Game-based Memetic Algorithm to the Vertex Cover of Networks

Jianshe Wu, *Member, IEEE*, Xing Shen, and Kui Jiao

***Abstract*—The minimum vertex cover (MVC) is a well known combinatorial optimization problem. A game-based memetic algorithm (GMA-MVC) is provided, in which the local search is an asynchronous updating snowdrift game and the global search is an evolutionary algorithm (EA). The game-based local search can implement (*k*,*l*)-exchanges for various numbers of *k* and *l* to remove *k* vertices from and add *l* vertices into the solution set, thus is much better than the previous (1,0)-exchange. Beyond that, the proposed local search is able to deal with the constraint, such that the crossover operator can be very simple and efficient. Degree-based initialization method is also provided which is much better than the previous uniform random initialization.**

Each individual of the GMA-MVC is designed as a snowdrift game state of the network. Each vertex is treated as an intelligent agent playing the snowdrift game with its neighbors, which is the local refinement process. The game is designed such that its strict Nash equilibrium is always a vertex cover of the network. Most of the strict Nash equilibriums are only local optima of the problem. Then an EA is employed to guide the game to escape from those local optimal Nash equilibriums to reach a better Nash equilibrium. Comparison with the state of the art algorithms in experiments on various networks, the proposed algorithm always obtains the best solutions.

***Index Terms*—Game optimization, memetic algorithm, Nash equilibrium, vertex cover, snowdrift game.**

I. Introduction

THE MINIMUM vertex cover (MVC) problem is an NP-hard combinatorial optimization problem, which aims at finding a minimum number of vertices of a network to cover all the edges [1]. Many optimization problems in cybernetic technology and engineering are finally described as a MVC problem. For example, in a surveillance system of a town, the designer has to place some cameras at the crossroads. Assume that a camera located at a crossroad can monitor every connected path. Since more cameras indicate more cost, the designer wants to know the minimum number of cameras to cover the town. Additional examples of the MVC problem include the communication controlling in wireless sensor networks [2],[3], multiple sequence alignments in computational biochemistry [4], phasor measurement units placement problem in power system [5], measuring the robustness of the networks [6], computing the path width of directed graphs [7], scheduling, networking, and various matching problems, etc [8],[9].

A. Introduction to the MVC

Various algorithms have been proposed for the MVC problem [6]-[10], which fall into two categories: centralized and distributed. For the centralized algorithms, the optimization process is supervised by an administrator with the assistance of global information. Examples include the evolutionary algorithms (EAs) [9]-[14]: the genetic algorithm (GA) proposed in [9] and the hybrid genetic algorithm (HGA) proposed in [14] are two representatives (please see in Section II.*D* for detail). On the contrary, without an administrator, distributed algorithms use only local information. Examples include the evolutionary game-based algorithms [15]-[18]. In [17], synchronous updating rule for the spatial snowdrift game with memory is designed such that its strict Nash equilibriums (SNEs) are vertex cover states of the network, named as memory-based best response (MBR) rule. The memory has two effects: (i) guarantee the game converging to an SNE and (ii) help to obtain a better SNE (see Section III.*A* for detail), which is actually a state of the art algorithm using only local information. Recently, the MBR is extended to weighted vertex cover problem [19].

The proposed GMA-MVC belongs to memetic algorithms (MAs), which use both local and global information.

B. Introduction to MAs

By using the notion of meme defined as a unit of cultural evolution [20],[21], MAs have been introduced as population-based global optimization with local refinement [22],[23]. The local refinement is often referred to as local search, lifetime learning or individual learning in different literatures [23]. MAs emphasize the balance between exploration by global search and exploitation by local refinement.

MAs have successfully applied for solving many complex problems in recent years [24]-[32]. An MA for multimodal nonseparable problems is proposed in [24], where a particle swarm optimizer is used for local search and a population-based harmony search is used for global search. In [25], a multi-restart MA framework for box-constrained problems is proposed, where an EA and a local optimizer are implemented

Manuscript received , 2017; revised . This work was supported by Natural Science Foundation (Nos. 61572383, 61571342, 61472306 and 61672405).
Wu and Shen are with the School of Electronic Engineering, Xidian University, Xi'an, China (jshwu@mail.xidian.edu.cn).
Jiao is with the Baidu.com Times Technology (Beijng) Co.,Ltd.

as separated building blocks. In Ref.[26], a MA for the location-oriented continuously operating reference stations placement problem is provided, which ingeniously combine repairing genetic algorithm and the effective heuristic algorithm as global search and local search respectively. In Ref.[27], an adaptive MA for multi-objective optimization problems is designed, in which the global optimization is the synergy of a GA, differential evolution (DE), and estimation of distribution algorithm (EDA) and the local search is the evolutionary gradient search algorithm. An EDA-based MA is proposed for the distributed assembly permutation flow-shop scheduling problem in [28], in which the EDA with a selective-enhancing sampling mechanism and the critical-path-based local search strategy are cooperated. A three-phase MA is proposed for real-time combinatorial stochastic simulation optimization problems with large discrete solution space in [29], the exploration stage (phase 1) uses an EA with a rough surrogate model to identify a set of likely solutions, the exploitation stage (phase 2) uses a local search assisted by a refined surrogate. A MA is proposed for the multidepot and periodic vehicle routing problem, in which the intelligent initialization and stochastic learning techniques are employed for the balance between exploration and exploitation [30].

MAs have successfully used in solving many other problems [31]-[38]. For surveys of MAs, please refer to [21] and [23].

C. *Introduction to MAs for Vertex Set Problems*

Similar as the MVC problem, there is a class of problems about a graph that aim[illegible]partitioning the vertices into two or more sets[illegible]reinafter referred to[illegible]ertex set problem. Other examples of the vertex set problem include the minimum dominating set problem, the maximum independent set problem, the maximum clique problem, and the set-packing problem [39].

MA is problem-specific. When applying an MA to the vertex set problem, there are two key issues that need to be well addressed simultaneously.

The first is how to deal with constraints. The vertex set problem usually has a constraint on the vertex set. So the challenge is how to apply an EA (with crossover and mutation operators) to a constrained optimization problem while ensuring non-violation of the constraint. There are two main approaches for handling the constraint problem in the literatures. (i) Converter to an unconstrained optimization problem by adding a penalty item to the fitness function. Infeasible solutions have lower fitness and feasible solution have higher fitness. For example, the minimum weight dominating set problem is converted to an unconstrained optimization problem by adding an adaptive penalty function [35]. Other examples can be found in [40]-[42]. This kind of approach is easy to be used, but it is difficult to decide the weight of penalty (parameter selection) [42]. (ii) Preserve or repair: specialized operators to make the solutions limited in or return to the feasible space. For example the MVC problem in [14], the individuals generated by the crossover operator are feasible vertex cover sets. In the resources allocation problem on node-weighted graph [43], the individuals generated by the crossover operator are conflict-free (feasible) solutions. This approach needs special or additional operations that may decrease the efficiency.

The second is the effectiveness and efficiency of the local search. For the vertex set problems, the local search techniques are generally based on (k,l)-exchange of vertices to increase the fitness [44]-[46], $k,l=0,1,\cdots,\infty$. For example, a 1-flip process (one vertex removing from the solution set or adding to the solution set) for the minimum dominating set problem is proposed in [35], which is really a (1,0)- and (0,1)-exchange. A (k,1)-exchange is proposed for the maximum independent set problem in [44], $k=0,1,\cdots$. A k-change is presented for the multidimensional multi-way number partitioning problem [45], indicating totally k vertices are exchanged in one iteration, k=1,2,3. A (1,1)-exchange is proposed for the multidimensional two-way number partitioning problem in [46]. A (2,1)-exchange is presented for the maximum clique problem [47]. Usually an improvement by a (k,l)-exchange cannot be obtained by (m,n)-exchanges in those problems ($k\neq m$, $l\neq n$). The difficulty is that we can not traverse k and l from 1 to ∞. So a more effective local search scheme is needed. Some of the methods may need to compute the global fitness value in the (k,l)-exchange, which made the local search non-local (global information is needed).

D. *Introduction to the Proposed GMA-MVC*

In the proposed GMA-MVC of this paper, the local search is carried out by the snowdrift game evolution and the global optimization technique is the EA. The SNEs, being the results of the game evolution using only local information, are encoded as the chromosomes of the population. The crossover and mutation operations of the EA are skillfully combined with the game evolution, and help those SNEs to escape from local optima to reach better SNEs.

The proposed GMA-MVC alternatively repeats by two stages: individual (game) evolution and population evolution. Each individual of the GMA-MVC describes a game state of a network (see Section III for detail). The individual evolution has two steps: (1) Each individual evolves by the proposed asynchronous updating rule to the SNE. As mentioned above, an SNE is a vertex cover state of the network. (2) Each SNE is further improved to a better SNE by the individual local evolution where only a small part of vertices update their strategies (see Section IV.*D* for detail). The crossover operator of the population evolution is a simple two-point crossover, since it does not need to deal with the constraint.

The contributions of this paper are summarized as follows.

(i) Asynchronous updating best response rule of snowdrift game is proposed as a local search technique for the MVC problem. Synchronous updating best response rule has been proposed for the MVC problem in [17], but it cannot be used as a local search. The reason is that the network is not guaranteed to converge to an SNE without memory (see Section III.*A* for details). We prove that the proposed asynchronous snowdrift game will always converge to an SNE without memory (see Theorem 1 in Section III.*A*).

The proposed game evolution as a local search technique do

改成improve更好

not need global information and can run in a distributed way, which guarantees its efficiency. The game evolution is completely different from the (k,l)-exchange based local search techniques (see Section IV.*B* for detail), which provides a new way to increase the effectiveness of the local search.

(ii) A new scheme for handling constraint of the MVC problem. After the local search by game, the individuals are feasible solutions. The SNEs are at least a vertex cover of the graph. Since the constraint is naturally guaranteed in the individual evolution, we do not need special crossover operator or additional repair operations. Thus more effective and efficient crossover operator can be used in the EA.

(iii) A degree-based initialization method. In previous algorithms for the MVC problem, including the GA [9], HGA [14], and MBR [17], the initialization methods are uniform random. In this paper, degree-based initialization method is proposed (see Section IV.*A*), which is much better than the uniform random method.

As a result, an effective and efficient MA is provided for the MVC problem. The rest of this paper is structured as follows. Section II is preliminaries. In Section III, related algorithms and motivations of the GMA-MVC are introduced. Analyses about the individual evolution are provided in Section IV. The detailed process of the GMA-MVC is provided in Section V. Extensive experiments are done in Section VI, where the GMA-MVC is compared with several recent algorithms on various networks. Section VII concludes the paper.

II. Preliminaries

A. The MVC Problem

An undirected graph is denoted as $G = (V, E)$, where $V = \{1, 2, \dots, N\}$ is the set of vertices and $E = \{e_{ij}\}$ is the set of edges. Suppose V_{VC} is a subset of V, such that for any $e_{ij} \in E$, there is $i \in V_{VC}$ or $j \in V_{VC}$. In other words, at least one endpoint (i or j) of e_{ij} is an element of V_{VC}, then V_{VC} is a vertex cover of the network. A minimum vertex cover, denoted as V_{MVC}, is a vertex cover with the minimum number of vertices. Fig. 1 shows a simple illustrative network, where the white vertices in Fig. 1(a) is a V_{VC} of the network and the white vertices in Fig. 1(b) is a V_{MVC}.

