A Review on Ripple-Spreading Genetic Algorithms for Combinatorial Optimization Problems

Xiao-Bing Hu, Mark S. Leeson, Evor L. Hines
School of Engineering
University of Warwick
Coventry, UK.

Ming Wang
State Key Laboratory of Earth Surface Processes and Resource Ecology
Beijing Normal University
Beijing, China

Ezequiel Di Paolo
Department of Informatics
University of Sussex
Brighton, UK

***Abstract*—In various implementations of genetic algorithms (GAs) to combinatorial optimization problems, permutation representations are often adopted. However, these permutation-representation-based implementations are often confronted with one or more of the following problems: (i) Evolutionary operations may generate infeasible solutions; (ii) The representations are not memory-efficient, and may hamper the scalability of algorithms; (iii) Many classic binary evolutionary operators can hardly apply without significant modifications. To address these issues, a novel scheme for applying binary-representation-based GAs to combinatorial problems has recently been proposed based on a ripple-spreading model. Since this model is the centerpiece of the new scheme, we call it the ripple-spreading genetic algorithm (RSGA). In previous studies, several bespoke RSGAs have been developed to tackle a range of different combinatorial optimization problems. Although these RSGAs differ in design details, they share the same motivation, follow the same methodology and illustrate the same advantages vis-a-vis other methods. Based on the previous studies, this paper aims to present a comprehensive review of the methodology of RSGA. Particularly, by analyzing those existing implementations of RSGA, this paper will discuss some generalized important technologies for designing RSGA.**

Keywords—Ripple Spreading Model, Genetic Algorithm, Combinatorial Problem, Permutation Representation, Binary String.

I. INTRODUCTION

Many combinatorial optimization problems, such as the evacuation of people and the allocation of resources in natural catastrophes, are NP-hard or even NP-complete problems. As large-scale parallel stochastic searching and optimizing algorithms, genetic algorithms (GAs) have a good potential for resolving such combinatorial optimization problems. In fact, ever since the GA was firstly developed in 1970's, it has been applied to various combinatorial problems and achieved impressive performance [1].

As is well known, the GA was originally designed based on a binary representation; in other words, binary strings were used to represent the values of those variables which needed to be evolved [2]. For instance, minimizing $(x^2+6x+10)$ is a numerical optimization problem, and therefore a binary string can be used to represent the value of x. However, the principle of using binary representation can hardly apply directly when designing GA for combinatorial problems because the solutions to such problems are based on order or permutation rather than value. Instead, to make GAs applicable to combinatorial problems, various permutation representations often have to be adopted. For example, in the travelling salesman problem (TSP), which is a well-known combinatorial optimization problem, the solution is the order of visiting different cities, and can be represented by an integer-based vector, e.g., [1 3 7 2 5 8 4 6 1] records a circle route to visit 8 cities. Unfortunately, permutation representations are often associated with serious feasibility issue, i.e. lots of infeasible offspring may be generated during evolutionary operations, even if all parents are feasible. In the case of the TSP, crossover could easily generate an infeasible solution which visits a city twice [3]. Another issue is scalability, i.e. permutation representations often have a poor scalability. For instance, an adjacency matrix is often used to record a network topology. However, for a complex network with thousands or even millions of nodes, the memory demand of such a matrix makes it very difficult, if not impossible, to apply a GA to evolve a population of topologies [4]. The third issue is compatibility, i.e. it is difficult or inefficient to apply many classic evolutionary operators to permutation representations. Particularly, the usefulness of crossover becomes doubtful when permutation representations are used. Many permutation-representation-based implementations have to develop highly problem-specific crossover operators [5], and some even discard crossover [6].

By noticing that all the above problems result, more or less, from using permutation representations, we then ask whether it will be possible to use very basic binary strings to represent the solutions of combinatorial problems. It should be noted that what we are concerned about here is not a problem-specific binary string for a certain combinatorial problem but a general methodology of applying binary strings to various combinatorial problems. A common characteristic of all combinatorial solutions is that it is not the value of the variables but the order and/or permutation of a set of elements that composes such a solution. Binary strings are suitable for representing value-based solutions. If we can design a mapping function, which maps each set of input values into a unique order and/or permutation of elements in the combinatorial problem concerned, then we can use binary strings to represent the input values, and apply a GA to evolve such input values in

Proc. 9th IEEE Int. Conf. on Cognitive Informatics (ICCI'10)
F. Sun, Y. Wang, J. Lu, B. Zhang, W. Kinsner & L.A. Zadeh (Eds.)
978-1-4244-8040-1/10/$26.00 ©2010 IEEE

order to find the optimal or sub-optimal solutions to the combinatorial problem. Inspired by the natural ripple spreading phenomenon, it is possible to establish a general methodology for designing a proper mapping function in order to transform a combinatorial problem into a numerical problem. To give a simple explanation of the inspiration, we assume there is a set of stakes standing in a pond. Some ripples are randomly generated in the pond. As the ripples spread out from their epicenters, they reach each stake at different times and with different amplitudes, according to which, an order or permutation of combining all stakes may be determined. Apparently, some parameters, such as the epicenter locations and initial amplitudes of the ripples, will determine how the ripples reach each stake. Let the stakes be the elements which compose the solution of a combinatorial problem, then in general, it is possible to design a mapping function from the ripple-spreading related parameters to the combinatorial solutions.

