completely random network generator as employed in Sub-section 5.1, it will be very difficult, if not impossible, to verify whether or not the re-

Table 3
Average Comparative Results of MOPOP with N_{Obj}>2

(N_N=100,N_L=400, and N_{PP}=5N_{Obj})		NSGA-II		MOEA/D		AOF		SOK	DL	New RSA
		Mean	SS	Mean	SS	Mean	SS			
N_{Obj}=3	CT	854.86	145.27	817.06	143.89	**0.71**	0.15	9.05	1.13	0.76
	N_{PPF}	6.62	3.13	6.95	3.02	4.71	1.03	**12**	**12**	**12**
	R_{FCPF}	0.33	0.08	0.36	0.07	0.00	0.00	**1.00**	**1.00**	**1.00**
N_{Obj}=4	CT	857.01	141.56	815.58	148.17	**0.85**	0.19	11.36	2.23	1.05
	N_{PPF}	9.03	3.25	9.76	3.09	6.28	1.28	**16**	**16**	**16**
	R_{FCPF}	0.30	0.06	0.31	0.06	0.00	0.00	**1.00**	**1.00**	**1.00**
N_{Obj}=6	CT	864.17	142.96	825.42	146.27	**1.94**	0.34	14.27	4.86	3.07
	N_{PPF}	13.08	5.24	15.17	6.76	7.83	1.72	**24**	**24**	**24**
	R_{FCPF}	0.21	0.07	0.24	0.06	0.00	0.00	**1.00**	**1.00**	**1.00**
N_{Obj}=10	CT	883.94	155.07	837.36	167.26	**4.18**	0.58	20.33	9.17	6.92
	N_{PPF}	17.93	9.27	18.78	6.85	9.38	1.52	**40**	**40**	**40**
	R_{FCPF}	0.18	0.07	0.18	0.08	0.00	0.00	**1.00**	**1.00**	**1.00**
N_{Obj}=20	CT	915.85	164.06	863.79	171.54	**6.71**	0.91	28.69	12.03	11.85
	N_{PPF}	41.72	13.29	44.05	13.70	12.49	1.44	**80**	**80**	**80**
	R_{FCPF}	0.16	0.08	0.18	0.06	0.00	0.00	**1.00**	**1.00**	**1.00**
N_{Obj}=40	CT	981.27	173.46	895.48	166.03	**13.28**	1.38	50.26	15.72	19.09
	N_{PPF}	58.34	16.22	63.06	14.95	14.80	1.62	**160**	**160**	**160**
	R_{FCPF}	0.12	0.04	0.13	0.05	0.00	0.00	**1.00**	**1.00**	**1.00**

ported RSA has found the complete Pareto front, because (i) it is almost impossible to deduce manually the complete Pareto front to a MOPOP randomly generated in Sub-section 5.1 with large N_{Obj}, N_N, N_L and N_{PP}, and (ii) the computational burden of relevant existing methods, such as SOK, soars up dramatically as N_{Obj}, N_N, N_L and N_{PP} increase.

To be able to test the reported RSA under large N_{Obj}, N_N, N_L and N_{PP}, a special MOPOP generator is needed, so that one can manually deduce the complete Pareto front of generated MOPOP test case, no matter how large N_{Obj}, N_N, N_L and N_{PP} are. Fortunately, the benchmark test problem toolkit reported in [41] can generate random MOPOPs of different problem scales with any pre-given Pareto fronts. Here, with the MOPOP toolkit of [41] and its Rule 1, 600 route networks are randomly generated with settings of $N_{Obj} \in [3, 4, 6, 10, 20, 40]$, N_N=100, N_L=400, and N_{PP}=4N_{Obj} (i.e., 100 networks for each set of N_{Obj}, N_N, N_L and N_{PP} values). Then, test the reported RSA and other methods on these 600 route networks. The average test results are given in Table 3, which shows:

- With no surprise (partially because of the theoretical proof in Section 4), the new RSA has found complete Pareto fronts in all of these 600 test cases. And the new RSA also finds complete Pareto front very fast when compared with other methods.
- Both SOK and DL have also found complete Pareto fronts in all test cases. SOK runs slower than both DL and the reported RSA. DL runs sometimes slower than RSA, and sometimes faster than RSA, largely depending on the settings of N_{Obj}, N_N, N_L and N_{PP} values (actually in this experiment, depending on the value of N_{Obj}, as N_N and N_L are fixed, and N_{PP} always equals to 4N_{Obj}). For instance, DL runs clearly slower than RSA when N_{Obj}=3, 4, 6, and 10, while faster than RSA when N_{Obj}=40. This might imply that DL could be a better choice when dealing with many objectives (under certain values of N_N, N_L and N_{PP}). However, in most real-world applications, there are usually only a few major objectives to be considered in a MOOP. Therefore, RSA still has a good application potential.
- Compared with those results in Table I, one might notice the CT of SOK does not increase as fast as that of the new RSA. This is mainly because, for a route network generated by the MOPOP toolkit of [41] under its Rule 1, SOK only needs to generate no more than 1 ripple at each node, while for a more random network used in Sub-section 5.1, SOK often has to generate a lot of ripples at each node, in order to guarantee optimality. As shown in [6], generating multiple ripples at each node will cause the computational burden increase exponentially. Although the new RSA also has no more than 1 PNDR because of the purpose-designed model here (in Sub-section 5.1, each node often has multiple PNDRs), it needs to keep comparing N_{Obj} objective values between ripples at each node, and this causes computational burden increase dramatically. There is no need of comparing N_{Obj} objective values between ripples at each node in each single run of single-objective ripple relay race employed in SOK, although at least N_{Obj} runs will be conducted to find a complete Pareto front. Therefore, based on the MOPOP toolkit under its Rule 1, SOK looks not too slow compared with the new RSA. Similarly, the CT of DL does not increase as fast in Table II as in Table I, and this is because, for a route network generated by the MOPOP toolkit of [41] under its Rule 1, DL often only needs to label a node once in terms of Pareto non-dominance, while for a network in Sub-section 5.1, DL often needs to label a node for quite a few or even many times.
- The AOF method has never found any complete Pareto front, because in these many-objective test cases, no Pareto front is convex. Actually, in terms of N_{PPF} and R_{FCPF}, AOF is the worst of all methods. Here, 2N_{PP} sets of weights are used to combine N_{Obj} objectives. Each set of weights can lead to a Pareto optimal path, but most sets lead to the same paths.
- In most test cases, either NSGA-II or MOEA/D has also failed to find complete Pareto front. In general, MOEA/D finds more Pareto points than NSGA-II, which means the problem decomposition technique of MOEA/D is helpful. NSGA-II is the slowest method of all. Here, please note that for whichever value of N_{Obj}, both NSGA-II and MOEA/D employ the same population size (i.e., 1000 individuals in a generation) and generation number (i.e., 600 generations to evolve), and therefore, the CT of NSGA-II or MOEA/D does not change significantly over N_{Obj}.
- For the same reason as discussed in Subsection 5.A, after analyzing the SS values of NSGA-II, MOEA/D and AOF based on their 50 runs for each of the 600 route networks, AOF is more stable than NSGA-II and MOEA/D. In Table 3, there is no SS value given for SOK, DL or the new RSA, because they can always find the complete Pareto front, which means, if given, their SS values of N_{PPF} and R_{FCPF} would always be zero.