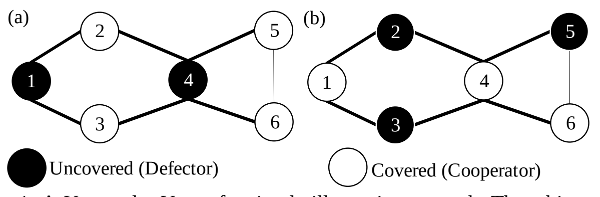

Fig. 1. A V_{VC} and a V_{MVC} of a simple illustrative network. The white vertices represent covered (cooperator) while the black ones are uncovered (defector). (a) $V_{VC} = \{2, 3, 5, 6\}$. (b) $V_{MVC} = \{1, 4, 6\}$.

B. Snowdrift Game

A game describes an interactive decision situation with conflicting interests [48]. As mentioned in Section I, the snowdrift game is employed in the GMA-MVC as the individual evolution. In a snowdrift game, each agent has two-strategies: C (cooperation) and D (defection). Imagine that two cars are trapped at opposite sides of a snowdrift. Both drivers have two options: get out of the car to shovel (cooperation C) or stay in the car (defection D). They both will gain the benefit b if getting home. Shoveling the snowdrift will cost a labor c. Thus, if both drivers cooperate, they share the cost and each of them gains b-c/2; if both drivers choose D, they will never go home and get nothing; if one driver shovels (C) whereas the other stays in the car (D), then both of them can get home but the defector gains a perfect payoff b without cost, the cooperator gets only b-c. Without losing generality, normalize b-c/2 to one, so that the snowdrift game can be described with a single parameter: the so called cost-to-benefit ratio $r = c/(2b$-$c)$. Hence, the utility (payoff) matrix of the snowdrift game is

$$\begin{array}{ccc} & C & D \\ C & 1 & 1-r \\ D & 1+r & 0 \end{array}$$

where $0 < r < 1$ is a constant.

In spatial game, each agent only interacts with its neighbors and the population is structured [49]. The structure of population is usually described by a graph or a network [48]. An agent is mapped to a vertex in the network, and the interaction between a pair of agents (vertices) is the edge between them. In this paper, the agents play the snowdrift game with their directly connected neighbors and obtain the accumulated utility.

C. Nash Equilibrium

Suppose N agents play the snowdrift game on a network. Each agent only plays with his neighbors and wants to improve his utility. Let x_i denote the strategy (C or D) that vertex i adopts, $x_i \in S_i$, where S_i is the strategy set of agent i. Obviously for the snowdrift game, $S_i = \{C, D\}$ for all i from 1 to N. Denote the utility of vertex i (agent i) as $U_i(C)$ and $U_i(D)$ respectively when he cooperates or defects with all his neighbors. If $U_i(C) > U_i(D)$, he chooses C as his strategy and vice versa, which is known as the best response updating rule.

Define vector $X=(x_1, x_2, \dots, x_N)$, which describes the strategies of all the N agents. The strategies of the N-1 agents except i are described as $x_{-i} = (x_1, x_2, \dots, x_{i-1}, x_{i+1}, \dots, x_N)$. If agent i adopts strategy x_i and the rest agents' strategies are x_{-i}, its utility function is $U_i(X) = U_i(x_i, x_{-i})$.

A Nash equilibrium $X^* = \{x_1^*, x_2^*, \dots, x_N^*\}$ is such a state that no a single agent can increase his utility by changing its strategy [48], in formula,

$$U_i(x_i^*, x_{-i}^*) \ge U_i(x_i', x_{-i}^*)\,. \quad (1)$$

If (1) holds strictly for every $x_i' \neq x_i^*$, X^* is a strict Nash equilibrium (SNE).

As shown in Fig. 1, let a V_{VC} be mapped into a number of cooperators in the network, whereas the rest vertices, also known as uncovered vertices, be mapped into defectors. If r=0.2, it is found that both the two game states (Fig.1(a) and (b)) are SNEs.

Given an undirected network, define $V_{SNE} = \{x_i^* | \, x_i^*=C, x_i^* \in X^*\}$, where $X^* = \{x_1^*, x_2^*, \dots, x_N^*\}$ is an SNE of the spatial snowdrift game. In other words, V_{SNE} is the set of vertices correspond to the cooperators of an SNE on the network. Let $\{V_{MVC}\}$ denotes all the V_{MVC}, $\{V_{SNE}\}$ denotes all the V_{SNE}, $\{V_{VC}\}$ for all the V_{VC}, and k_{max} is the maximum degree of the network.

Lemma 1 [17]: If $r\,k_{max}<1$, then $\{V_{MVC}\}\subseteq\{V_{SNE}\}\subseteq\{V_{VC}\}$. Specifically, each V_{MVC} corresponds to a V_{SNE} of the game, and each V_{SNE} corresponds to a V_{VC} of the network.

A detailed proof of Lemma 1 can be found in Ref.[17].

D. Sequence of Update

In spatial game, the agents may change their strategies synchronously or asynchronously [48]. In synchronous updating rule, all the nodes compute their best responses and update their strategies simultaneously in one time step. On the contrary, in asynchronous updating rule, the nodes compute and update their strategies one by one. In other words, after a node has updated its strategy, the next node computes its best response. When all the nodes have updated their strategies, an iteration step is finished. The dynamics of synchronous and asynchronous are dramatically different.

Remark 1. Note that from any initial states, the network is not always guaranteed to converge to an SNE by (1). The game states may be trapped into oscillation (see Table II in Section IV for an example). That is a key to design the updating rule of game for individual evolution.

III. Related Algorithms and Motivations

Several representative and state of the art algorithms are introduced in this section, which facilitates to make clear the motivations of this paper.

A. Related algorithms

GA [9]. The chromosome of the GA is a string of 0 and 1. Take the network shown in Fig.1(a) for an example, the chromosome is 011011, indicating V_{VC} = {2,3,5,6}. The initialization method is uniform random: give 0 or 1 randomly to each bit of the string. Each individual is assigned an index by the following formula.

$$index(i) = q\text{-}number(i) \quad i=1,\dots,N, \tag{2}$$

where *number*(*i*) stands for the number of uncovered edges of individual *i* and the *index*(*i*) is the index of individual *i*, *q* is a constant and *q* is not less than the number of edges of the network. The index will be larger if the number of uncovered edges is less. Half of the chromosomes with larger indices are selected, and the rest are removed. Then new chromosomes are generated by the crossover and mutation operation. The fitness of each chromosome is calculated by Eq.(3) and reproduction is carried out by Roulette Wheel Selection.

$$f(X) = 1/(1+e^{-\lambda}), \tag{3}$$

where X stands for a chromosome and λ is the number of "1" in the chromosome. The results of Roulette Wheel Selection depend on the fitness value.

In the GA, no local information is used. The crossover is not specially designed for the constraint. Eq.(2) can be regarded as the penalty function, which is separated with the fitness function Eq.(3). Feasible solutions have the highest value of index calculated by Eq.(2), but is not guaranteed that solutions obtained by GA are always feasible.

Fitness function Eq.(3) has a disadvantage when the number of λ becomes large. For example, the fitness value is 0.999955 when $\lambda = 10$ and is 1.000000 when $\lambda = 20$. The difference is too small, thus it has not ar[illegible]ect on the selection operation on large scale networks. So in experiments of the GA, the fitness is calculated by Eq.(4) (will be introduced in Section IV) instead of Eq.(3).

改为but solutions which obtained by GA could not been guaranteed as always feasible.

HGA [14]. The key of the HGA is the combination of a local search technique with a GA. The local search technique in the HGA is very simple. Let us take the network in Fig. 1(b) to illustrate. For chromosome 100111, V_{VC}={1,4,5,6}. If vertex 5 is removed from V_{VC}, chromosome 100101 is still a feasible solution, then 100111 is replaced by 100101. That is the local search technique.

The local search technique of HGA is equivalent to a (1,0)-exchange: removing a vertex from the covered set and checking its feasibility. Obviously, it cannot realize a (2,1)-exchange, (3,1)-exchange, (3,2)-exchange, etc. It is necessary to design a local search that can realize all those (*k*,*l*)-exchanges, $k,l=0,1,\cdots,\infty$. Since we cannot traverse *k* and *l* from 0 to ∞, a new scheme of local search other than a (*k*,*l*)-exchange is required, that is the motivation of individual evolution by game as the local search.

Another difference between the HGA and the GA is the crossover operator. The operator in the HGA selects vertex with the highest degree one by one from the parents to their offspring, until all edges are covered. The crossover operator is specially designed for the constraint, which guarantees the feasibility of solutions, but decreases its efficiency.

VCAR[8]. The algorithm can be described by rough set theory. It proves that finding the V_{MVC} of a network can be translated into finding the attribute reduction of a decision information table. The VCAR is designed by attribute reduction. In fact, the procedure of the VCAR is repeatedly moving the nodes with the highest degree into V_{VC} and deletes the connected edges. Till all the edges are deleted, the nodes in V_{VC} cover the edges. Of course, finding the nodes with the highest degree requires global information of the network.

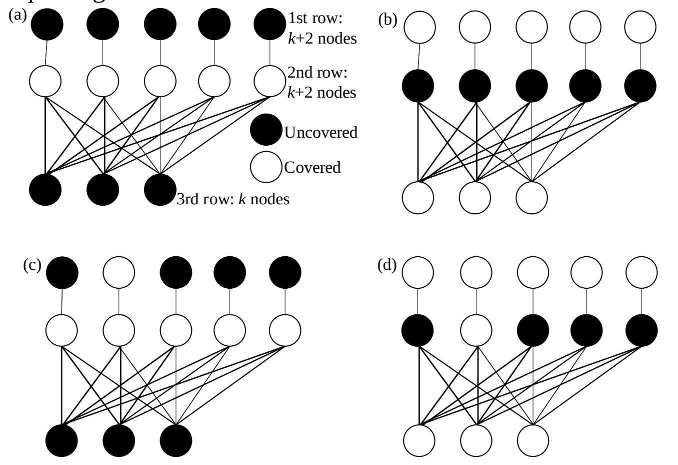

Fig. 2. The V_{VC}, V_{SNE}, and V_{MVC} of the PS graph with k=3. (a) V_{SNE} and V_{MVC}. (b) V_{SNE} and V_{VC} but not V_{MVC}. (c) V_{VC} but not V_{SNE}. (d) V_{VC} but not V_{SNE}.