It is based on the above inspiration of designing a mapping function that we have developed practicable ripple-spreading models for quite a few combinatorial problems, such as the TSP [7], the network topology optimization problem [8], the airport gate assignment problem (GAP) [9], the aircraft sequencing problem (ASP) [10] and the network coding problem (NCP) [11]. It is because of the ripple-spreading models that we have successfully applied binary GAs to these combinatorial problems, free of feasibility, scalability and compatibility issues. Due to the crucial role played by the ripple-spreading models, we call the algorithms reported in the above references ripple-spreading GA (RSGA). In this paper, we will analyze the designs of these RSGAs, in order to summarize this methodology of using ripple-spreading models to enable the general application of binary GA to various combinatorial problems.

II. BASIC IDEA OF RSGA

As discussed in Section 1, permutation representations often cause problems in feasibility, scalability and compatibility. To address these issues, an often used methodology is to transform the original problem into a new problem whose solutions are based on value rather than order or permutation, and then apply a basic binary GA to resolve the transformed problem. The RSGA scheme in this paper involves such a problem transforming process, which is inspired by the natural ripple spreading phenomenon. Basically, a solution of combinatorial problems determines how to organize a set of elements, particularly, in what kind of order. Suppose such elements are associated with some points in an artificial space where some ripples will be generated randomly. Then as the ripples spread out in the space, those element-related points can be organized according to, say, the order by which the ripples reach each of them. Obviously, such an order is determined by the values of some ripple-spreading related parameters, e.g., the epicenter, speed and amplitude of ripples. Therefore, a combinatorial solution is transformed into a value-based solution. This is the original inspiration of remodeling/transforming a combinatorial problem into a ripple spreading process, and then comes the proposed RSGA scheme.

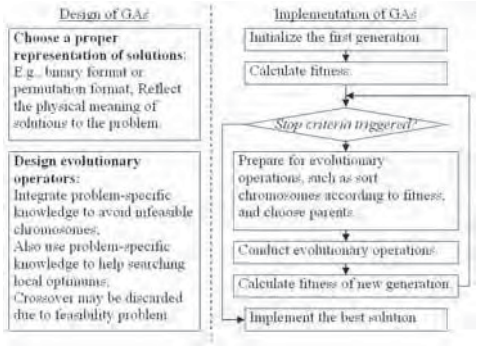

Fig.1. Conventional way to apply GAs

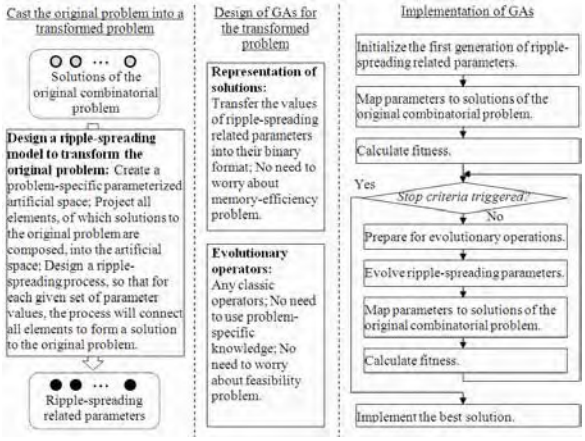

Fig.2. The basic idea of the RSGA scheme

The RSGA scheme aims to test the idea of using very basic binary GAs to solve those combinatorial problems which usually require permutation representations. The basic binary GAs used must be free of feasibility and memory-efficiency problems and compatible with all classic evolutionary operators. The basic idea of the RSGA scheme is illustrated in Fig.2 (Fig.1 summarizes the conventional way of applying GAs). Usually, basic binary GAs are easy to design for those problems where the solutions are based on value, and to such problems all classic evolutionary operators, such as mutation, one-point crossover, weighted crossover and uniform crossover, are applicable. However, if the solutions are based on permutation, such as the order of visiting cities in the TSP [7], and the combination of edges in a random complex network [8], then it is often difficult to use value-based binary strings to represent them directly. In the RSGA scheme, an original combinatorial problem, whose solutions are based on permutation and therefore are unsuitable for binary representation, needs to be transformed into a ripple-spreading model whose solutions are based on the values of some ripple-

spreading related parameters. Then a basic binary GA is designed to evolve the values of these parameters, and all classic GA techniques can therefore be applied straightforwardly. In the implementation of the RSGA, the only thing different from a classic GA is that, before the fitness of a chromosome is calculated, the represented solution to the transformed problem (i.e., a set of values for the ripple-spreading related parameters) needs to be mapped back into the associated solution to the original combinatorial problem. Obviously, the most important and also the most difficult step in the RSGA scheme is to design a proper ripple-spreading model, which largely depends on each individual combinatorial problem.