5.3. Test on a complicated one-to-all static 3-objective path optimization problem

Now it should be safe to say that the results in Sub-sections 5.1 and 5.2 have satisfactorily demonstrated the reported RSA can find complete Pareto front for one-to-one static MOPOP with optimality guaranteed. Here is a further test on some three-objective one-to-all static problems

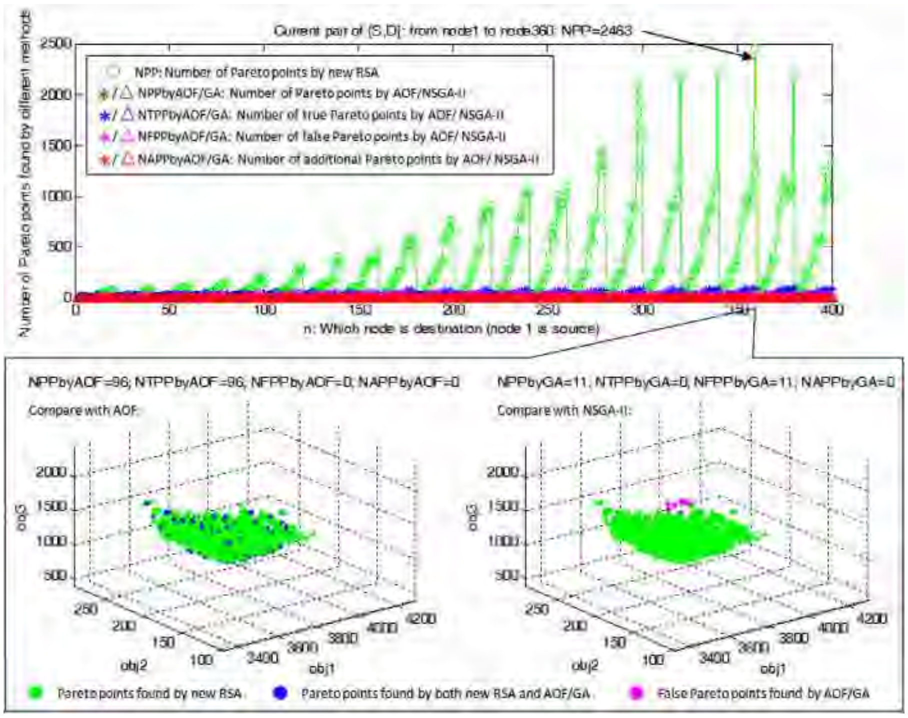


Fig. 9. An example of one-to-all static MOPOP (NN=400 and NL=1600).

with N_N=400, N_L=1600, in order to show the efficiency of the reported RSA. Here the network model of [41] is not used, as it might still look simple, no matter how large N_{Obj}, N_N, N_L and N_{PP} are. Instead, the much more random model of Sub-section 5.1 is used to generate route network.

In this experiment, only the NSGA-II and the AOF method are used for comparative purposes. This sub-section randomly generates 100 route networks, where each link has 3 costs for the calculation of 3 objectives. In each network, node 1 (the node at the left-bottom of network) is the source, and "one-to-all" means it must find the complete Pareto front when node 2 is destination, the complete Pareto front when node 3 is destination, ..., and the complete Pareto front when node 400 is destination. In other words, there are totally 399 Pareto fronts for each single one-to-all problem with N_N=400. This means, for each route network, it has to run NSGA-II or AOF for 399 times to resolve this one-to-all problem. Differently, the new RSA only needs to run for just once, i.e., only a single run of ripple relay race needs to be carried out on the route network, and then all the 399 complete Pareto fronts are available at hand.

Fig. 9 gives some main results of a single test network in this experiment. In the top subplot of Fig. 9, the numbers of Pareto points found by different methods are given when different node in the route network is chosen as destination (node 1 is always source). It clearly shows that the NSGA-II and the AOF method usually only find a very small proportion of true Pareto points when a complete Pareto front is big. In the 399 complete Pareto fronts of this test network, the most complicated Pareto front has 2463 Pareto points in total (when node 360 is destination), as calculated by the new RSA. The left-bottom and the right-bottom subplots of Fig. 9 compare the Pareto points found by the AOF method and by the NSGA-II, respectively, with those by the new RSA for the case where node 360 is destination (the most complicated case). Actually, although the NSGA-II in this experiment has a population size of 1000 and a generation number of 600, which are not small, it fails to find any true Pareto point in this case.

Table 4 gives some average results of 100 route networks. CT stands for computational time (in second) consumed by a method to resolve a one-to-all MOPOP; N_{PPF} the number of Pareto points found (no matter true or false) by a method for a one-to-all MOPOP; N_{TPPF}/N_{FPPF} the number of true/false Pareto points found by a method for a one-to-all MOPOP; $N_{ATPPFbyGA/AOF}$ the number of additional true Pareto points found by NSGA-II or AOF (i.e., true Pareto points that RSA fails to find); R_{FSCPF} the ratio indicating for how many of those 399 nodes (except node 1) in a single route network, a method has found complete Pareto front; R_{FACPF} the ratio indicating for how many of those 100 route networks, a method has found all complete Pareto fronts. From Table 4, one may have the following observations.

- Regarding computational efficiency, the new RSA takes just about 500 seconds on average to find all of the 399 Pareto fronts by a single run. The NSGA-II usually takes hours but ends up with many false Pareto points. The AOF method (with $2N_N$=800 sets of weights) takes about 400 seconds, but as shown in Table 4, it finds a very small proportion of true Pareto points.
- NSGA-II has a larger N_{PPF} than AOF, but it finds many false Pareto points, while AOF never outputs any false Pareto points. Actually, NSGA-II finds much less true Pareto points than AOF does.
- Since SOK is not adopted here due to its poor scalability, $N_{ATPPFbyGA/AOF}$ is used as an indirect/implicit indicator to verify whether RSA has failed to find complete Pareto fronts. Every time when NSGA-II or AOF has found a Pareto point that RSA has failed to find and is not dominated by any Pareto points found by RSA, $N_{ATPPFbyGA/AOF}$ is increased by 1. Thus, after a ripple relay race stops for a route network, if $N_{ATPPFbyGA/AOF}$>0, one knows RSA fails. Fortunately, as expected, this does not happen, because always $N_{ATPPFbyGA/AOF}$=0 in the experiment.
- When a node very close to node 1 is chosen as destination (e.g., a node that has a direct link to node 1), the associated complete Pareto front is usually simple with just a few Pareto points. So, either NSGA-II or AOF stands a chance to find such a complete Pareto front. This explains why N_{FSCPF}>0 for both NSGA-II and AOF.
- However, either NSGA-II or AOF fails to find all complete Pareto fronts for any of those 100 one-to-all MOPOPs in the experiment. This is why N_{FACPF}=0 for both NSGA-II and AOF.
- One might wonder: compared with the N_{PPF} of RSA, NSGA-II finds a smaller proportion of true Pareto points than AOF does, then, why does NSGA-II find more complete Pareto fronts than AOF, i.e., why