The Papadimitriou-Steiglitz (PS) graph is usually employed to evaluate the performance of an algorithm for the MVC problem [9],[50]. As shown in Fig. 2(a), a standard PS graph has three rows: two rows of *k*+2 vertices and one row of *k* vertices, $k\geq1$. Thus N=3k+4 and there are totally (k+2) (k+1) edges. Each vertex in the first row has an edge to the vertex of the second row in the same column. Each vertex in the second row has connections with every vertex of the third row. Take the network shown in Fig.2(a) as an example for the VCAR, the 3 nodes in the third row are moved into V_{VC} firstly and then 5 nodes in the first and second rows. The obtained solution is

TABLE I
EXAMPLE ILLUSTRATION OF THE MBR ON THE NETWORK IN FIG. 1

Best response of the vertices							States in the memory (*ml*=3) (*φ* indicates empty)						
Vertex / Iteration	1	2	3	4	5	6	Vertex / Memory	1	2	3	4	5	6
Initial state	*C*	*D*	*D*	*D*	*D*	C	1	*φ*	*φ*	*φ*	*φ*	*φ*	*φ*
							2	*φ*	*φ*	*φ*	*φ*	*φ*	*φ*
							3	*φ*	*φ*	*φ*	*φ*	*φ*	*φ*
1	*C*	*C*	*C*	*C*	*C*	C	1	*C*	*C*	*C*	*C*	*C*	*C*
							2	*φ*	*φ*	*φ*	*φ*	*φ*	*φ*
							3	*φ*	*φ*	*φ*	*φ*	*φ*	*φ*
	X(1): choose from memory							*C*	*C*	*C*	*C*	*C*	*C*
2	*D*	*D*	*D*	*D*	*D*	*D*	1	*D*	*D*	*D*	*D*	*D*	*D*
							2	*C*	*C*	*C*	*C*	*C*	*C*
							3	*φ*	*φ*	*φ*	*φ*	*φ*	*φ*
	X(2): choose from memory							*C*	*D*	*D*	*D*	*D*	*D*
3	*C*	*C*	*C*	*C*	*C*	*C*	1	*C*	*C*	*C*	*C*	*C*	*C*
							2	*D*	*D*	*D*	*D*	*D*	*D*
							3	*C*	*C*	*C*	*C*	*C*	*C*
	X(3): choose from memory							*C*	*C*	*C*	*C*	*C*	*D*
4	*D*	*D*	*D*	*C*	*C*	*D*	1	*D*	*D*	*D*	*C*	*C*	*D*
							2	*C*	*C*	*C*	*C*	*C*	*C*
							3	*D*	*D*	*D*	*D*	*D*	*D*
	X(4): choose from memory							*D*	*D*	*D*	*D*	*D*	*C*
5	*C*	*C*	*C*	*C*	*C*	*C*	1	*C*	*C*	*C*	*C*	*C*	*C*
							2	*D*	*D*	*D*	*C*	*C*	*D*
							3	*C*	*C*	*C*	*C*	*C*	*C*
	X(5): choose from memory							*C*	*C*	*C*	*C*	*C*	*D*
6	*D*	*D*	*D*	*C*	*C*	*D*	1	*D*	*D*	*D*	*C*	*C*	*D*
							2	*C*	*C*	*C*	*C*	*C*	*C*
							3	*D*	*D*	*D*	*C*	*C*	*D*
	X(6): choose from memory							*D*	*C*	*D*	*C*	*C*	*D*
7	*C*	*C*	*C*	*C*	*C*	*D*	1	*C*	*C*	*C*	*C*	*C*	*D*
							2	*D*	*D*	*D*	*C*	*C*	*D*
							3	*C*	*C*	*C*	*C*	*C*	*C*
	X(7): choose from memory							*C*	*D*	*C*	*C*	*C*	*D*
8	*C*	*D*	*D*	*C*	*C*	*D*	1	*C*	*D*	*D*	*C*	*C*	*D*
							2	*C*	*C*	*C*	*C*	*C*	*D*
							3	*D*	*D*	*D*	*C*	*C*	*D*
	X(8): choose from memory							*C*	*C*	*D*	*C*	*C*	*D*
9	*C*	*D*	*D*	*C*	*C*	*D*	1	*C*	*D*	*D*	*C*	*C*	*D*
							2	*C*	*D*	*D*	*C*	*C*	*D*
							3	*C*	*C*	*C*	*C*	*C*	*D*
	X(9): choose from memory							*C*	*D*	*D*	*C*	*C*	*D*
10	*C*	*D*	*D*	*C*	*C*	*D*	1	*C*	*D*	*D*	*C*	*C*	*D*
							2	*C*	*D*	*D*	*C*	*C*	*D*
							3	*C*	*D*	*D*	*C*	*C*	*D*

$|V_{VC}|$=8, but $|V_{MVC}|$= 5. The VCAR cannot obtain the MVC for the PS graphs.

MBR[17]. MBR is a spatial snowdrift game based optimization algorithm utilizing only local information. As shown in Lemma 1:$\{V_{MVC}\}\subseteq\{V_{SNE}\}\subseteq\{V_{VC}\}$, the sets of V_{MVC} are among the sets of V_{SNE} of the snowdrift game. With synchronous updating rule, the network will always converge to an SNE with memories (Theorem 2 in [17]). Usually, the number of SNEs is much larger than the number of V_{MVC}. An SNE is usually a local optimum of the MVC problem. The main idea of MBR is using memory not only for helping the game escaping from oscillation but also for a better SNE.

All the agents calculate their best response strategies synchronously, and then store into their memories respectively. Let the memory length is denoted as *ml*, the memory stores the strategies of the nearest *ml* updating steps of the agent (vertex). The oldest strategies are abandoned. Instead of adopting the best response directly in the game, each agent randomly chooses a strategy from his memory for the next step. Repeat the process until the states of all the agents never change, an SNE can be reached. Table I illustrates a complete evolution procedure of the MBR on the 6-vertex network in Fig. 1 from a random initial state: *X*(0)=(*CDDDDC*). After 10 iterations, the states stored in the three memories are equal and the game converges to an SNE.

MBR uses only local information, which is its key weakness from the viewpoint of an MA design. Without memory, the synchronous updating scheme of MBR may result in oscillation (see Section IV). With a longer memory, a better solution may be obtained. But a long memory also increases the difficulty to converge, especially for modular networks (see the experiments in Section VI).

SBTS. Recently, a (k,1)-s[illegible]-based tabu search (SBTS) is proposed for the maximum independent set (MIS) problem [51]. Since the relation between the MVC problem and the MIS problem: $V\backslash V_{MVC}= V_{MIS}$, the algorithm can be used to the MVC problem [51]. In the (k,1)-swap (exchange), k can be 0, 1, 2 or a number more than 2. The main operation is to insert one vertex

改为Assumed the memory length as ml

that are not in the V_{IS} (set of vertices in the independent set (IS)) into V_{IS}, and remove vertices ir that are adjacent to the vertex. The solution is improved if k=0; the solution is not improved but the structure is changed if k=1; the solution is degenerated and the (k,1)-swap is only a diversification if k>1. In SBTS, once a vertex is removed from the IS, the SBTS set a tabu list for this vertex to avoid it goes back to the IS again in the next t_t iterations, where t_t is the tabu tenure of the vertex. The algorithm iterated until a given number of iter ns. The SBTS uses only local information, but the exchange is guided by a tabu list.

B. Motivations

Motivation 1. Game-based techniques should be designed as local search and combined with other global optimization techniques, since only local information is used in game. But synchronous updating scheme in MBR is not suitable for a local search technique. The reasons are: (i) The network may fall into oscillation without memory; (ii) for a local refinement technique s usually enough for a few of vertices changing their strategies. All the vertices calculating their best response strategies and updating synchronously may increase the computation burden meaninglessly. So the motivation is developing asynchronous updating rule of game as the local search technique (we prove that it always converge to an SNE without memory in Section IV).

From Lemma 1, if $r\ k_{max}$<1, the {V_{MVC}} are included in those {V_{SNE}}. If all the SNEs are available, all {V_{MVC}} can be obtained, not to say one V_{MVC} solution. Because {V_{MVC}}⊆{V_{SNE} }, by searching a V_{MVC} only in those SNEs, many V_{VC} which are not SNEs are eliminated out of the searching territory.

Motivation 2. GA and VCAR use only global information for the MVC problem. MBR uses only local information. HGA uses both global and local information, but the local search is only a (1,0)-exchange, which is too inefficient. Moreover, the crossover operator in HGA is specially designed for constraint, which is not efficient. Since the constraint is naturally handled by the proposed individual evolution, the global optimization technique can be simple and efficient. The second motivation is balanced g both of local and global information.

Motivation 3. The method of initialization usually has an important effect on the MA [27],[30]. In Ref.[30], an intelligent initialization technique is proposed in a MA for the multidepot and periodic vehicle routing problem, which is very effective. The essence of the intelligent initialization technique is emphasizing both randomness and greediness. In their method, half of the initial population is generated by randomness and the other half is by a heuristic. In the GA, HGA, and MBR, uniform random initialization is employed. The VCAR is repeatedly moving the nodes with the highest degree into V_{VC} and deletes the connected edges, which is really a greedy procedure. Inspired by the intelligent initialization, the third motivation is to balance the using of both of randomness and greediness in the initialization.

改为 the third motivation is to achieve the balance between the randomness and the greediness in the initialization.是否可以？

IV. Individual Evolution as Local Search

A. Individual Evolution by Game

Lemma 2. If $r\ k_{max}$<1, the necessary and sufficient condition for an SNE is: For each vertex i, (1) if x_i=D, then it has only cooperative neighbors, and (2) if x_i=C, then it has at least one defective neighbor.

Proof. *Sufficient part*. For each vertex i, (1) if x_i=D and it has only k_i cooperative neighbors, then $U_i(C, x_{-i}) = k_i$ and $U_i(D, x_{-i}) = k_i(1+r)$ from the payoff matrix (see in Section II). Because $U_i(D, x_{-i}) > U_i(C, x_{-i})$, the agent will not change its strategy from D to C; (2) if x_i=C and it has m defective neighbors (m⩾1) along with k_i-m cooperative neighbors, then $U_i(C, x_{-i}) = (k_i-m)+ m(1-r)$ and $U_i(D, x_{-i}) = (k_i-m)(1+r)$ from the payoff matrix. Because $U_i(D, x_{-i}) - U_i(C, x_{-i}) = m-k_ir > 0$, the agent will not change its strategy from C to D. From the definition, X=(x_1, x_2, … , x_N) is an SNE.

Necessary part. Suppose X=(x_1, x_2, … , x_N) is an SNE, then $U_i(x_i, x_{-i}) > U_i(x_i', x_{-i})$ for each i from 1 to N (x_i'=x_i). For each i with m defective neighbors and k_i-m cooperative neighbors: (1) If x_i=D, then $U_i(D, x_{-i}) =(k_i-m) (1+r)$ and $U_i(C, x_{-i})= (k_i-m)+ m(1-r)$. From $U_i(D, x_{-i}) > U_i(C, x_{-i})$, one has m=0, vertex i has only cooperative neighbors; (2) If x_i=C, then $U_i(C, x_{-i}) = (k_i-m)+ m(1-r)$, $U_i(D, x_{-i}) = (k_i-m)(1+r)$, and $U_i(D, x_{-i})-U_i(C, x_{-i}) =m- k_i\,r$. From $U_i(D, x_{-i}) > U_i(C, x_{-i})$, one has m⩾1, vertex i has at least one defective neighbor. This completes the proof.

In [17], it has proved that the snowdrift game will always onverge to an SNE when $r\ k_{max}$<1 by synchronous updating rule with memory length larger than or equals to 2 (see theorem 2 in [17]). We will show that the game may fall into oscillation without a memory.

Take the 6-vertex network in Fig. 1 for an example. Table II shows a lutionary process from a random generated initial state: X(0)=($CDDDDC$). With synchronous updating rule, the 6 vertices calculate their best response strategies simultaneously. After simple computation we obtain X(1)=($CCCCCC$), ⋯, etc. seen that X(1)= X(3) and X(2)= X(4) for synchronous updating rule, the state ped in X(1)→X(2)→X(1)→X(2), which is a periodic oscillation and r converge to an SNE. Thus synchronous updating scheme is not suitable for a local search.