In the next few sections, we will make a comprehensive review of those important aspects in the design of RSGAs in our previous studies, in order to give some general guidelines for applying the RSGA scheme to more other combinatorial problems.

III. Design of Ripple-spreading models

A. Artificial space

In order to transform a combinatorial problem into a ripple-spreading model, we firstly need an artificial space where we can project all elements composing combinatorial solutions and generate ripples. Basically, the design of artificial space is a highly problem-specific task, but there is a lot of freedom in the design.

The artificial space may originate from a real space. For instance, the artificial space used in the TSP is actually the geographic map of cities [7], and similar are those used in the study of some real world complex networks such as power grids and transportation networks [8]. Therefore there is no need to project any elements.

However, in most cases, it is difficult to refer to any real space to design the artificial space. In this case, we may construct an imaginary space based on the categories/features of the elements composing combinatorial solutions, i.e., each axis of the artificial space is corresponding to a certain category/feature of elements. For instance, the artificial space used in the study on the ASP has an axis of time and an axis of aircraft category [10], and the RSGA for the GAP constructed an imaginary space with an axis of time, an axis of passenger number, and an axis of aircraft grounding time [9]. The elements composing combinatorial solutions can then be projected into the artificial space according to their categories/features.

Sometimes there is no useful information about the categories/features of elements, such as the nodes in non-geographic networks or graph and information theories [8], [11]. The artificial spaces used in these studies are purely imaginary, and the elements are projected in a rather arbitrary manner. However, some problem-specific tricks might be helpful. For instance, the source node in the network coding problem is always projected to the origin of the space [11], and the nodes in a non-geographic network can be projected according to statistical features, such as a Gaussian distribution [8].

The artificial space can be any dimensional for the sake of problem-resolving. For instance, in our previous studies on the RSGA scheme, there are one-dimensional [11], two-dimensional [7], [8], [10], and three-dimensional artificial spaces [9].

In an artificial space with more than one dimension, different axes may have different distance units. For instance, the distance units for the time axis and the aircraft category axis in [10] are completely different and even incomparable. In this case, we need to introduce an artificial coefficient to unify two different distance units. Such coefficients, along with other ripple-spreading parameters, can be later evolved by GA [9], [10]. This can help to optimize the ripple-spreading model, in order to improve the performance of RSGA.

B. Ripple-spreading process

After projecting all elements into the artificial space, we need to design a ripple-spreading process in the space, so that the behavior of ripples, which is determined by ripple-spreading related parameters, will determine how to combine the elements to generate combinatorial solutions. The design of ripple-spreading process is also a highly problem-specific task. Depending on the nature and characteristics of the combinatorial problem concerned, the ripple behaviors in our previous implementations of the RSGA scheme can be roughly classified into three categories with different degrees of complexity:

Category I: The time by which a ripple reaches an element point is the key factor to determine how to combine all elements. This category is particularly suitable for many sequencing problems, such as the TSP [7], the ASP [10] and the GAP [9]. Basically, the elements are sequenced mainly according to the order by which they are reached by the ripples, but other information, e.g., the angle from a city to the ripple epicenter in the TSP, may also be used. Fig.3 and Fig.4 illustrate how the ripple-spreading processes determine combinatorial solutions in the TSP and the ASP, respectively.

Category II: The amplitude of a ripple when it reaches an element point is the key factor to determine how to generate combinatorial solutions. This category is useful when each element has many states/features/values to choose from. Different ripple amplitude will trigger or weigh different state/feature/value for an element appearing in a combinatorial solution. The implementation of RSGA to the NCP is a good example of this category [11]. In the NCP, an outgoing link needs to choose a signal or a combination of signals from associated incoming links. Fig.5 shows how the choice is made by a ripple-spreading process. In Fig.5, each ripple is associated with the signal on a specific incoming link. Whether the associated signal will be chosen is determined by comparing the amplitude of the ripple at a sampling time against a threshold. In Fig.5, the signals associated with ripple

1 and ripple 2 are chosen as they are above the threshold at the sampling time.

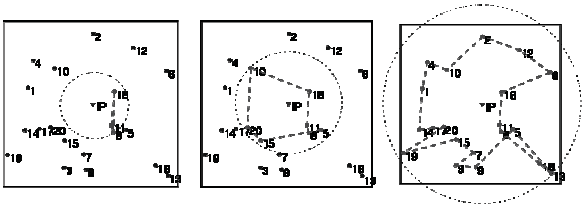

Fig.3. The ripple-spreading process in the TSP [7]