Table 4
Average Comparative Results of One-to-All 3-Objective POPs

(N_N=400 and N_L=1600)	NSGA-II		AOF		New RSA
	Mean	SS	Mean	SS	
CT	18707.52	3506.28	**417.55**	34.17	509.62
N_{PPF}	9785.23	1021.63	6471.08	482.27	**191183.26**
N_{TPPF}	3038.37	363.41	6471.08	482.27	**191183.26**
N_{FPPF}	6746.86	749.62	**0.00**	0.00	**0.00**
$N_{ATPPFbyGA/AOF}$	0.00	0.00	0.00	0.00	**N.A.**
R_{FSCPF}	0.03	0.01	0.02	0.00	**1.00**
R_{FACPF}	0.00	0.00	0.00	0.00	**1.00**

does NSGA-II have a larger N_{FSCPF} than AOF? Again, this is largely because AOF can only find Pareto points on convex parts of Pareto front, but in this experiment, most Pareto fronts (even many of those Pareto fronts between node 1 and its nearby nodes) are not convex.

- One might also wonder: NSGA-II only finds less than 2% true Pareto points, then, why does it find about 3% complete Pareto fronts. This is because the number of Pareto points is significantly uneven between Pareto fronts. As illustrated in Fig. 9, a complicated Pareto front may have over 2000 Pareto points, while a simple Pareto front only has less than 10 Pareto points. So, finding a large proportion of complete Pareto fronts does not necessarily mean finding a large proportion of true Pareto points.
- Again, according to those SS values, the random feature in AOF (i.e., randomly generating weights set for aggregate objective function) has relatively small influence on method performance than those random features in NSGA-II do. The new RSA has no SS values, because it always guarantees finding all complete Pareto fronts for a one-to-all MOPOP.
- Again, it should be noted that, if increase the number of weight sets for AOF, or replace NSGA-II with more recent more powerful multi-objective evolutionary algorithms, then those associated experimental results may be improved, but AOF still cannot find Pareto points on non-convex parts of Pareto front, and evolutionary methods will still output some false Pareto points.

5.4. Test on a one-to-one dyanimcal 2-objective path optimization problem

There lacks test case or method for dynamical MOPOP in existing literatures, so, it is very difficult, if not impossible, to make comparisons. In order to make dynamical MOPOP digestible to readers, as well as making manual analysis on the complete Pareto front possible, just a simple test case of dynamical MOPOP is given in Fig. 10. The dynamical route network in Fig. 10 has N_N=9 nodes and N_L=12 links. Each link has N_{Obj}=2 costs: $C_{1,z|0}(i,j)$ is the traveling time to pass the link, and $C_{2,z|0}(i,j)$ is something like service charge. In this experiment, for the sake of simple demonstration, $C_{1,z|0}(i,j)$=1 for all links at any time $z|0$, i.e, $C_{1,z|0}$ is constant and will not change over time. Therefore, $C_{1,z|0}(i,j)$ is not given in Fig. 10 (otherwise, Fig. 10 would look too busy). Link accessibility can change from time to time (i.e., $A_{z|0}(i,j)$ can change between 1 and 0 over simulated time $z|0$). For instance, the link between nodes 2 and 5 is not passible until time 3|0, so, $A_{z|0}(2,5)$=0 for z=1,2, and then $A_{z|0}(2,5)$=1 for z=3,…,7. $C_{2,z|0}(i,j)$ also changes over time, and the data set of $C_{2,z|0}(i,j)$ is given in the first sub-plot of Fig. 10. For instance, the link between nodes 2 and 5 has a $C_{2,z|0}$ data set of "(2,3);(1,∞)", so, $C_{2,z|0}(2,5)$=2 for z=1,2,3, and then $C_{2,z|0}(2,5)$=1 for z>3. The link between nodes 4 and D has a $C_{2,z|0}$ data set of "(20,∞)", so, $C_{2,z|0}(4,D)$=20 for all times. Other sub-plots of Fig. 10 only show instantaneous values of $C_{2,z|0}$. As time-varying $A_{z|0}(i,j)$ is used in the calculation of objective f_1, and time-varying $C_{2,z|0}(i,j)$ used in in the calculation of objective f_2, clearly, this is a dynamical MOPOP.

From Fig. 10, one can also see how the route network changes simultaneously as the ripple relay race of RSA goes on, and how 4 Pareto optimal paths are found out one by one during the ripple relay race. All possible paths between node O and node D, as well as the complete Pareto front of the test problem are given in Fig. 11. By analyzing Fig. 10 and Fig. 11, one can clearly see that the RSA reported in this paper is effective and correct to resolve dynamical MOPOP.

In Fig. 10, at predicted time 1|0, node 5 and links (2,5), (4,5) and (5,7) are inaccessible. The initial ripple R1 spreads from the origin, i.e., node *O*, and reaches node 1 and node 3. Since there is no PNDR at either node 1 or node 3, R1 becomes their first PNDR, and therefore triggers new ripple R2 at node 1 and R3 at node 3, respectively.

At predicted time 2|0, the route network changes a little bit as link (6,D) becomes inaccessible. R2 spreads and reaches node 2. Since there is no PNDR at node 2, R2 becomes the first PNDR at node 2, and triggers new ripple R4. R3 arrives at node 6, becomes the first PNDR at node 6, and triggers new ripple R6 at node 6. Both R2 and R3 arrives at node 4. The so-far path to node 4 travelled by R3 has objective values [2,2] while the so-far path to node 4 travelled by R2 has objective values [2,13]. This means R3 dominates R2 at node 4, so, R3 becomes a PNDR of node 4, and triggers new ripple R5 at node 4.