With asynchronous updating rule, from the same initial state, vertex 1 calculates and changes to its best response strategy firstly, then vertex 2 calculates and changes to its best response strategy, ⋯, till vertex 6 changes its strategy, we obtain X(1)=($CCCCDC$). It is seen that X(2)=X(3)=X(4) for asynchronous updating rule, which is an SNE.

TABLE II
Difference Between Synchronous and Asynchronous Updating on the Network in Fig. 1

	Synchronous updating						Asynchronous updating					
vertex / iteration	1	2	3	4	5	6	1	2	3	4	5	6
X(0)	C	D	D	D	D	C	C	D	D	D	D	C
X(1)	C	C	C	C	C	C	C	C	C	C	D	C
X(2)	D	D	D	D	D	D	D	C	C	C	D	C
X(3)	C	C	C	C	C	C	D	C	C	C	D	C
X(4)	D	D	D	D	D	D	D	C	C	C	D	C

The individual evolution in the proposed GMA-MVC is snowdrift game with an asynchronous updating rule without memory.

Theorem 1. Given an undirected network, if $r\, k_{max}<1$, from any initial state $X(0)=(x_1(0), x_2(0), \ldots, x_N(0))$, the network's game state will converge with probability one to an SNE in asynchronous updating rule without memory.

Proof. From Lemma 2, if $X(0)$ is not an SNE, there are two cases in the network: (1) at least a *D-D* edge and/or (2) at least a vertex *C* surrounded by *C* (a cooperator has only cooperative neighbors). In asynchronous updating scheme, both the *D-D* edge and the cooperative surrounded *C* will disappear in the next step $X(1)$. If there is no new *D-D* edge and cooperative surrounded *C* appear in $X(1)$, then $X(1)$ is an SNE. In other words, there is a positive probability that $X(1)$ is an SNE.

The same reason, if $X(1)$ is not an SNE, there is a positive probability that $X(2)$ is an SNE. In general, if $X(k)$ is not an SNE, there is a positive probability that $X(k+1)$ is an SNE. Thus the network will converge with probability one to an SNE.

Remark 2. Theorem 1 guarantees the convergence of the proposed individual evolution rule. Without memory, the computation burden is also decreased.

B. Individual Evolution for (k,l)-exchange

The local search of the HGA is only a (1,0)-exchange [14], which is equivalent to removing a vertex from the solution set and checking its feasibility. In this subsection, examples are given to show that the proposed individual evolution can implement (k,l)-exchange for different values of k and l on the network in Fig. 3. The asynchronous updating sequence in the individual evolution is simply 1-2-3-4-5-6-7-8 for all examples.

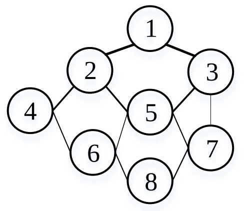

Fig. 3. An illustrate network.

(i) (2,1)-exchange: removing 1, 4, and adding 7.
(*CCCCDCDD*)→(*DCCDCCCD*)→(*DCCDDCCD*)

(ii) (2,1)-exchange: removing 2, 6, and adding 3.
(*CCDCCCDC*)→(*CDCCCDDC*)

(iii) (3,1)-exchange: removing 1, 4, 8, and adding 3.
(*CCDCDCCC*)→(*CCCDDCCD*)→(*DCCDDCCD*)

(iv) (3,0)-exchange: removing 4, 5, and 8.
(*DCCCCCCC*)→(*DCCDDCCD*)

(v) (1,0)-exchange: removing 2.
(*CCDCCDCC*)→(*CDDCCDCC*)

C. Search Ability of Individual Evolution

For convenience to analyze, we separate the searching ability of the proposed GMA-MVC into two parts: the searching ability of the local search and the searching ability of the global optimization. From any initial state, the network will converge to an SNE after the first step of individual evolution. So the searching ability of the local search can be estimated by the number of SNEs, which is related with the topology of the network. The smaller the number of SNEs is, the higher the searching ability.

改为Consider the convenience of analyze,

Take the PS graph in Fig. 2 for an example. As shown in Fig. 2(a) and (b) respectively, there are only two SNEs: all the vertices in the second row cooperate (Fig. 2(a)) and all the vertices in the first and third row cooperate (Fig. 2(b)). Obviously, Fig. 2(a) is a V_{MVC} which has only k+2 vertices ($|V_{MVC}|=k+2$). Though Fig. 2(b) is an SNE, it is not a V_{MVC} and has 2k+2 vertices in the set. Change one or more vertices of Fig. 2(a) from defection into cooperation (see Fig. 2(c)), the obtained state is also a vertex cover but is not an SNE anymore by Lemma 2. Identically, change one or more vertices of Fig. 2(b) from defection into cooperation (see Fig. 2(d)), the obtained state is also a V_{VC} but is not an SNE by Lemma 2. Assuming the probabilities to converge to the SNEs are equal, for the PS graph, from any one of the 2^N initial states, after the first step of individual evolution, there is half possibility to obtain the V_{MVC}. That is the searching ability of the individual evolution for the PS graph.

Fig. 4 is another simple illustrative example: 10-vertex ring, which has only two V_{MVC}: V_{MVC} = {1, 3, 5, 7, 9} (shown in Fig. 4(a)) and V_{MVC} = {2, 4, 6, 8, 10}. But there are totally 16 SNEs with 14 of them are only local optima: V_{VC} = {1, 3, 5, 6, 8, 10} (Fig. 4(b)); V_{VC} = {1, 3, 5, 7, 8, 10} (Fig. 4(c)); V_{VC} = {1, 3, 4, 6, 8, 10}; V_{VC} = {1, 2, 4, 5, 7, 9}(Fig. 4(d)); V_{VC} = {1, 2, 4, 6, 7, 9}; V_{VC} = {1, 2, 4, 6, 8, 9}; V_{VC} = {2, 3, 5, 6, 8, 10}; V_{VC} = {2, 3, 5, 7, 8, 10}; V_{VC} = {2, 3, 5, 7, 9, 10}; V_{VC} = {1, 3, 4, 6, 7, 9}; V_{VC} = {1, 3, 4, 6, 8, 9}; V_{VC} = {2, 4, 5, 7, 8, 10}; V_{VC} = {2, 4, 5, 7, 9, 10}; V_{VC} = {1, 3, 5, 6, 8, 9}.

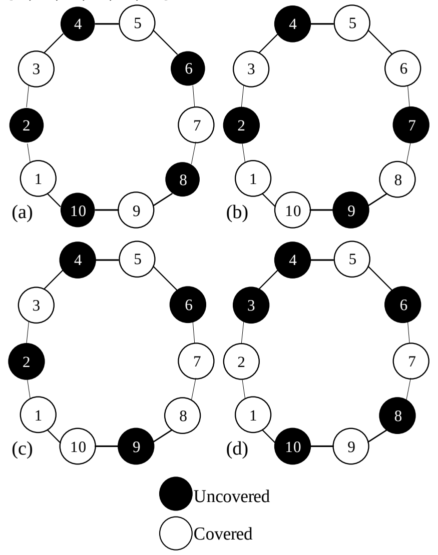

Fig. 4. The SNEs in the10-vertex ring.

From any one of the 2^{10} initial states, the snowdrift game on the 10-vertex ring converges to one of the 16 SNEs. Suppose the probabilities of converging to the 16 SNEs are equal, then the probability to reach the two V_{MVC} is 2/16 = 12.5% in one run of the first step of individual evolution. That shows the searching ability of the individual evolution.

D. Individual Local Evolution

In the first step of individual evolution, the individual is just evolved from a non-SNE to an SNE. All the vertices are involved in the evolution in general. From an SNE, the individual can further evolve locally. In other words, only a few vertices are involved in the evolution. We denote the vertices in V_{SNE} that have only one edge to a vertex in V/V_{SNE} as V^1_{SNE}. example in Fig. 5(a), V_{SNE}={1,3,4,6}, V^1_{SNE}={1,3,6}. Denote the vertices in V/V_{SNE} that have at least two edges connecting to two vertices in V^1_{SNE} as V_{CAN}. For example in Fig. 5(a), vertex 5 has two edges to vertices 3 and 6 in V^1_{SNE}, V_{CAN}={5}.

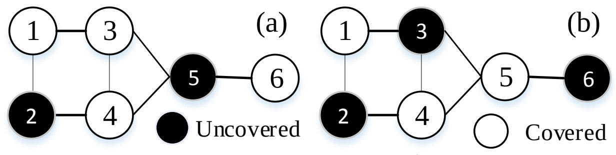

Fig. 5. A network illustrating V^1_{SNE} and V_{CAN}.

The process of local evolution is as follows.

Step 1: choose a vertex *j* randomly in V_{CAN} and change its state from *D* to *C*.

Step 2: play the game asynchronously for the neighbors of *j*. Theorem 2 will prove that the local game is enough to guarantee the convergence to an SNE.

Step 3: update V^1_{SNE} and V_{CAN}, and repeat from step 1 imes.

For example in Fig. 5(a), vertex 5 in step 1 is chosen, then plays the game in sequence 3-4-6-5 (4-3-6-5 or others) in step 2, the SNE shown in Fig. 5(b) is obtained: V_{SNE}={1,4,5}, which is an improvement of V_{SNE}={1,3,4,6}.

Theorem 2. From an SNE, change the strategy of a vertex *j* in V_{CAN} from *D* to *C*, and play the snow drift game for the neighbors of *j* , the game will converge to an SNE.

Proof. From Lemma 2, vertex *j* has only cooperative neighbors. We separate the neighbors of *j* into two sets: $N_1 \in V^1_{SNE}$ ($|N_1| \geqslant 2$) and $N_2 \in V_{SNE}/V^1_{SNE}$ er *j* changed from *D* to *C*, the vertices in N_2 have at least a defective neighbor, whereas the vertices in N_1 have no defective neighbor. Thus the strategies of the vertices in N_2 will not change emma 2 (and the proof). The strategies of both the two (or more) vertices in N_1 will change to *D* if the two (or more) vertices are not connected (because they have only cooperative neighbors, see the proof of lemma 2), or at least one vertex in N_1 will change to *D*. The strategies of the rest vertices are not affected and the game converges to an SNE by Lemma 2.

In fact, the individual local evolution realize a (*k*, 1)-exchange by only updating the strategies of the neighbors of the vertex in V_{CAN}, $k \geqslant 1$.

V. The Proposed GMA-MVC

The proposed GMA-MVC has two repetitive stages: individual evolution and population evolution. In the individual evolution stage, each individual evolves by the snowdrift game to an SNE of the network and then evolves locally for *le* times. In the population evolution stage, crossover and mutation are operated to those SNEs, offspring individuals are generated. Then repeat the individual evolution, each individual will reach to an SNE again by the snowdrift game and evolve locally for *le* times.

A. Elements of the GMA-MVC

Individual. In general sense, an individual is a sequence of the strategies (*C* or *D*) corresponding to the vertices, which describes a game state of the network. For example, the states shown in Fig. 1(a) and (b) are described by individual X_1= (*D C C D C C*) and X_2= (*C D D C D C*).

Individual evolution. Each individual is regarded as a state of the snowdrift game and the agents evolve by the asynchronous updating rule until an SNE is reached, and then evolves locally for *le* times. The result of the individual evolution is an SNE.