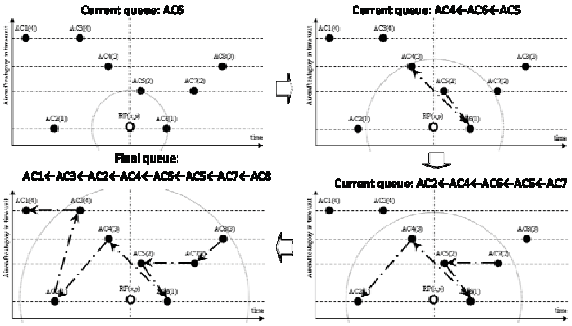

Fig.4. The ripple-spreading process in the ASP [10]

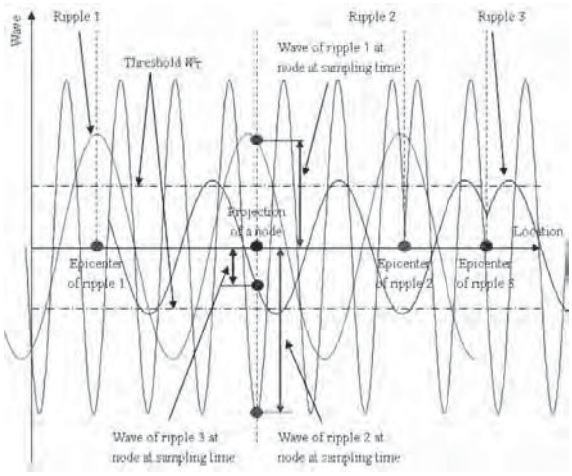

Fig.5. The ripple-spreading process in the NCP [11]

Category III: An element point can be activated by a ripple to generate new ripples, which may activate other elements in return. This is the most complicated category, and it can help where there are complex interactions or relationships between elements in a combinatorial solution, e.g., how to distribute connections between nodes to generate a random complex network [8]. Fig.6 gives a simple illustration about how a network is formed through a ripple-spreading process. Basically, whether a node will be connected and whether a node will be activated to generate a new ripple depends both on the point energy of an incoming ripple when it reaches the node, and the point energy of a ripple decay as it spreads. As illustrated in Fig.6, firstly, an initial ripple (the red ripple in Fig.6) is generated randomly in the artificial space. As the initial ripple spreads out, it triggers node 1 to node 3 to generate 3 new ripples, but fails to activate node 4 as its point energy becomes too weak when it reaches node 4 (the strength of point energy is indicated by the thickness of the ripple in Fig.6). The three new ripples establish 5 connections between the 4 nodes, but fail to trigger any more new ripples.

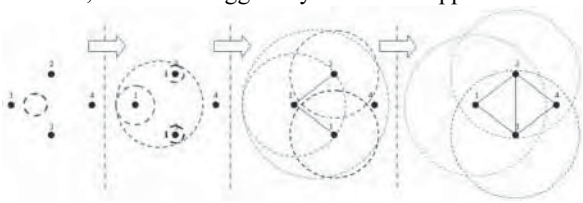

Fig.6. The ripple-spreading process in the NCP [8]

Whichever category a ripple-spreading process belongs to, there needs to be a mathematical model to describe the process. Basically, such a model includes inputs and dynamic functions, and there is great freedom to design the inputs and dynamic functions of ripple-spreading model, as long as the resulting model helps to resolve the concerned combinatorial problem. In our previous studies on RSGA [7]-[11], quite a few different inputs and dynamic functions are developed, as summarized in following:

A most important input is the location of ripple epicenter. All the models in our previous studies on RSGA have this input. Any change in the location of ripple epicenter may cause a different output of the model, i.e. generate a different combinatorial solution.

Initial states of a ripple, including initial amplitude, initial point energy, frequency, initial phase, et al. These inputs are mainly used in modeling ripple-spreading processes of category II and category III, e.g., see [8] and [11].

All the models in our previous studies have a set of dynamic functions of ripples, which defines how each ripple will spread in the artificial space. For ripples of category I, such as in [7], [9] and [10], the time by which a ripple reaches an element point is the key factor, so, simple linear motion functions like following are used:

$$d=s\times t, \tag{1}$$

where d is the distance to the ripple epicenter, s the speed, and t the time.

For ripples of category II, like in [11], sine functions similar to the following are used in order to describe the amplitude of a ripple at a certain point at a certain time:

$$a(d,t)=\begin{cases} A\times\sin(f\times(t-d/s)+\theta), & t\geq d/s \\ 0, & t<d/s \end{cases} \tag{2}$$

where A, f and θ are the initial amplitude, frequency and initial phase of a ripple, respectively, and $a(d,t)$ is the amplitude of the ripple at distance d at time t.

For ripples of category III, like in [8], an energy decaying function may be necessary such as:

$$e(d,t)=\begin{cases} (E\times\alpha^{-\lambda\times t})/(2\times\pi\times d), & t\geq d/s \\ 0 & t<d/s \end{cases}, \ \alpha\geq 1, \tag{3}$$

where E is the initial point energy of a ripple, i.e., the point energy at the ripple epicenter, $e(d,t)$ is the point energy at distance d at time t, and α and λ are two coefficients determining the energy decaying rate.