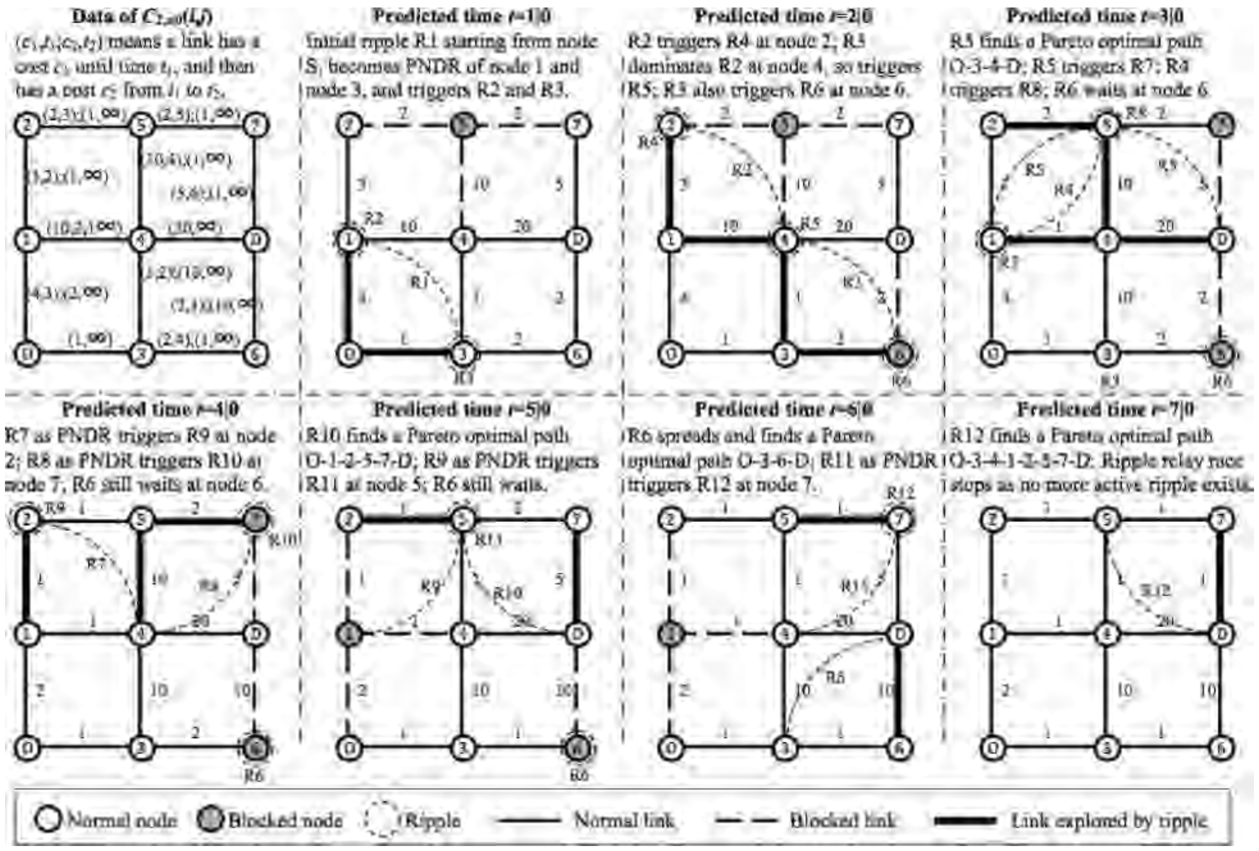
At predicted time 3|0, the route network changes further as node 5 and its associated 3 links become accessible, link (7,D) becomes inaccessible, and besides, link costs $C_{2,z|0}(1,2)$ and $C_{2,z|0}(3,4)$ also change. R5 reaches the destination D, and finds a Pareto optimal path O-3-4-D. R5 also reaches node 1. Node 1 already has a PNDR, i.e., R1, which had objective values [1,4] when reaching node 1 at time 1|0. R5 has objective values [3,3] when reaching node 1 at time 3|0. So, R5 is not dominated by R1 at node 1. So, R5 becomes the second PNDR at node 1, and triggers new ripple R7 at node 1. Both R4 and R5 reach node 5. Because R4 has objective values [3,9] and R5 has [3,12] when reaching node 5, R5 is dominated by R4 at node 5. Therefore, R4 becomes the first PNDR at node 5, and triggers new ripple R8 at node 5. Because link (6,D) is inaccessible, R6 holds and waits at node 6.

At predicted time 4|0, some link costs change. R7 arrives at node 2, and it is not dominated by the first PNDR of node 2, so, R7 is the second PNDR at node 2, and triggers new ripple R9 at node 2. R8 reaches node 7, becomes the first PNDR of node 7, and triggers new ripple R10 at node 7. R5 also arrives at node 4, but it is dominated by the existing PNDR of node 4, so, it does not trigger a new ripple at node 4. R6 keeps waiting at node 6.

At predicted time 5|0, the route network continues changing in terms of topology and link costs. R10 arrives at node D, identifying a Pareto optimal path O-1-2-5-7-D. R9 arrives at node 5, and it is not dominated by any existing PNDR of node 5, so, R9 is a new PNDR at node 5, and triggers new ripple R11 at node 5. R6 keeps waiting at node 6.

At predicted time 6|0, the route network keeps changing in terms of topology and link costs. In particular, link (6,D) becomes accessible, so R6 starts to spread and then arrives at D. Since R6 is not dominated by any existing PNDR of D, R6 finds a new Pareto optimal path O-3-6-D. R11 reaches node 7 and becomes a new PNDR of node 7, so, R11 triggers new ripple R12 at node 7.

At predicted time 7|0, both network topology and link costs continue changing. R12 reaches node D. Since R12 is not dominated by any existing PNDR of node D, R12 finds a new Pareto optimal path O-3-4-1-2-5-7-D. Then, there is no longer any active ripple. So, the ripple relay


Data of $C_{2,s0}(i,j)$
$(c_1,t_1);(c_2,t_2)$ means a link has a cost c_1 until time t_1, and then has a cost c_2 from t_1 to t_2.

Predicted time t=1|0
Initial ripple R1 starting from node S, becomes PNDR of node 1 and node 3, and triggers R2 and R3.

Predicted time t=2|0
R2 triggers R4 at node 2; R3 dominates R2 at node 4, so triggers R5; R3 also triggers R6 at node 6.

Predicted time t=3|0
R3 finds a Pareto optimal path O-3-4-D; R5 triggers R7; R4 triggers R8; R6 waits at node 6.