An SNE is also a sequence of the strategies corresponding to the vertices of the network, which describes a specific game state. In narrow sense, an individual indicates an SNE corresponding to the individual in general sense it evolves from. After all the randomly generated individuals evolve to SNEs by the snowdrift game, we may say that the individuals of the population are SNEs. For clarity, in the following of this paper, if an individual is an SNE, we say it an SNE directly.

Fitness function. To evaluate a solution *X*, the fitness function *f*(*X*) used in the GMA-MVC is the same as in previous literature [53]:

$$f(X)=\sum_{i=1}^{N}\left(x_i+N(1-x_i)\sum_{j=i}^{N}(1-x_j)e_{ij}\right), \tag{4}$$

where x_i = 1 if $i \in V_{VC}$; otherwise x_i = 0. The first part of the fitness function counts the number of vertices in V_{VC}, and the second part penalizes any uncovered edge e_{ij} with the magnitude *N*. Thus, any covered solution has a lower fitness value than that of an uncovered solution. Of course for covered solutions, the lower the value of *f*(*X*) is, the smaller the number of vertices in V_{VC}.

In the HGA [14], the fitness function is only the first part of (4). Comparin th that in Ref. [14], Eq.(4) is more general. If the solution is a vertex cover of the network, the two fitness functions are equivalent; or else they are quite different. In the experiments of this paper, Eq.(4) is used as the fitness function of the HGA, the same as the proposed GMA-MVC.

Crossover operator. The crossover is operated on the SNEs instead of general individuals. Two-point crossover operation is employed in the GMA-MVC. Randomly select two points in one of the two parental SNEs, and exchange the part between the two points of the SNE with the corresponding part of another SNE, then two offspring individuals are generated.

Mutation operator. Every element of the individual changes its state (from *C* to *D* or from *D* to *C*) by the probability *mr*. Mutation is also a technique for the game to escape from a local optimal SNE.

Population evolution. All the individuals of the population evolve from the current generation to the next generation. Choose two SNEs from the population randomly and do the two-point crossover operation, then two offspring individuals are obtained. Perform mutation to the two offspring individuals with the given mutation rate. Perform individual evolution to the two offspring individuals, then two offspring SNEs are obtained. Among the two parental SNEs and the two offspring SNEs, the two with lower values of *f*(*X*) calculated by (4) are selected as two SNEs in the next generation population (**Selection**), the other two are rejected (**Replacement**). Choose another two SNEs from the population and repeat the above process until all the m_{size} SNEs of the next generation population are generated.

Initialization. In the GA, HGA, and MBR, uniform random initialization is employed. In the SBTS, the initial solution is also randomly generated. As we aim at covering all the edges of a network with as few vertices as possible, the GMA-MVC attempts to allocate *C* with higher probability to a vertex with larger degree. Precisely, the probability of allocating C to vertex i is calculated by

$$p_i = \frac{1}{K}\sum_{k_j \le k_i} k_j, \quad K = \sum_{i=1}^{N} k_i, \tag{5}$$

where k_i is the degree of vertex i and K is the summation of all the vertices' degrees. Namely, vertex i gets strategy C with probability p_i whereas D with probability 1- p_i. Simulations show that the proposed degree-based initialization can promote the performance of the GMA-MVC (see Section VI.*A*).

B. Procedure of the GMA-MVC

The flowchart of the GMA-MVC is shown in Fig. 6. The detailed procedure is as follows.

Input: Maximum number of generations: g_{max}, adjacency matrix of the network: A, population size: m_{size}, mutation rate: mr, times of local evolution: le, and cost-to-benefit ratio: r.

Output: A covered vertex set V_{VC} of the network.

Step 1: Initialization. Generate a population of m_{size} individuals according to the degrees of the vertices.

Step 2: Individual evolution. Each individual evolves by the asynchronous updating rule and converges to an SNE, and then evolves locally for le times.

Step 3: Repeat the population evolution g_{max} generations. The SNE with the lowest $f(X)$ in the last generation is the solution of the MVC problem.

In the GMA-MVC, individual evolution is embedded in population evolution; on the other hand, the results of population evolution start another round of individual evolution. Comparing with the MBR algorithm which has only the memory-based individual evolution, the GMA-MVC is easier to break away from a local optimum to obtain a better solution.

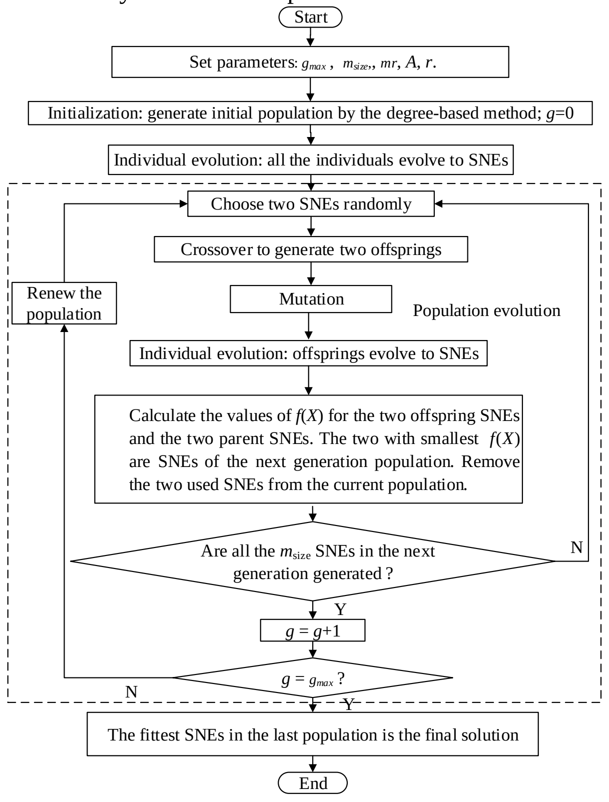
Start
Set parameters: g_{max} , m_{size}, mr, A, r.
Initialization: generate initial population by the degree-based method; g=0
Individual evolution: all the individuals evolve to SNEs
Choose two SNEs randomly
Crossover to generate two offsprings
Renew the population
Mutation
Population evolution
Individual evolution: offsprings evolve to SNEs
Calculate the values of $f(X)$ for the two offspring SNEs and the two parent SNEs. The two with smallest $f(X)$ are SNEs of the next generation population. Remove the two used SNEs from the current population.
Are all the m_{size} SNEs in the next generation generated ?
N
Y
$g = g$+1
$g = g_{max}$?
N
Y
The fittest SNEs in the last population is the final solution
End

Fig. 6. Flowchart of the GMA-MVC

VI. Experiments

Experiments are done to compare the GMA-MVC with state of the art algorithms, including the MBR [17], GA [9], HGA [14], VCAR [8], and SBTS [51]. The GA and HGA belong to EAs, the SBTS is a local search technique, and the MBR is a game-based algorithm.

For the snow drift [illegible]e in both the MBR and GMA-MVC, r=0.001 in all the experiments in this paper, which allows k_{max} =999 to satisfy $r\ k_{max} < 1$. The experiments are done on different sizes of the Barabási–Albert (BA) scale-free networks [54], Erdös–Rényi (ER) random networks [55], PS networks [9],[27], Watts and Strogatz (WS) small-world networks [56], and the Lancichinetti-Fortunato-Radicchi (LFR) modular benchmark networks [57]. The experiments are done on a computer with 2.53 GHz CPU and 2.0 G memory.

The BA scale-free network model is employed to generate scale-free networks synthetically [54]. Beginning with an initial network of m_0 vertices, a new vertex is added to the network at each step until the network size reaches N. Each new vertex is connected to $m \le m_0$ existing vertices with a probability that is proportional to the number of edges that the existing vertices already have [54], which is known as the preferential attachment mechanism.

The ER random network model has two parameters: the number of vertices (N) and the number of edges ($|E|$) [55], where edges are randomly distributed among the vertices. The number of edges can be equivalently given by the probability that each pair of vertices has an edge or by the average degree $<k>$.

The WS small-world model has three parameters: the number of nodes (N), the average degree $<k>$, and the rewiring probability p [56]. Beginning from a regular ring of N vertices, where each vertex is connected with its $<k>$ neighbors with $<k>/2$ on each side, rewire each edge by probability p, then a WS small-world network is generated.

The LFR modular network model is proposed to generate networks with modular structure, which captures the feature of many real world networks [57]. The LFR model has six parameters: the average degree $<k>$, the maximum degree k_{max}, the number of nodes N, the exponent for the degree distribution γ, the exponent for the distribution of community size β, and the mixing parameter μ [57].

A. Effects of the Initialization

In this subsection, experiments are done to verify the effect of the proposed degree-based initialization c[illegible]aring with the uniform random initialization in the GMA-MVC. In MBR [17], HGA [14], and GA [9], the initial individuals are generated randomly. The initial solution of the SBTS is also generated randomly [51]. In uniform random initialization, each vertex is set to C or D with equal probability. As shown in Table III, the simulations are done on four networks. One of the four networks is a WS small-world network with 200 vertices, average degree $<k>$=4 and rewiring probability p=0.1. One is a BA scale-free network with 1024 vertices and average degree $<k>$=8. Two of the four networks are ER networks with 500 and 2000 vertices respectively, and there average degrees are $<k>$=10 and $<k>$=8. In the simulations, the GMA-MVC uses different initialization methods but with the same parameters: m_{size}=100, mr=1/N, ls=10, and g_{max}=100.

The key parameters of the networks are also shown in the network names in Table III. For example, WS200<4> indicates the WS network with N=200 and <k>=4, etc. From Table III, it is seen that both the fitness and the runtime are improved by the degree-based initialization. Each value in Table III is an average over 30 runs. The number of iterations from the degree-based initial states to SNEs is decreased, that is the reason for less runtime in degree-based initialization.

Remark 3. In the experimental results of Section VI, the solutions are all feasible, thus the fitness $f(X)$ is really the number of vertices in V_{VC} (see Eq.(4)).

Table3和Table4表格内的字不一致，Table4中字体都靠上很多。

TABLE III

COMPARISON BETWEEN THE UNIFORM AND DEGREE-BASED INITIALIZATION: AVERAGE FITNESS $F(X)$ / STANDARD DEVIATION /AVERAGE RUNTIME (SECOND)

Networks	Uniform	Degree-based
WS200<4>	128.00/ 0/ 2.34	127.00/0/2.12
BA1024<8>	553.10/ 0.70/ 13.59	551.60/0.49/8.12
ER500<10>	347.10/ 0.30/ 2.67	346.70/ 0.64/ 2.63
ER2000<8>	1318.50/2.37/41.64	1316.70/2.15/40.64

For the WS200<4>, the uniform initialization yields 30 number of 128, whereas the degree-based initialization yields 30 number of 127. In other words, in all the 30 experiments, the probability that the degree-based initialization is strictly better than the uniform initialization is 100%. For the BA1024<8> network, the uniform initialization yields 9 number of 554, 15 number of 553, and 6 number of 552 in the 30 independent experiments, whereas the degree-based initialization yields 18 number of 552 and 12 number of 551. In other words, in all the 30 experiments, the probability that the degree-based initialization is not worse than the uniform initialization is 100%; the probability that the degree-based initialization is strictly better than the uniform initialization is (12/30)+(18/30)(24/30)=88%.