Sometimes, e.g., in category II and category III, we need to define some reaction functions for element points when they are reached by a ripple. For instance, in [11], a linear function as below is introduced for an outgoing link in order to combine signals on associated incoming links according to the amplitude of ripples:

$$S_{Out} = \sum_{i=1}^{H} w(i) S_{In}(i), \tag{4}$$

where

$$w(i) = \frac{\max(0, a(d,t,i) - A_T)}{\max_{j=1,\ldots,H}(a(d,t,j) - A_T)}, \tag{5}$$

S_{Out} is the signal on a outgoing link, $S_{\mathrm{In}}(i)$ is the signal on the ith associated incoming link, $a(d,t,i)$ is the amplitude of the ripple related to the ith associated incoming link, and A_{T} is a threshold which determines whether the signal on an incoming link, i.e., $S_{\mathrm{In}}(i)$, will be used to generate S_{Out}. Eq.(4) can be simplifies to a 0-1-weighted combination if, say

$$w(i) = \begin{cases} 1, & a(d,t,i) \geq A_T \\ 0, & a(d,t,i) < A_T \end{cases}. \tag{6}$$

If a self-adjusting threshold is used in Eq.(6), such as

$$A_T = \max_{j=1,\ldots,H} a(d,t,j), \tag{7}$$

then, each time, an element simply chooses one state/feature/value from a given set.

In [8], two thresholds, E_{T1} and E_{T2}, are used, E_{T1} to determine whether a connection is triggered by ripples, and E_{T2} to determine whether a new ripple is triggered by an incoming ripple. Once a new ripple is triggered, its initial point energy can be calculated as following:

$$E = \rho(e(d,t) - E_{T2}), \tag{8}$$

where ρ is an amplifying coefficient.

Besides the above inputs and dynamic functions, some other parameters, such as the number of ripple epicenters and the coefficients for unifying the distance units of different axes, also play an important role in determining the output of a ripple-spreading model. Particularly, the number of ripple epicenters needs to be set up carefully, in order to simplify the model without degrading the performance. For sequencing problems like those in [7], [9] and [10], the number of ripple epicenters can equal to the number of sequences needed to be generated, and this gives very satisfactory performance, although a model with more ripple epicenters also works. For category II, the number of ripple epicenters may rely on the total number of states/features/values an element may have. For instance, in [11], the number of ripple epicenters depends on the maximum number of incoming links to a node in a network. Sometimes, it may be difficult to decide the number of ripple epicenters. In this case, it can be evolved along with other parameters by GA, just like in [8].

Obviously, all the inputs, coefficients in dynamic functions, and other parameters as discussed above can affect the output of a ripple-spreading model. Therefore, they are all called ripple-spreading related parameters and are all potentially evolvable by GA in order to search for optimal solutions to a combinatorial problem. However, in our previous studies, we usually evolve only some ripple-spreading related parameters in GA, and pre-set and then fix other parameters when running the GA. Which parameters should be evolved in the GA and which may be fixed largely depends on whether, if we do so, we could simplify the model without sacrificing the performance. Basically, all inputs should be evolved by the GA. Regarding the coefficients in dynamic functions and other parameters, our previous studies have shown that there is much flexibility and it is worth further investigation in future research. In general, when applying RSGA to a specific combinatorial problem, one may start with all ripple-spreading related parameters evolved in GA, and then remove some one by one by monitoring the performance.

It should be noted that, in our previous studies, a ripple-spreading model is always a multiple-to-one, or one-to-one if possible, mapping process. In other words, for a given set of values for the parameters, the model will give a unique output. To guarantee this, sometimes problem-specific knowledge needs to be integrated into the model. For instance, in the study on the ASP, when two or more aircraft points have the same distance to the ripple epicenter, then they will be uniquely sequenced according to some common senses/rules in air traffic control practices [10]. However, as illustrated by the semi-deterministic sub-model and the stochastic sub-model in the study on complex random networks, a ripple-spreading model could itself be designed as a multiple-to-multiple, or even one-to-multiple, mapping process [8]. Such models are potentially useful to complex network theories, but how to design an effective RSGA based on them is an open question and then is worth further investigation in future research.

IV. Design of Binary GA

In the RSGA scheme, there is no need for any special effort to design the evolutionary algorithm, and the very basic binary GA can apply directly.

A. Chromosome structure: Binary representations vs. permutation representations

Problem-specific permutation representations are usually the natural and intuitive choice to describe combinatorial solutions, because such representations embody the underlying physical meanings of combinatorial solutions in a very straightforward way. Basically, there are two categories of popular permutation representations: vector and matrix. Usually, vector representations directly deal with the elements of combinatorial solutions, whilst matrix representations record the relationships between every two elements. For instance, a vector directly gives the order to visit each city in

the TSP [3], and a matrix clearly indicates the connections between nodes in a network [4]. Therefore, it is fair enough to say, that permutation representations are the first choice to study combinatorial problems. However, the first choice may not be the best choice. Scalability is an issue. The size of either vector or matrix is dependent of the problem scale. Usually, the scale of a combinatorial problem can be measured by N_E, the number of all elements composing combinatorial solutions. Clearly, the memory demand of vector representations is $O(N_E)$, whilst that of matrix representations is $O(N_E^2)$. Feasibility is another issue. Since permutation representations are actually combinatorial solutions, they have to observe all physical constraints imposed on the solutions. For instance, a city must be visited once and only once in the TSP, and an aircraft must be included in a landing sequence once and only once in the ASP. These constraints coming with permutation representations make it very difficult to guarantee the feasibility of chromosomes in the evolutionary process.