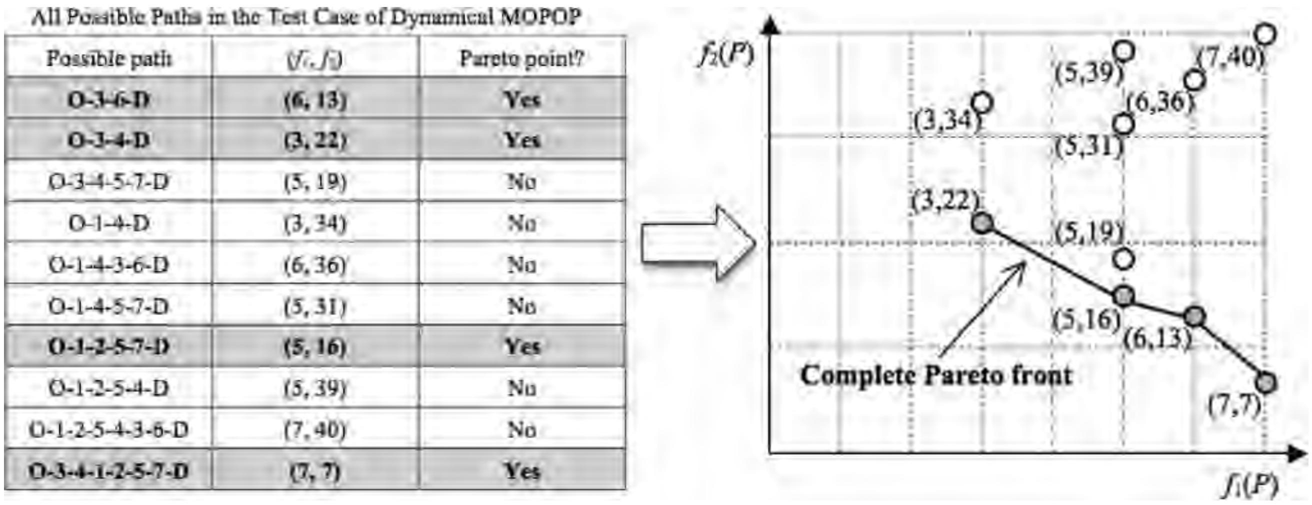
Predicted time t=4|0
R7 as PNDR triggers R9 at node 2; R8 as PNDR triggers R10 at node 7; R6 still waits at node 6.

Predicted time t=5|0
R10 finds a Pareto optimal path O-1-2-5-7-D; R9 as PNDR triggers R11 at node 5; R6 still waits.

Predicted time t=6|0
R6 spreads and finds a Pareto optimal path O-3-6-D; R11 as PNDR triggers R12 at node 7.

Predicted time t=7|0
R12 finds a Pareto optimal path O-3-4-1-2-5-7-D; Ripple relay race stops as no more active ripple exists.

Fig. 10. The test case of RSA for one-to-one dynamical MOPOP.


All Possible Paths in the Test Case of Dynamical MOPOP

Possible path	(f_1, f_2)	Pareto point?
O-3-6-D	**(6, 13)**	**Yes**
O-3-4-D	**(3, 22)**	**Yes**
O-3-4-5-7-D	(5, 19)	No
O-1-4-D	(3, 34)	No
O-1-4-3-6-D	(6, 36)	No
O-1-4-5-7-D	(5, 31)	No
O-1-2-5-7-D	**(5, 16)**	**Yes**
O-1-2-5-4-D	(5, 39)	No
O-1-2-5-4-3-6-D	(7, 40)	No
O-3-4-1-2-5-7-D	**(7, 7)**	**Yes**

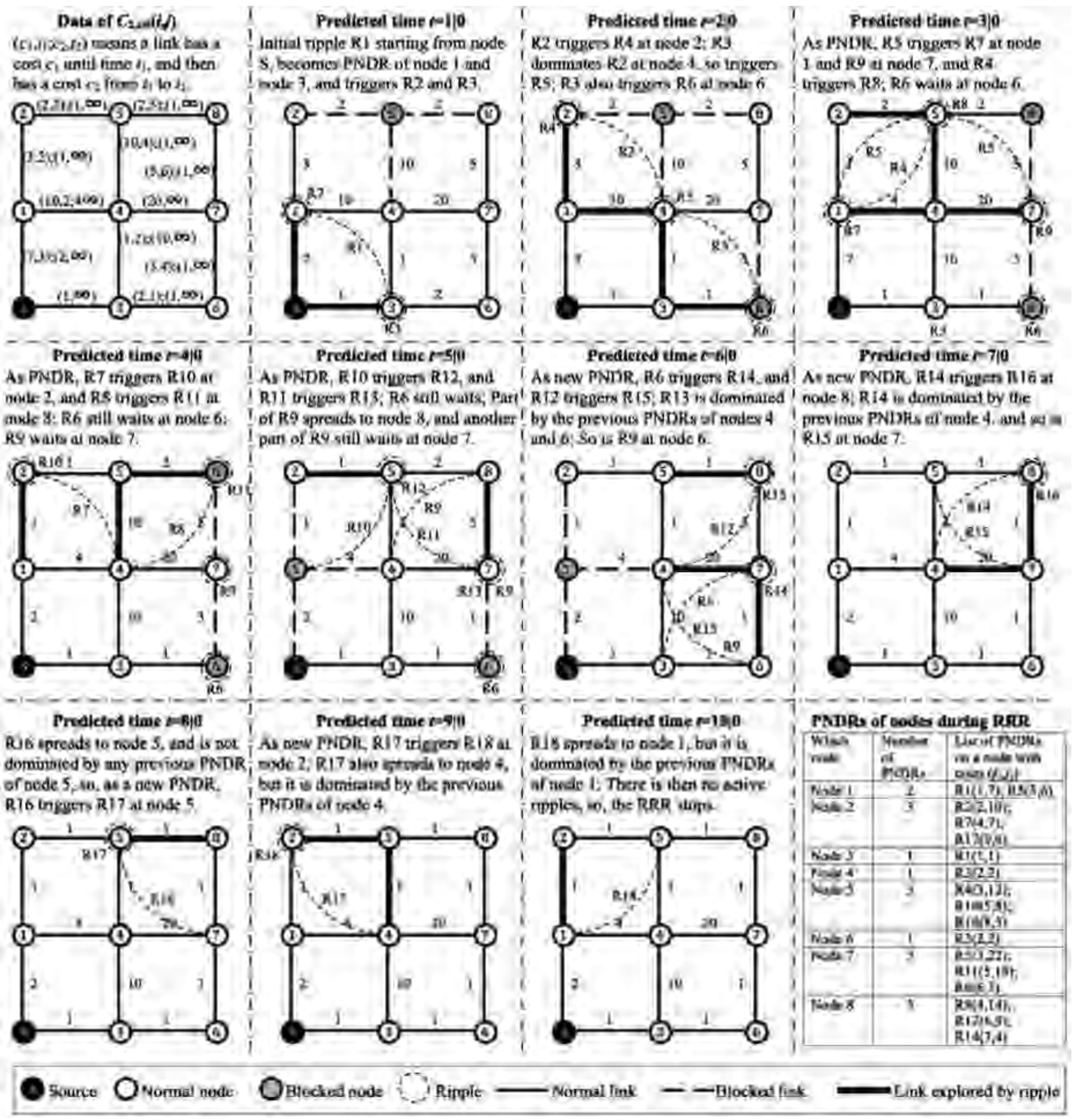
Fig. 11. Complete Pareto front of the test case in Fig. 10.

race stops, and all Pareto optimal paths between nodes O and D in the given dynamical route network have been found out successfully by the reported RSA.