For the ER500<10> network, the uniform initialization yields 3 number of 348 and 27 number of 347, whereas the degree-based initialization yields 3 number of 348 and 15 number of 347 and 12 number of 346. The probability that the degree-based initialization is not worse than the uniform initialization is 100%; the probability that the degree-based initialization is strictly better than the uniform initialization is (12/30)+(15/30)(3/30)=45%. For the ER2000<8> network, after similar statistic process, the probability that the degree-based initialization is not worse than the uniform initialization is 100%; whereas the probability that the degree-based initialization is strictly better than the uniform initialization is 50%. Since the degree distribution of the ER network is uniform random, the improvement on the ER network is less significant than the WS and BA networks.

B. Effects of the Individual Evolution

In this subsection, experiments are done to verify the effect of the proposed individual evolution. As shown in Table IV, the simulations are done on the same four networks as in Table III. Each value in Table IV is an average over 30 runs. The GMA-MVC runs with individual evolution and without individual evolution respectively. The parameters are: m_{size}=100, mr=1/N, and g_{max}=100. Without individual evolution, the GMA-MVC deals with the constraint by the penalty item of the fitness function (see Eq.(4)).

改为 as shown in

From Table IV, it is seen that the fitness is dramatically improved by the individual evolution. It verifies again that both the local information and global information should be used, which is emphasized by an MA.

TABLE IV

GMA-MVC WITH AND WITHOUT INDIVIDUAL EVOLUTION: AVERAGE FITNESS F(X) /STANDARD DEVIATION/RUNTIME (SECOND)

Networks	With individual evolution	Without individual evolution
WS200<4>	127.00/ 0/ 2.12	130.20/ 0.60/ 0.12
BA1024<8>	551.60/0.49/8.12	733.53/ 7.43/ 0.58
ER500<10>	346.70/ 0.64/ 2.63	383.30/ 1.27/ 0.30
ER2000<8>	1316.70/2.15/40.64	1453.80/ 3.28/ 1.16

C. Comparisons between the GMA-MVC and MBR

Both the MBR and GMA-MVC employ the snowdrift game. The difference is that MBR is just a game with memory, whereas GMA-MVC is an MA.

In this subsection, extensive comparison simulations between the MBR and GMA-MVC are done on various networks: a synthetic WS small-world network with N=1000, <k>=4, and p=0.1; the real world Dolphin social network which has N=62 vertices [58]; a synthetic LFR modular network with N=500, <k>=15, k_{max}=50, γ=2, β=1, and μ=0.2; a synthetic BA scale-free network with N=2000，m_0=3, and m=3.

TABLE V

COMPARISON BETWEEN THE MBR AND GMA-MVC: AVERAGE FITNESS $F(X)$ /AVERAGE RUNTIME (SECOND)

Networks / Algorithms		WS 1000<4>	Dolphin	LFR 500<15>	BA 2000<4>
MBR	ml=10	654.20 /0.11	34.70 /0.06	333.17 /4.31	847.30 /1.12
	ml=10	643.50 /59.85	34.50 /19.19	329.00 /117486.00	841.30 /2.16
	ml=50	N/A	34.40 /4657.29	N/A	841.00 /6.54
GMA-MVC	g_{max} =10	640.20 /5.41	34.00 /0.13	329.60 /1.96	841.80 /9.84
	g_{max} =50	636.30 /19.32	34.00 /0.54	328.20 /8.42	840.70 /22.73
	g_{max} =100	635.90 /34.71	34.00 /1.00	328.00 /16.05	840.30 /38.22
	g_{max} =200	635.60 /71.90	34.00 /2.08	328.00 /31.92	840.30 /72.18

[a]*The underlined result is not the average of 10 runs, but only the one that converges in many runs.*

For the MBR, ml is the key parameter. For the GMA-MVC, g_{max} is the key parameter. So the experiments are done in variant values of ml and g_{max} (denoted in Table V). The rest two parameters of the GMA-MVC are: m_{size}=100, le=10, and mr = 1/N. Table V shows the average values of fitness $f(X)$ calculated by (4) and the average runtime over 10 runs except the underlined.

As shown in Table V, for the WS1000<4> network, the values of fitness ($|V_{VC}|$, see Remark 3) of the GMA-MVC are always smaller than that of the MBR. For the Dolphin network, the GMA-MVC obtains the optimal solution ($f(X)=|V_{MVC}|=34$) gmax=10? in all those runs even when $g_{max}=100$, but the MBR obtains 34 or 35 occasionally even when $ml=50$. The MBR obtains 34 in 3 of the 10 runs and 35 in the rest 7 runs when $ml=10$; when $ml=30$, the MBR obtains 34 in 5 runs and 35 in the rest 5 runs; when $ml=50$, the MBR obtains 34 in 6 runs and 35 in 4 runs. For the LFR500<15> modular network, the values of fitness of the GMA-MVC are always smaller than that of the MBR when $ml=10$. The MBR cannot converge when $ml=50$ and is difficult to converge when $ml=30$. We tested it many times with the runtime over 24 hours, only ones it converges when $ml=30$ and the result is underlined in Table V. For the BA2000<4> network, the MBR always runs faster than the GMA-MVC, but the GMA-MVC always obtains smaller number of vertices that cover all the edges.

In summary, in the aspect of the qualities of the obtained solutions, the GMA-MVC is much better than the MBR in all the four networks. The reason is that MBR uses only local information, but GMA-MVC uses both local and global information. In the aspect of the runtimes, they are network topology dependent. The MBR is difficult to converge on the LFR500<15> modular network and the real world Dolphin network, but runs fast on the BA500<6> scale-free network. The GMA-MVC is not network topology sensitive.

D. Comprehensive Comparison

In this subsection, experiments are done to comprehensively compare the mentioned six algorithms on various networks.

Parameters. The parameters of the four LFR modular networks are:
(1) LFR1000<15>: N=1000, <k>=15, k_{max}=50, γ=2, β=1, μ=0.2;
(2) LFR1000<20>: N=1000, <k>=20, k_{max}=60, γ=2, β=2, μ=0.3;
(3) LFR2000<20>: N=2000, <k>=20, k_{max}=50, γ=2, β=1, μ=0.2;
(4) LFR2000<25>: N=2000, <k>=25, k_{max}=60, γ=2, β=2, μ=0.3.
The parameters of the rest networks are shown in the network names or labeled under the names.

Since the six algorithms have different parameters, especially the game based and the population based algorithms. We attempted to explore the best ability of the algorithms through tuning the parameters. Detailed values of the parameters are given as follows.

GMA-MVC:	m_{size}=100, g_{max}=100, $mr = 1/N$, r=0.001, ls=10.
HGA:	m_{size}=100, g_{max}=1000, $mr = 1/N$.
GA:	m_{size}=100, g_{max}=1000, mr=0.1, pc=0.85.
MBR:	ml=50, r=0.001.

For the PS network, g_{max} is reduced from 100 to 10 for the GMA-MVC and is reduced from 1000 to 100 for the HGA and GA. Furthermore, le=1 for the GMA-MVC. The reasons are: (i) g_{max}=10 (with le=1) and g_{max}=100 are enough for the GMA-MVC and HGA to obtain the optimal solution, respectively; (ii) There is no improvement for the GA in larger value of g_{max}. For the SBTS, the maximum iteration times are 10000.

Results. Table VI shows the simulation results. Each of the algorithms runs 10 times on those networks, the average fitness $f(X)$, standard deviation, and average runtime are shown in Table VI.

TABLE VI
COMPREHENSIVE COMPARISON: AVERAGE FITNESS $F(X)$ /STANDARD DEVIATION/RUNTIME (SECOND)

Networks	GMA-MVC	MBR	HGA	GA	VCAR	SBTS
PS1000 |E|=111222	**334.00** /0 /2.09	**334.00** /0 /1.75	**334.00** /0 /962.22	666.00 /0 /1634.00	666.00 /0 /86400	**334.00** /0 /51.67
WS100 p=0.1 |E|=200	**64.00** /0 /1.12	65.40 /0.49 /4818.61	**64.00** /0 /23.44	94.80 /0.98 /25.30	87.00 /0 /0.10	**64.00** /0 /0.306
WS100 p=0.5 |E|=200	**55.00** /0 /0.19	57.20 /0.60 /440.99	57.10 /0.83 /20.83	89.00 /0 /25.30	89.00 /0 /0.11	**55.00** /0 /0.320
WS500 p=0.1 |E|=1000	**314.00** /0 /9.15	317.90 /0.7 /27328.90	316.30 /0.46 /340.01	484.20 /5.81 /572.60	493.00 /0 /7.45	315.87 /1.31 /2.002
WS1000 p=0.1 |E|=2000	**635.90** /0.83 /34.71	N/A,655.50 /2.58 /0.52	638.90 /0.30 /1933.34	943.40 /3.53 /5109.00	990.00 /0 66.53	640.43 /2.860 /3.093
LFR1000 <15> |E|=7578	**668.50** /0.50 /46.67	N/A,673.80 /1.47 /1.55	675.00 /0 /1920.48	914.20 /1.33 /4982.00	992.60 /0.96 /512.50	670.40 /2.10 /7.607
LFR1000 <20> |E|=10086	**746.20** /0.87 /25.46	N/A,755.7 /2.10 /2.12	753.70 /0.46 /2221.46	937.40 /1.50 /2231.00	978.90 /0.62 /828.41	747.70 /2.116 /8.516
LFR2000 <20> |E|=19436	**1481.40** /0.92 /246.79	N/A,1496.80 /2.12 /6.28	1497.80 /1.25 /6912.42	1867.80 /3.82 /9700.00	1994.00 /0.98 /4804.0	1487.7 /3.283 /16.70
LFR2000 <25> |E|=24792	**1598.80** /1.08 /135.18	N/A,1614.50 /2.58 /8.00	1604.90 /1.50 /8297.97	1901.80 /3.25 /10267.0	1987.80 /3.99 /5762.0	1603.5 /3.451 /19.48
BA1024 <4> |E|=2045	**429.10** /0.54 /8.93	429.20 /0.75 /3.62	430.21 /0.40 /1201.48	503.60 /0.80 /4813.00	504.00 /0 /39.57	429.30 /0.862 /3.314
BA2000 <4> |E|=3997	**840.30** /0.64 /38.22	840.90 /0.83 /7.60	844.80 /0.87 /2331.83	1001.00 /0.49 /9556.00	1003.00 /1.98 /302.10	842.43 /1.32 /6.626
BA1024 <8> |E|=4083	**551.60** /0.49 /8.12	553.30 /1.10 /5.18	559.40 /0.92 /1499.40	678.30 /0.80 /4940.00	679.00 /1.36 /140.10	552.57 /1.67 /4.852
ER100 <8> |E|=384	**65.00** /0 /0.37	66.00 /0 /12.21	**65.00** /0 /29.37	83.00 /0 /6.83	93.00 /0 /0.11	**65.00** /0 /0.45
ER2000 <8> |E|=8099	**1317.70** /2.45 /49.15	1332.00 /216 /8.10	1341.00 /1.00 /6081.48	1756.00 /0.18 /9063.12	1467.00 /1.12 /705.93	1318.9 /2.91 /10.33
ER2000 <10> |E|=9992	**1385.50** /2.64 /46.44	1395.33 /1.25 /6.43	1407.10 /1.04 /6477.56	1794.00 /0.60 /9122.00	1435.00 /1.79 /1083.0	1386.0 /2.72 /11.30

[a]The underlined results with N/A, e. g., N/A,655.50/2.58/0.52, indicates that the MBR does not converge when ml=50, but when ml=10 the result is 655.50/2.58/0.52.
[b]The underlined result 666.00/0/86400 is not the average of 10 runs, but only one run because of long runtime.