The RSGA scheme requires no effort to design the chromosome structure. No matter what kind of combinatorial problems are to be resolved, all chromosomes used in RSGA are just the very basic binary strings storing the values of ripple-spreading related parameters, i.e., they are the values of those parameters in binary format. The memory demand of RSGA is very low, just depending on the number of ripple-spreading related parameters. For instance, to represent a network with 1000 nodes, a permutation-representation-based GA needs a matrix of 1 Megabyte, whilst the RSGA uses a binary string recording just tens of parameters [8]. Therefore, the RSGA has a very good scalability in terms of memory demand. Since the binary strings used in RSGA have nothing to do with combinatorial solutions, they have to observe no physical constraints. This makes the evolutionary process in RSGA free of feasibility problems, as will be discussed further in the next sub-section "Evolutionary operators".

Table I summarized the main features of different representations.

TABLE I
FEATURES OF DIFFERENT REPRESENTATIONS

	Meaning of a gene	Meaning of a chromosome	Size of a chromosome	Constraints
Vector representation	Directly related to an element	Represents a combinatorial solution	Depends on the problem scale	Constraints on elements
Matrix representation	Explicitly defines a relationship between two elements	Represents a combinatorial solution	Depends on the power of problem scale	Constraints on the relationship between elements
Binary representation	Just a digit bit in the binary string. Has nothing to do with the elements	Ripple spreading related parameters	Depends on the number of ripple spreading related parameters	None

B. Evolutionary operators

It is often a challenging task to design evolutionary operators based on permutation representations. To address the feasibility issue caused by evolutionary operations, some special modifications have to be made to classic evolutionary operators. The modified operators are often highly problem-specific and therefore have little generality. For instance, classic crossover operators, e.g. one-point crossover and uniform crossover, can hardly apply in the study on TSP, and researchers have developed many purpose-designed crossover operators, such as partially matched crossover, cycle crossover, order crossover and edge assembly crossover, which are rarely extended to other problems [3]. Sometimes, the modified operators shift their emphasis from evolving chromosomes to fixing infeasible chromosomes, such as in some studies of applying GA to the ASP [12]. The modified operators are usually computationally expensive. The matrix based crossover operator in [5] is a typical example of a heavy computational burden. Some researchers even discard crossover because, based on permutation representations, it seems more destructive rather than effective to evolve chromosomes [6].

Thanks to ripple-spreading models, we do not need to operate directly on the elements to search for good combinatorial solutions. Instead, we just need to evolve the values of ripple-spreading related parameters. This means, when applying GA to combinatorial problems, we can thoroughly say goodbye to permutation representations, and simply operate on binary strings as if we are dealing with numerical optimization problems. Therefore, no matter what combinatorial problem is to be resolved, we are free to use any classic evolutionary operators with no need to make any modification, and no need to worry about any feasibility problem. In other words, the RSGA scheme requires no effort to design evolutionary operators. This is exactly the case in all implementations of RSGA as reported in [7]-[11].

C. Heuristic rules

In permutation-representation-based GAs for combinatorial problems, many problem-specific heuristic rules are integrated into their evolutionary operations. In contrast, in the evolutionary operations of RSGA, we mainly focus on some pure GA-related heuristic rules. For example, evenly distributing a certain proportion of chromosomes within the solution space of the ripple-spreading model when initializing a generation; online adjust mutation probability and crossover probability according to the fitness of each individual chromosome as well as the overall fitness level of the current generation of chromosomes [13]; dynamically restrict the mutation operation to a certain part of a chromosome based on its fitness [14]; some GA-related parameters may be included in the chromosome structure and then get evolved along with those ripple-spreading related parameters, just like genetic strategies do [15]. As for problem-specific knowledge, it should be mainly integrated into the ripple-spreading model. This is why, for instance, the model for the ASP can automatically filter out most bad landing sequences [10], and

the model for the TSP can usually avoid unnecessary zigzag routes [7].

V. Further Analysis of RSGA

A. Completeness of solution space

An obvious question about the RSGA is: Will the ripple-spreading model cover the whole solution space of the original combinatorial problem?