5.5. Test on a one-to-all dyanimcal 2-objective path optimization problem

As emphasized through this paper, thanks to its multi-agent nature, RSA, without losing its optimality guarantee, is more flexible than classical deterministic methods in terms of extending to various complex problems. The test on one-to-one dynamical MOPOP in the previous subsection has already shown a preliminary proof about the modification flexibility of RSA, as no existing method can guarantee to find the complete Pareto front for such one-to-one dynamical MOPOP. This subsection will test the merits of RSA in an even harder problem, i.e., one-to-all dynamical MOPOP, which demands to find all Pareto fronts from a given source node to all other nodes in a dynamical route network. As explained in Section 4.C, it does not need to repeat running the one-to-one-dynamical-MOPOP-oriented RSA for N_N-1 times, but just modify it simply by changing the termination condition of ripple relay race, and then, still a single run of modified RSA can guarantee to find all Pareto fronts for a dynamical one-to-all MOPOP.

Because, to our best knowledge, one-to-all dynamical MOPOP is a problem never discussed in any existing literatures, here still a simple test case is used, so that readers may manually check the correctness of


Data of $C_{2\times4}(i,j)$
$(c_1,t_1)(c_2,t_2)$ means a link has a cost c_1 until time t_1, and then has a cost c_2 from t_1 to t_2.
Predicted time t=1|0
Initial ripple R1 starting from node S, becomes PNDR of node 1 and node 3, and triggers R2 and R3.
Predicted time t=2|0
R2 triggers R4 at node 2; R3 dominates R2 at node 4, so triggers R5; R3 also triggers R6 at node 6.
Predicted time t=3|0
As PNDR, R5 triggers R7 at node 1 and R9 at node 7, and R4 triggers R8; R6 waits at node 6.
Predicted time t=4|0
As PNDR, R7 triggers R10 at node 2, and R8 triggers R11 at node 8; R6 still waits at node 6; R9 waits at node 7.
Predicted time t=5|0
As PNDR, R10 triggers R12, and R11 triggers R13; R6 still waits; Part of R9 spreads to node 8, and another part of R9 still waits at node 7.
Predicted time t=6|0
As new PNDR, R6 triggers R14, and R12 triggers R15; R13 is dominated by the previous PNDRs of nodes 4 and 6; So is R9 at node 6.
Predicted time t=7|0
As new PNDR, R14 triggers R16 at node 8; R14 is dominated by the previous PNDRs of node 4, and so is R15 at node 7.
Predicted time t=8|0
R16 spreads to node 5, and is not dominated by any previous PNDR of node 5, so, as a new PNDR, R16 triggers R17 at node 5.
Predicted time t=9|0
As new PNDR, R17 triggers R18 at node 2; R17 also spreads to node 4, but it is dominated by the previous PNDRs of node 4.
Predicted time t=10|0
R18 spreads to node 1, but it is dominated by the previous PNDRs of node 1. There is then no active ripples, so, the RRR stops.
PNDRs of nodes during RRR
● Source
○ Normal node
◉ Blocked node
Ripple
Normal link
Blocked link
Link explored by ripple

Which node	Number of PNDRs	List of PNDRs on a node with costs (f_1,f_2)
Node 1	2	R1(1,7), R5(3,6)
Node 2	3	R2(2,10); R7(4,7); R17(9,6)
Node 3	1	R1(1,1)
Node 4	1	R3(2,2)
Node 5	3	R4(3,12); R10(5,8); R16(8,5)
Node 6	1	R3(2,2)
Node 7	3	R5(3,22); R11(5,19); R9(6,3)
Node 8	3	R8(4,14); R12(6,5); R14(7,4)

Fig. 12. A test case of RSA for a one-to-all dynamical MOPOP.

RSA. Otherwise, if use random, complex test cases just like in Subsections 5.A and 5.B, then readers will have no idea whether and how RSA works correctly in one-to-all dynamical MOPOP.

The left-top subplot of Fig. 12 defines a one-to-all dynamical 2-objective MOPOP, the other subplots of Fig. 12 show how a single run of RSA can find all Pareto fronts, and the right-bottom subplot of Fig. 12 list the PNDRs on each node during the single ripple relay race. Fig. 13 gives all possible paths in the test case of Fig. 12, and all Pareto fronts are also plotted. From Fig. 12 and Fig. 13, one can see clearly that RSA can correctly find all Pareto fronts in a dynamical route network still by just a single run of ripple-relay race.

Dynamical MOPOP has great application potentials. For instance, in an adverse weather scenario (e.g., rainstorm, typhoon, blizzard), road conditions in network are time varying due to the dynamical weather developing progress. Although the CEPO method of [7] can find single-objective optimal paths in such a dynamical routing environment, it only considers how to minimize traveling time. However, in such an adverse weather scenario, how to minimize traveling risk is at least of the same importance. For emergence evacuation in disaster events (such as fire, earthquake, flooding), traveling risk even has a higher priority. Apparently, a short route segment does not necessarily correspond to a small risk. The test networks in these two sub-sections can well model such time-risk dynamical problems, and the reported RSA can achieve optimal results no other existing methods can guarantee.

Conclusions and Future Work

Ripple-spreading algorithm (RSA) is a relatively new method in computational intelligence. Existing RSAs are all focused on single-objective optimization problems (SOOPs). For the first time, this paper proposes an RSA which itself truly targets at multi-objective path optimization problems (MOPOPs). The new RSA can calculate the complete Pareto front by running a ripple relay race. The new RSA is an agent-based bottom-up simulation model, and by defining micro agent behavior, i.e., how a node will generate a new ripple according to the Pareto dominance state of an incoming ripple, the complete Pareto front will emerge