As introduced in Section II (see Fig. 2), a PS graph has two SNEs: one is the optimal solution which has k+2 vertices in V_{MVC}, the other is a non-optimal solution which has 2k+2 in V_{VC}. For the PS1000 network in Table VI, one has k=332 from N=3k+4. The number of vertices in V_{MVC} is k+2=334 and the number of vertices in the non-minimum set V_{VC} is 2k+2=666. The MBR, HGA, SBTS, and GMA-MVC obtain the optimal solution (SNE) every time in the 10 runs. The VCAR and GA only obtain $|V_{VC}|$=666.

The underlined items for the MBR in Table VI, for example "N/A,655.50/2.58/0.52" indicates that the MBR does not converge when ml=50, but when ml=10 the result is 655.50/2.58/0.52. From Table VI, the GMA-MVC obtains the best solutions on all the 15 networks. The performance of the MBR relates to the structure of the network, which performs better on a scale-free network, the obtained solutions are similar to that of the GMA-MVC with short runtime. However, the MBR performs poorly on WS small-world and LFR modular networks, which usually need very long runtime or do not converge. The SBTS obtains the best solutions on 4 of the 15 networks. In general, the SBTS obtains a solution approximate to that of the GMA-MVC with less runtime.

Statistical tests between the GMA-MVC and MBR. We do the *t* test to compare the performance between the GMA-MVC and MBR. From the 15 pairs of fitness values of the GMA-MVC and MBR in Table VI, their differences are calculated: D={0,-1.40,-2.20,-3.90,-19.60,-5.30,-9.50,-15.40, -15.70,-0.10,-0.60,-1.70,-1.00,-14.30,-9.83}. The elements in D are independent from each other and obey the same distribution. Without loss of the generality, suppose the elements in D obey the normal distribution $D_i \sim N(\mu_D, \sigma_D^2)$ $(i = 1,2,\dots,n)$ where μ_D and σ_D^2 are unknown. In other word, $D_1, D_2, \dots, D_n$ is a sample of the normal distribution $N(\mu_D, \sigma_D^2)$. In the *t* test of our experiments, the hypotheses are:

H_0: The average fitness value of the GMA-MVC is smaller than that of the MBR;

H_1: The average fitness value of the GMA-MVC is not smaller than that of the MBR.

The hypotheses are equivalent to the following hypothetical format:

The 前加上As, As the ...

H_0: $\mu_D < 0$;
H_1: $\mu_D \geq 0$.

In the significance level $\alpha = 0.005$, the critical region of this test problem is calculated by:

$$t = \frac{\bar{d}}{s_D/\sqrt{n}} \geq t_\alpha(n-1), \qquad (6)$$

where $\bar{d}$ is the average of the samples, s_D is the variance of the samples, n is the number of the samples, $t_\alpha(n-1)$ is the α quantile of the $t(n-1)$ distribution with the degree of the freedom $n-1$.

In our experiment, $n = 15$. From D, we obtain $\bar{d} = -6.7020$ and $s_D = 6.5309$. As a result, $t = \frac{\bar{d}}{s_D/\sqrt{n}} = -3.9745$. From the critical value table of the t distribution, we obtain that $t_\alpha(n-1) = t_{0.005}(14) = 2.978$. It's clear that $t < t_{0.005}(14)$, which means that the value of t is not in the critical region. So we accept H_0: the GMA-MVC performs better than the MBR.

Statistical tests between the GMA-MVC and SBTS. Similarly, from the 15 pairs of fitness values of the GMA-MVC and SBTS in Table VI, their differences are calculated: D={0,0,0,-1.87,-4.53,-1.90,-1.50,-6.37,-4.73,-0.20,-2.13,-9.70, 0,-1.20,-0.50}. In the *t* test, the hypotheses are as follows:

H_0: The average fitness value of the GMA-MVC is smaller than that of the SBTS;

H_1: The average fitness value of the GMA-MVC is not smaller than that of the SBTS.

In the significance level $\alpha = 0.005$, the critical region is $t \geq t_\alpha(n-1) = t_{0.005}(14) = 2.786$. From D, we obtain that $t = \frac{\bar{d}}{s_D/\sqrt{n}} = -3.2543$. Again we obtain that $t < t_{0.005}(14)$, hypothesis H_0 is accepted. As a result, GMA-MVC performs better than the SBTS.

We also do statistical tests between the GMA-MVC and HGA. Similar result is obtained: in the significance level $\alpha = 0.005$, the average fitness value of the GMA-MVC is smaller than that of the HGA.

As for the runtime, the GMA-MVC always has less runtime than HGA in Table VI. The reasons are: (i) GMA-MVC uses the asynchronous snowdrift game as the local search which is much effective than that of the HGA; (ii) The two-point crossover of the GMA-MVC is very simple and efficient, whereas the crossover operator of the HGA select vertex one by one to the offspring to handle the constraint, which spends more time. Since the SBTS is really a local search technique, which usually has less runtime than the GMA-MVC.

Comparing the HGA with GA, the HGA always obtains a better solution than the GA due to the local search technique. Because the indices assignment (step 2) in the GA needs to traverse the edges for each individual, the GA always consumes more runtime than the HGA.

E. *Experiments on Benchmark Networks*

For the networks in Table VI, we do not know the optimal solutions, but only comparing the solutions from different algorithms. Recently, a method to generate benchmark networks for the MVC problem was proposed by Xu [59], and some benchmark networks are also available in the website. The number $|V_{MVC}|$ for each benchmark networks is known, so we can see how obtained solutions by algorithms approximate to the optimal solutions.

The parameters of the algorithms are the same as that in the last subsection excepting g_{max}=100 and m_{size}=1000 for the HGA, GA, and GMA-MVC. Table VII shows the smallest number of vertex cover and the average number of vertex cover of 1[illegible]ns. The GMA-MVC always obtains the best result among the algorithms.

TABLE VII
EXPERIMENTS ON BENCHMARK NETWORKS:
BEST FITNESS *F*(*X*) / AVERAGE FITNESS *F*(*X*)

Networks	GMA-MVC	MBR	HGA	GA	SBTS
Frb450 $\lvert V_{MVC}\rvert$=**420** $\lvert E\rvert$=17827	**420 /420**	N/A,423 /423.80	424 /424.80	444.00 /445.20	**420 /420.90**
Frb760 $\lvert V_{MVC}\rvert$=**720** $\lvert E\rvert$=41314	**721 /721**	N/A,726 /727.40	727 /727.10	753.00 /754.60	**721** /722.00
Frb945 $\lvert V_{MVC}\rvert$=**900** $\lvert E\rvert$=59186	**901 /901.60**	N/A,906 /908.30	907 /907.20	×	902 /902.90

[a]Symbol "×" indicates that the obtained solution cannot cover all the edges.
[b]The underlined results with N/A, indicates that the MBR does not converge when *ml*=50, but obtained with *ml*=10.

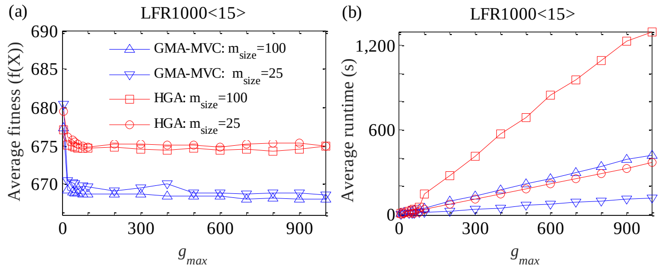

Fig. 7. Effects of m_{size} and g_{max}. (a) Average fitness. (b) Average runtime.

F. Sensitivity Analyses of the Parameters

The GA, HGA, and GMA-MVC have three common parameters: m_{size}, g_{max}, and mr. Among them, mr is not sensitive to all the three algorithms. Parameter r of the proposed GMA-MVC works well in the range: $0<r<1/k_{max}$. The GMA-MVC and HGA are two MAs for the MVC problem; both of them have two key parameters: m_{size} and g_{max}. In this subsection, experiments are done to reveal the effects of the two parameters. For convenience to analyze, we compare the GMA-MVC with the HGA. A series experiments are done on the LFR1000<15> network with different values of g_{max} and m_{size}. Fig. 7(a) and (b) are the fitness and runtime respectively, each point is an average value on 10 runs of the algorithms.

In order to analyze conveniently

over

Sensitivity of g_{max}. From Fig. 7(a), as the increasing of g_{max}, the number of covered vertices is decreased rapidly for both GMA-MVC and HGA when $g_{max}\leqslant 100$. When $g_{max}>100$, the decreasing speed is slow.

Sensitivity of m_{size}. From Fig. 7(a), as m_{size} is increased from 25 to 100, the number of covered vertices is decreased for both GMA-MVC and HGA. With the same values of m_{size} and g_{max}, the runtime of the GMA-MVC is always smaller than that of the HGA (see Fig. 7(b)). The reason is that the crossover operator of the HGA has to deal with the offspring one vertex after another for the constraint, which is more complicated.

Sensitivity of *le*. The GMA-MVC has a particular parameter *le*. Experiments are done on two networks to reveal the effects. Fig. 8 shows the number of covered vertices vary with different values of *le*, g_{max}=100 and m_{size}=100 are fixed. Each point is an average of 10 runs. When *le*=10, the number of covered vertices can be reduced by 1 to 2 comparing with *le*=1.

图7图8都做出了fitness图，图7的标注是fitness，图8的标注是网络名，感觉很奇怪

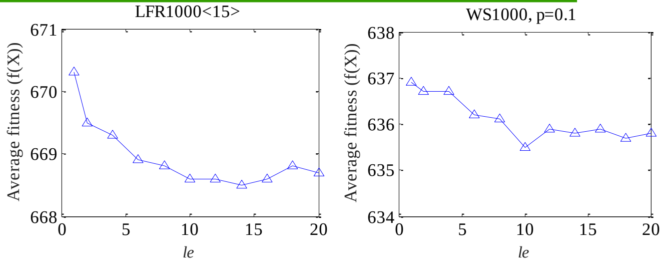

Fig. 8. Effects of *le* with g_{max}=100 and m_{size}=100. (a) LFR1000<15>. (b) WS1000, p=0.1.

Average fitness of LFR1000<15>

VII. CONCLUSION

By encoding the strict Nash equilibrium of the snowdrift game as individuals (chromosomes), the game-based individual evolution is embedded into the population-based global optimization techniques. The proposed GMA-MVC for the minimum vertex cover problem always obtains a better solution than the game-based algorithm (e.g., MBR) and the population-based evolutionary algorithms (e.g., GA and HGA).

The game evolution is easy to be trapped in local optima. Premature convergence is one of the most common difficulties of the population-based evolutionary process. Encoding the strict Nash equilibriums of the game as individuals of the population-based evolutionary algorithms may be a promising method for many optimization problems besides the minimum vertex cover, which should be further studied in the fields of game theory and evolutionary computation.

ACKNOWLEDGMENT

The authors would like to thank Dr. X. Rong Li, Dr. Vesselin P Jilkov, and Dr. Huiming Chen (in Department of Electronic Engineering, University of New Orleans, USA) for valuable discussion.