In [10], a ripple-spreading model is indeed proposed that can in theory cover the whole solution space of the original ASP. Basically, the model has as many ripple epicenters as arriving aircraft. Each epicenter has a serial number and will generate its own ripple. The first aircraft point which is reached by a ripple will be assigned with the same serial number as that of the associated epicenter. Then aircraft will be sequenced according to their assigned serial numbers. It is easy to see that, by changing the locations of these epicenters, one can get any candidate landing sequence, which means the solution space of the original ASP is completely covered by this ripple-spreading model.

Basically, the above ripple-spreading model with a guarantee of completeness in [10] can easily be extended to encompass many other problems, i.e. by using as many epicenters as all the elements that compose combinatorial solutions, it is possible to design a ripple-spreading model which covers the whole solution space of the original combinatorial problem. However, such models with a guarantee of completeness are often of low efficiency and poor performance.

From a practical point of view, guaranteeing the completeness of solution space is a much less important issue, as long as some good solutions, not necessary optima, can be found in a timely manner. In [10], another simpler ripple-spreading model for the ASP uses only one epicenter and just covers a small portion of the original solution space, but it is theoretically proved to be much more efficient than the model with a guarantee of completeness. It is because this simpler model can automatically filter out most bad solutions in the first place. As a result, although the simpler model cannot guarantee the completeness of solution space, it is very likely to cover a considerable proportion of good solutions. Therefore, instead of searching for 10 needles in 10 haystacks, the simpler model stands a very good chance to search for 5 needles in 1 haystack, although the best needle might not be hidden in this particular haystack. To give another example, the ripple-spreading model in the study on the TSP can usually avoid inserting a far away city between two adjacent cities [7]. Actually, most ripple-spreading models in our previous studies cannot guarantee the completeness of solution space, but they do cover sufficient good solutions and then can find some of them very quickly, and comparative experiments with those permutation-representation-based GAs, which can cover the whole solution space, show that the RSGAs have achieved similar or even better performance [7]-[11].

B. Memory efficiency vs. time efficiency

Another question about the RSGA is: Binary representations may save a lot of computational memory, but it requires an additional ripple-spreading process to decode a binary string to a combinatorial solution, and such a decoding process is sometimes not cheap in terms of computational time, like in [7] and [8]. Therefore, is it worth sacrificing time efficiency, in order to improve memory efficiency? In fact, introducing a ripple-spreading process is a double-edged sword in terms computational efficiency. On one hand, the decoding process will cause additional computational burden. On the other hand, the resulting binary representations may significantly improve the computational efficiency of evolutionary operations, particularly when considering those computationally expensive measures which are taken by permutation-representation-based GAs in order to guarantee feasibility. Actually, as reported in [10], the total computational time consumed by RSGA is the least among all GAs, whilst the performance of RSGA is the best. In other words, in the implementation to the ASP, the RSGA scheme exhibits advantages in both memory efficiency and computational time efficiency. However, no relevant experimental comparative results are given in other implementations, and this is definitely an issue needed to be clarified in future research as well as in any new implementation of the RSGA scheme.

C. Problem-specific knowledge in RSGA

As mentioned before, the design of a ripple-spreading model is a highly problem-specific task. This is not only because the inputs and dynamic functions used in a model may be highly problem-dependent, but also because the design should embody as much problem-specific knowledge as possible that is useful to resolve the concerned combinatorial problem, in order to give good performance. For instance, in the study on the TSP [7], the ripple-spreading model can usually avoid inserting a far away city between two adjacent cities, which is a useful rule when generating a TSP route. In the study on the ASP [10], it is even theoretically proven that the proposed model can automatically observe some important air traffic control rules, such as no need to swap two aircraft of the same category, and low probability to swap two aircraft that are far away from each other. It is difficult to give general guidelines regarding how to integrate problem-specific knowledge into ripple-spreading models, but extra attentions do need to be paid to this issue.

VI. Conclusions and Future Work

This paper, based on some relevant studies [7]-[11], attempts to give a comprehensive review of a recently developed novel methodology for applying basic binary genetic algorithms (GAs) for combinatorial problems. As is well known, permutation representations are often used in implementations of GA to various combinatorial problems, and

feasibility, scalability and compatibility are some problems often confronted by such permutation-representation-based implementations. Inspired by the natural ripple-spreading phenomenon, a novel design methodology has recently be proposed that transforms a combinatorial problem into a numerical optimization problem, and then uses the very basic binary GA, free of the feasibility, scalability and compatibility problems, to evolve new value-based solutions rather than the original combinatorial solutions. Since a ripple-spreading model is the centerpiece of the methodology, the resulting algorithm is therefore called ripple-spreading genetic algorithm (RSGA). Simply speaking, in an RSGA, a ripple-spreading model needs to be developed for the concerned combinatorial problem: An artificial space needs to be designed firstly, so that all elements composing combinatorial solutions can be projected into such a space. Then a ripple-spreading process needs to be designed, so that for a given set of values for ripple-spreading related parameters, the process will, according to the behavior of ripples, generate a unique combination of the elements, i.e. generate a unique combinatorial solution to the original problem. Finally, a very basic binary GA is used to evolve the values of ripple-spreading related parameters, in order to find optimal or sub-optimal solutions to the combinatorial problem. The design of ripple-spreading models is a highly problem-specific task, but there is great flexibility and freedom.