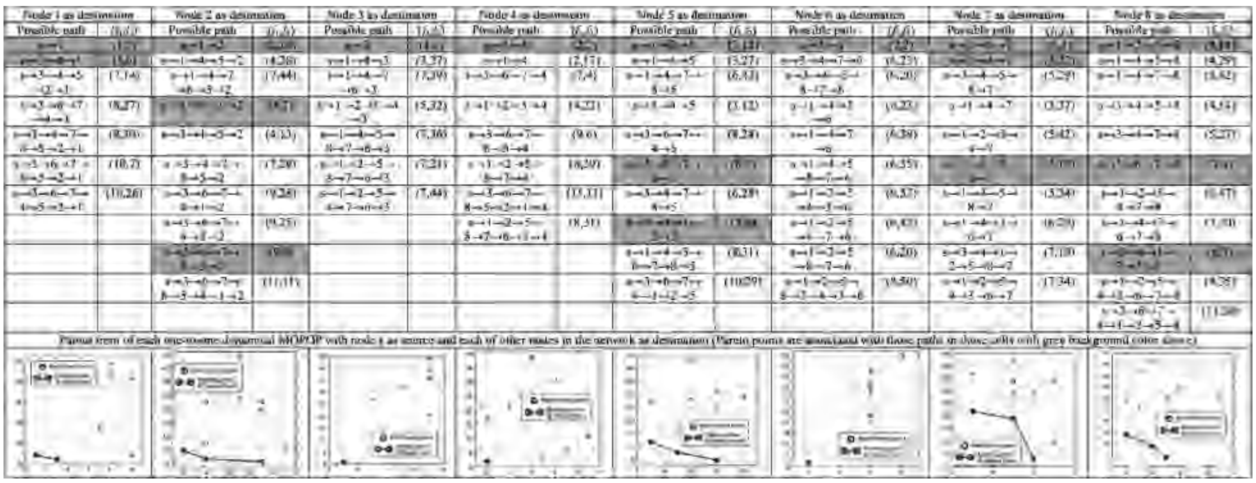


Fig. 13. Theoretical results and complete Pareto fronts of the one-to-all dynamical MOPOP in Fig. 13.

at the macro level of ripple relay race with a guarantee of optimality. Different from some other methods that can find the complete Pareto front for static MOPOPs, the proposed RSA can easily deal with more complicated MOPOPs, such as with dynamical networks. Experimental results illustrate the effectiveness and efficiency of the proposed RSA for MOPOPs. Because of its capability of finding complete Pareto front, the new RSA can serve as a benchmark method for studying MOOPs, and the resolved MOPOPs can be used as benchmark problems to assess the performance of other methods for discrete MOOPs, such as genetic algorithm and particle swarm optimization. Future attentions may be put on conducting more theoretical analyses (e.g., space complexity), applying the reported RSA to real-world transportation problems which can be converted to MOPOPs, and making comparisons and combinations with more recent MOOP methods in specific applications, particularly with special-purpose-designed evolutionary operations and problem decomposition strategies. Because of the multi-agent nature of RSA, it is worth exploring how to design a decentralized parallel coding architecture of RSA. Effort should also be made to optimize the codes of RSA and then make them publicly available.

Author Statement

The authors would like to thank the editor and the reviewers very much for their extensive and helpful comments for revising the manuscript!

Declaration of Competing Interest

The authors declare no conflict of interest.

Acknowledgments

This work was supported by the National Natural Science Foundation of China (Grant No.61472041), and some preliminary results were presented at the 2018 IEEE Symposium Series on Computational Intelligence (SSCI 2018), 18 to 21 Nov 2018, Bengaluru, India.

References

[1] M.T. Goodrich, Algorithm Design: Foundations, Anal., and Internet Examples, Wiley, 2001.

[2] S. Russell, P. Norvig, Artificial Intelligence: A Modern Approach, 3rd ed, Prentice Hall, 2010.

[3] X.B. Hu, E. Di Paolo, A ripple-spreading genetic algorithm for the aircraft sequencing problem, Evol. Comput. 19 (1) (2011) 77–106.

[4] X.B. Hu, M. Wang, M.S. Leeson, E.L. Hines, E. Di Paolo, Deterministic ripple-spreading model for complex networks, Phys. Rev. (E) 83 (4) (2011) 046123.

[5] J.Q. Liao, X.B. Hu, M. Wang, M.S. Leeson, Epidemic Modelling by Ripple-Spreading Network and Genetic Algorithm, Math. Probl. Eng. (2013) 506240.

[6] X.B. Hu, M. Wang, M.S. Leeson, E. Di Paolo, H. Liu, Deterministic agent-based path optimization by mimicking the spreading of ripples, Evol. Comput. 24 (2) (2016) 319–346.

[7] X.B. Hu, M.K. Zhang, Q. Zhang, J.Q. Liao, Co-evolutionary path optimization by ripple-spreading algorithm, Transport. Res. (B) 106 (2017) 411–432.

[8] X.B. Hu, C. Zhang, G.P. Zhang, M.K. Zhang, H. Li, M.S. Leeson, J.Q. Liao, Finding the *k* shortest paths by ripple-spreading algorithms, Eng. Appl. Artif. Intell. 87 (2020) 103229.

[9] R.M. Guo, X.B. Hu, An effective method to find the k shortest paths in a generalized time-window network, ACTA Electronica Sinica 48 (7) (2020) 1387–1395.

[10] L. Fu, D. Sun, L.R. Rilett, Heuristic shortest path algorithms for transportation applications: State of the art, Comput. Operations Res. 33 (11) (2006) 3324–3343.

[11] G. Liu, Z. Qiu, H. Qu, L. Ji, A. Takacs, Computing k shortest paths from a source node to each other node, Soft Comput. 19 (8) (2015) 2391–2402.

[12] A.J.V. Skriver, K.A. Andersen, A label correcting approach for solving bicriterion shortest-path problems, Computers & Operations Research 27 (6) (2000) 507–524.

[13] A. Konak, D.W. Coit, A.E. Smith, Multi-objective optimization using genetic algorithms: A tutorial, Reliability Engi. and Sys. Safety 91 (9) (2006) 992–1007.

[14] Y. Hong, B. Choi, G. Oh, K. Lee, Y. Kim, Nonlinear conflict resolution and flow management using particle swarm optimization, IEEE Trans. on Intell. Transport. Syst. 18 (12) (2017) 3378–3387.

[15] A. Elmia, S. Topaloglu, Multi-degree cyclic flow shop robotic cell scheduling problem: Ant colony optimization, Comput. Operations Res. 73 (2016) 67–83.

[16] D. Trachanatzi, M. Rigakis, M. Marinaki, Y. Marinakis, A firefly algorithm for the environmental prize-collecting vehicle routing problem, Swarm Evolutionary Comput. 57 (2020) 100712.

[17] X.B. Hu, H. Li, J. Zhou, M.K. Zhang, J.Q. Liao, Finding all Pareto optimal paths for dynamical multi-objective path optimization problems, The 2018 IEEE Symposium Series on Computational Intelligence (SSCI 2018), 2018 18 Nov-21 Nov.

[18] Y. Wang, L. Wang, Z.P. Peng, G.C. Chen, Z.Q. Cai, L.N. Xing, A multi ant system based hybrid heuristic algorithm for vehicle routing problem with service time customization, Swarm Evolutionary Comput. 50 (2019) 100563.

[19] R.T. Marler, J.S. Arora, Survey of multi-objective optimization methods for engineering, Struct. Multidisc. Optim. 26 (2004) 369–395.

[20] J. Figueira, S. Greco, Multiple criteria decision analysis: state of the art surveys, Matthias Ehrgottc, 2005.

[21] N. Barr, Economics of the welfare state, Oxford University Press (USA), New York, 2004.

[22] Y. Sawaragi, H. Nakayama, T. Tanino, Theory of Multiobjective Optimization, Academic Press Inc., Orlando, FL, 1985.

[23] L.A. Zadeh, Optimality and Non-Scalar-Valued Performance Criteria, IEEE Trans. Autom. Control 8 (1963) 59–60.

[24] V. Chankong, Y.Y. Haimes, Multiobjective Decision Making Theory and Methodology, Elsevier Science Publishing, New York, 1983.

[25] A. Messac, From dubious construction of objective functions to the application of physical programming, AIAA J. 38 (2000) 155–163.

[26] E.N. Gerasimov, V.N. Repko, Multicriterial optimization, Sov. Appl. Mech. 14 (1978) 1179–1184.

[27] R.T. Marler, J.S. Arora, Survey of multi-objective optimization methods for engineering, Struct. Multidisc. Optim. 26 (2004) 369–395.

[28] I. Das, J.E. Dennis, Normal-boundary intersection: a new method for generating the

pareto surface in nonlinear multicriteria optimization problems, SIAM J. Optim. 8 (1998) 631–657.

[29] A. Messac, A. Ismail-Yahaya, C.A. Mattson, The normalized normal constraint method for generating the Pareto front, Structural and multidisciplinary optimization 25 (2) (2003) 86–98.

[30] A. Messac, C.A. Mattson, Normal constraint method with guarantee of even representation of complete Pareto front, AIAA J. 42 (10) (2004) 2101–2111.

[31] T. Erfani, S.V. Utyuzhnikov, Directed Search Domain: A Method for Even Generation of Pareto Front in Multiobjective Optimization, Eng. Optim. 43 (5) (2011) 467–484.

[32] N. Srinivasand, K. Deb, Multi-objective optimization using nondominated sorting in genetic algorithms, Evol. Comput. 2 (3) (1994) 221–248.

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

[34] Y. Hou, N.Q. Wu, Z.W. Li, Y. Zhang, T. Qu, Q.H. Zhu, Many-objective optimization for scheduling of crude oil operations based on NSGA-III with consideration of energy efficiency, Swarm Evol. Comput. 57 (2020) 100714.

[35] W.F. Leong, G.G. Yen, PSO-Based Multiobjective Optimization With Dynamic Population Size and Adaptive Local Archives, IEEE Trans. Systems, Man, and Cybernetics, Part B 38 (5) (2008) 1270–1293.

[36] J.D. Knowles, D.W. Corne, Approximating the nondominated front using the Pareto archived evolution strategy, Evol. Comput. 8 (2) (2000) 149–172.

[37] Y. Yuan, Y.S. Ong, A. Gupta, H. Xu, Objective reduction in many-objective optimization: evolutionary multiobjective approaches and comprehensive analysis, IEEE Trans. Evol. Comput. 22 (2) (2018) 189–210.

[38] Q.F. Zhang, H. Li, MOEA/D: A multiobjective evolutionary algorithm based on decomposition, IEEE Trans. Evol. Comput. 11 (6) (2008) 712–731.

[39] R. Tanabe, H. Ishibuchi, A framework to handle multimodal multiobjective optimization in decomposition-based evolutionary algorithms, IEEE Trans. Evol. Comput. 24 (4) (2020) 720–734.

[40] S. Huband, P. Hingston, L. Barone, R.L. While, A review of multiobjective test problems and a scalable test problem toolkit, IEEE Trans. on Evol. Comput. 10 (5) (2006) 477–506.

[41] X.B. Hu, H.L. Zhang, C. Zhang, M.K. Zhang, H. Li, M.S. Leeson, A benchmark test problem toolkit for multi-objective path optimization, Swarm and Evol. Comput. 44 (2019) 18–30.

[42] Q. Chen, J Ding, S. Yang, T. Chai, A Novel Evolutionary Algorithm for Dynamic Constrained Multiobjective Optimization Problems, IEEE Trans. Evol. Comput. 24 (4) (2020) 792–806.

[43] M. Rong, D. Gong, Y. Zhang, Y. Jin, W. Pedrycz, Multidirectional Prediction Approach for Dynamic Multiobjective Optimization Problems, IEEE Trans. Cybernetics 49 (9) (2019) 3362–3374.

[44] X.B. Hu, M. Wang, E. Di Paolo, Calculating Complete and Exact Pareto Front for Multi-Objective Optimization: A New Deterministic Approach for Discrete Problems, IEEE Trans. on Cybernetics 43 (3) (2013) 1088–1101.

[45] A. Raith, M. Ehrgott, A comparison of solution strategies for biobjective shortest path problems, Comp. Operations Res. 36 (2009) 1299–1331.

[46] A. Raith, M. Ehrgott, A two-phase algorithm for the biobjective integer minimum cost flow problem, Comp. Operations Res. 36 (2009) 1945–1954.

[47] F.J. Pulido, L. Mandow, J.L. Perez-la-Cruz, Dimensionality reduction in multiobjective shortest path search, Comp. Operations Res. 64 (2015) 60–70.

[48] A. Sedeno, A. Raith, A Dijkstra-like method computing all extreme supported non–dominated solutions of the biobjective shortest path problem, Comp. Operations Res. 57 (2015) 83–94.

[49] D. Duque, L. Lozano, A. Medaglia, An exact method for the biobjective shortest path problem for large-scale road networks, Eur. J. Operational Res. 242 (2015) 788–797.

[50] M. Okulewicz, J. Mańdziuk, A metaheuristic approach to solve Dynamic Vehicle Routing Problem in continuous search space, Swarm Evol. Comput. 48 (2019) 44–61.

[51] R.S. Luo, T.J.J. Van den Boom, B. De Schutter, Multi-Agent Dynamic Routing of a Fleet of Cybercars, IEEE Trans. on Intell. Transport. Sys. 19 (5) (2018) 1340–1352.

[52] D. Frigioni, A. Marchetti-Spaccamela, U. Nanni, Fully dynamic algorithms for maintaining shortest paths trees, J. Algorithms 34 (2) (2000) 251–281.

[53] R. Bauer, D. Wagner, Batch dynamic single-origin shortest-path algorithms: An experimental study, Experimental Algorithms (2009) 51–62.

[54] A. Orda, R. Rom, Shortest-path and minimum delay algorithm in networks with time-dependent edge-length, J. of ACM 37 (3) (1990) 607–625.

[55] K. Sung, M. Bell, M. Seong, S. Park, Shortest paths in a network with time-dependent flow speeds, Eur. J. Operational Res. 121 (1) (2000) 32–39.

[56] L.T. Mei, Reliable shortest path problems in stochastic time-dependent networks, J. Intell. Transport. Sys. 18 (2) (2014) 177–189.