Thanks to the ripple-spreading model, RSGA does not need to operate directly on combinatorial solutions but just evolves the values of ripple-spreading related parameters. Therefore, no permutation representations or problem-specific evolutionary operators are needed, and by using the very basic binary GA there is no need of any extra effort for design. The binary representation in RSGA is independent of the problem scale, so, the algorithm has a good scalability; the binary representation is also compatible with all classic evolutionary operators with no need of any modification; There is no physical constraint to the binary representation, so the RSGA is thoroughly free of feasibility problems, i.e. no infeasible chromosome will be produced during the evolutionary process. Because of the effectiveness and efficiency of binary representation, the RSGAs in previous studies have delivered satisfactory performance when compared with permutation-representation-based GAs.

However, as a new methodology, most results reported so far are very preliminary. Much more effort is needed in order to fulfill the full potential of RSGA in resolving various combinatorial problems. To this end, future work may include: (I) Study the existing ripple-spreading models in more depth, e.g. conduct necessary theoretical analyses of the models in terms of completeness and effectiveness; (II) Develop and investigate ripple-spreading models for some complex problems, such as the outbreak of plagues and the development of risk chains, and compare with other models; (III) Test the RSGA on new combinatorial problems, particularly some real-world problems, such as in the area of integrated risk governance; (IV) Extend the basic idea of using ripple-spreading models to other population-based evolutionary algorithms, such as particle swarm optimization and ant colony optimization, for combinatorial problems.

ACKNOWLEDGMENT

This work was supported by the State Key Laboratory of Earth Surface Processes and Resource Ecology, Beijing Normal University, China.

REFERENCES

[1] Eiben, A.E. and J. E. Smith, *Introduction to Evolutionary Computing*, Berlin, Germany: Springer-Verlag, 2003.

[2] Holland, J.H. *Adaptation in Natural and Artificial Systems*, University of Michigan Press, Ann Arbor, MI, 1975.

[3] Reinelt, G., *TSPLIB, Travelling Salesman Problem*, Universität Heidelberg, 2004, found on http://www.iwr.uni-heidelberg.de/groups/comopt/software/TSPLIB95/tsp/.

[4] Boccaletti, S., V. Latora, M.Y. Chaves, and M. Hwang, D.U., "Complex networks: Structure and dynamics". *Physics Reports* vol. 424, 175-308, 2006.

[5] Hu, X.B. and E. Di Paolo, "An Efficient Genetic Algorithm with Uniform Crossover for Air Traffic Control", *Computers & Operations Research*, Vol.36, pp. 245-259, 2009.

[6] Cheng, V., L. Crawford and P. Menon, "Air traffic control using genetic search techniques", Proceedings of the IEEE International Conference on Control Applications, 1999.

[7] Hu, X.B. and E. Di Paolo, "A Hybrid Genetic Algorithm for the Travelling Salesman Problem", International Workshop on Nature Inspired Cooperative Strategies for Optimization (NICSO 2007), 08-10 Nov 2007, Italy.

[8] Hu, X.B., E. Di Paolo and L. Barnett, "Ripple-Spreading Model and Genetic Algorithm for Random Complex Networks: Preliminary Study", The 2008 IEEE World Congress on Computational Intelligence (WCCI2008), 01-06 June 2008, Hong Kong.

[9] Hu, X.B. and E. Di Paolo, "An Ripple-Spreading Genetic Algorithm for Airport Gate Assignment Problem", The 2009 IEEE Congress on Evolutionary Computation (CEC2009), 18-21 May 2009, Norway.

[10] Hu, X.B. and E. Di Paolo, "A Ripple-Spreading Genetic Algorithm for Aircraft Sequencing Problem", the MIT Evolutionary Computation Journal, in press, 2010.

[11] Hu, X.B. M. S. Leeson and E. L. Hines, "Minimization of Network Coding Resources: Ripple-Spreading Model and Genetic Algorithm", The 2010 IEEE World Congress on Computational Intelligence (WCCI2010), 18-23 July 2010, Barcelona, Spain.

[12] Hansen, J.V., "Genetic search methods for air traffic control", *Computers & Operations Research*, vol.31, pp. 445-459, 2004

[13] Zhang, J., H.S.H. Chung and W.L. Lo, "Clustering-Based Adaptive Crossover and Mutation Probabilities for Genetic Algorithms", *IEEE Transaction on Evolutionary Computation*, vol.11, pp. 326-335, 2007.

[14] Hu, X.B. and S.F. Wu, "Self-Adaptive Genetic Algorithm Based on Fuzzy Mechanism", *Proceedings of the 2007 IEEE Congress on Evolutionary Computation*, Singapore, 25-28, Sep 2007.

[15] Eiben, A.E. and J.E. Smith, *Introduction to Evolutionary Computing*, Berlin, Germany: Springer-Verlag, 2003.