REFERENCES

[1] I. Dinur and S. Safra, "On the hardness of approximating minimum vertex cover," *Annals of Mathematics*, vol. 162, pp. 439-485, Jul., 2005.
[2] M. Safar, M. Taha, and S. Habib, "Modeling the communication problem in wireless sensor networks as a vertex cover," in 2007 *IEEE/ACM Int. Conf. on Comp. Syst. and Appl.*, pp. 592-598.
[3] W.-C. Ke, B.-H. Liu, and M.-J. Tsai, "Efficient Algorithm for Constructing Minimum Size Wireless Sensor Networks to Fully Cover Critical Square Grids," *IEEE Trans. Wireless Comm.*, vol. 10, no. 4, pp. 1154-1164, Apr. 2011.
[4] L. Wang and T. Jiang, "On the Complexity of Multiple Sequence Alignment," *Journal of Computational Biology*, vol. 1, no. 4, pp. 337-48, 1994.
[5] J. E. Anderson and A. Chakrabortty, "Minimum Cover Algorithm for PMU Placement in Power System Networks Under Line Observability Constraints," in 2012 *IEEE Power and Energy Society General Meeting*, pp. 1-7.
[6] E. Sáenz-de-Cabezón and H. P. Wynn, "Measuring the robustness of a network using minimal vertex covers", Mathematics and Computers in Simulation, vol. 104, pp. 82-94, 2014.
[7] B. Monien and E. Speckenmeyer, "Ramsey numbers and an approximation algorithm for the vertex cover problem," *Acta Informatica*, vol. 22, no. 1, pp. 115-123, 1985.
[8] J. Chen, Y. Lin, J. Li, G. Lin, Z. Ma, and A. Tan, "A rough set method for the minimum vertex cover problem of graphs, " *Applied Soft Computing*, vol. 42, pp. 360-367, 2016.
[9] B. Harsh and A. Geetanjli, "Harnessing Genetic Algorithm for Vertex Cover Problem", *International Journal on Computer Science and Engineering*, vol. 4 no. 02, pp. 218-223, Feb. 2012.
[10] D. S. Hochbaum, "Approximation algorithms for the set covering and vertex cover problems," *SIAM Journal on Computing*, vol. 11, no. 3, pp. 555-556, 1980.
[11] S. Kratsch and F. Neumann, "Fixed-parameter evolutionary algorithms and the vertex cover problem," *Algorithmica*, vol. 65, no. 4, pp. 754-771, 2013.
[12] X. L. Liu, H. L. Lu, W. Wang, and W. L. Wu, "PTAS fot the minimum k-path connected vertex cover problem in unit disk graphs," *Journal of Global Optimization*, vol. 56, no. 2, pp. 49-458, 2013.
[13] P. S. Oliveto, J. He, and X. Yao, "Evolutionary Algorithms and the Vertex Cover Problem ", in *2007 IEEE Congress on CEC*, pp. 25-28.
[14] K. Kotecha and N. Gambhava， "A Hybrid Genetic Algorithm for Minimum Vertex Cover Problem," in *the First Indian International Conference on Artificial Intelligence*, 2003.
[15] Q. Z. Fang and L. Kong, "Core stability of vertex cover games," in 2007 *Proc. of Int. and Netw. Econ.*, pp. 482-490.
[16] A. Li, C. B. Tang, and X. Li, "An Evolutionary Game Optimization to Vertex Cover of Dynamic Networks", in *2014 33rd Chinese Control Conference (CCC)*, pp. 2757-2762.
[17] Y. Yang and X. Li，"Towards a Snowdrift Game Optimization to Vertex Cover of Networks," *IEEE Trans. on Cybe.* vol. 43，no. 3, pp. 948-956 Jun. 2013.
[18] J. Cardinal and M. Hoefer, "Non-cooperative facility location and covering games," *Theoretical Computer Science*, vol. 411, pp. 1855-1876, 2010.
[19] C. Tang, A. Li, and X. Li, "Asymmetric Game: A Silver Bullet to Weighted Vertex Cover of Networks," *IEEE Trans. on Cybe.* Doi: 10.1109/TCYB.2017.2731598, accepted paper, 2017.
[20] P. Moscato, "On evolution, search, optimization, genetic algorithms and martial arts: Toward memetic algorithms," Caltech Concurrent Computation Program, California Instit. Technol., Pasadena, CA, USA, Tech. Rep. 826, 1989.
[21] Y. Ong, M. Lim, N. Zhu, and K. Wong, "Classification of adaptive memetic algorithms: A comparative study," *IEEE Trans. Syst., Man Cybern., B, Cybern.*, vol. 36, no. 1, pp. 141–152, Feb. 2006.

[22] P. Moscato and C. Cotta, "Introduction to Memetic Algorithms," *Handbook of Metaheuristics, International Series in Operations Research and Management Science*, vol. 57, pp. 105-144, 2003.
[23] X. Chen, Y.-S. Ong, M.-H. Lim, and K. C. Tan, "A Multi-Facet Survey on Memetic Computation," *IEEE Trans. Evol. Comput.*, vol. 15, pp. 591-607, Oct. 2011.
[24] G. Zhang and Y. Li, "A Memetic Algorithm for Global Optimization of Multimodal Nonseparable Problems," *IEEE Trans. Cybern.*, vol. 46, pp. 1375-1387, Jun. 2016.
[25] J. Sun, J. M. Garibaldi, N. Krasnogor, and Q. Zhang, "An intelligent multi-restart memetic algorithm for box constrained global optimization," *Evol. Comput.*, vol. 21, no. 1, pp. 107–147, Mar. 2013.
[26] M. Tang, "A Memetic Algorithm for the Location-Based Continuously Operating Reference Stations Placement Problem in Network Real-Time Kinematic," *IEEE Trans. Cybern.*, vol. 45, no. 10, pp. 2214-2223, Oct 2015.
[27] V. A. Shim, K. C. Tan, and H. Tang, "Adaptive Memetic Computing for Evolutionary Multi objective Optimization," *IEEE Trans. on Cybe.*,vol. 45, no. 4, pp: 610-621, 2015.
[28] S.-Y. Wang and L. Wang, "An Estimation of Distribution Algorithm-based Memetic Algorithm for the Distributed Assembly Permutation Flow-Shop Scheduling Problem," *IEEE Trans. Syst., Man, Cybern. A, Syst.*,vol. 46, pp. 139-149, Jan. 2016.
[29] S.-C. Horng, S.-Y. Lin, L. H. Lee, and, C.-H. Chen, "Memetic Algorithm for Real-Time Combinatorial Stochastic Simulation Optimization Problems With Performance Analysis," *IEEE Trans. Cybern.*, vol. 43, no. 5, pp. 1495-1509, Oct. 2013.
[30] A. S. Azad, M. M. Islam, and S. Chakraborty, "A Heuristic Initialized Stochastic Memetic Algorithm for MDPVRP With Interdependent Depot Operations," *IEEE Trans. Cybern.*, doi: 10.1109/TCYB.2016.2607220, accepted paper.
[31] R. Shang, K. Dai, L. Jiao, and R. Stolkin, "Improved Memetic Algorithm Based on Route Distance Grouping for Multiobjective Large Scale Capacitated Arc Routing Problems," *IEEE Trans. Cybern.*, vol. 46, pp. 1000-1013, Apr. 2016.
[32] P. Rakshit, A. Konar, P. Bhowmik, I. Goswami, S. Das, L. C. Jain, and A. K. Nagar, "Realization of an Adaptive Memetic Algorithm Using Differential Evolution and Q-Learning: A Case Study in Multirobot Path Planning," *IEEE Trans. Syst., Man, Cybern. A, Syst.*,vol. 43, pp. 814-831, Jun. 2013.
[33] M. Gong; Z. Peng, L. Ma, J. Huang, "Global Biological Network Alignment by Using Efficient Memetic Algorithm," *IEEE/ACM Tran. Comp. Biol. Bioinform.*, vol. 13, pp. 1117-1129, 2016.
[34] C.-P. Chen, S. C. Mukhopadhyay, C.-L. Chuang, T.-S. Lin, M.-S. Liao, Y.-C. Wang, and J.-A. Jiang, "A Hybrid Memetic Framework for Coverage Optimization in Wireless Sensor Networks," *IEEE Trans. Cybern.*, vol. 45, pp. 2309-2322, Oct. 2015.
[35] G. Lin, W. Zhu, and M. M. Ali, "An Effective Hybrid Memetic Algorithm for the Minimum Weight Dominating Set Problem," *IEEE Trans. Evol. Comput.*, vol. 20, pp. 892-907, Dec. 2016.
[36] C.-C. Liao and C.-K. Ting, "A Novel Integer-Coded Memetic Algorithm for the Set k-Cover Problem in Wireless Sensor Networks," *IEEE Trans. Cybern.*, doi: 10.1109/TCYB.2017.2731598, accepted paper, 2017.
[37] P. Vijayaraju, B. Sripathy, D. Arivudainambi, and S. Balaji , "Hybrid Memetic Algorithm With Two-Dimensional Discrete Haar Wavelet Transform for Optimal Sensor Placement," *IEEE Sensors Journal*, vol. 17, no. 7, pp. 2267-2278, Apr. 2017.
[38] M. L. Nguyen, S. C. Hui, and A. C. M. Fong , "Submodular Memetic Approximation for Multiobjective Parallel Test Paper Generation," *IEEE Trans. Cybern.*, vol. 47, no. 6, pp. 1562-1575, Jun. 2017.
[39] P. Gururu and R. Dantu, "An impatient evolutionary algorithm with probabilistic tabu search for unified solution of some NP-hard problems in graph and set theory via clique finding," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 38, no. 3, pp. 645–666, Jun. 2008.
[40] M. Hifi, "A genetic algorithm-based heuristic for solving the weighted maximum independent set and some equivalent problems," *J. Oper. Res. Soc.*, vol. 48, no. 6, pp. 612–622, Jun. 1997.
[41] R. B. Bai, E. K. Burke, G. Kendall, J. P. Li, and B. McCollum, "A hybrid evolutionary approach to the nurse rostering problem," *IEEE Trans. Evol. Comput.*, vol. 14, no. 4, pp. 580–590, Aug. 2010.
[42] C. A. C. Coello, "Theoretical and numerical constraint-handling techniques used with evolutionary algorithms: A survey of the state of the art," *Comput. Methods, Appl. Mech. Eng.*, vol. 191, pp. 1245–1287, Jan. 2002.
[43] J. Wu, Z. Chang, L. Yuan,Y. Hou, and M. Gong, "A Memetic Algorithm for Resource Allocation Problem Based on Node-Weighted Graphs, " IEEE CIM, pp. 58-69, Apr. 2014.
[44] Y. Jin and J.-K. Hao, "General swap-based multiple neighborhood tabu search for the maximum independent set problem," *Eng. Appl. Artif. Intell.*, vol. 37, pp. 20–33, Jan. 2015.
[45] P. C. Pop and O. Matei, "A memetic algorithm approach for solving the multidimensional multi-way number partitioning problem," *Appl. Math. Model.*, vol. 37, no. 22, pp. 9191–9202, Nov. 2013.
[46] J. Kratica, J. Kojić, and A. Savić, "Two metaheuristic approaches for solving multidimensional two-way number partitioning problem," *Comput. Oper. Res.*, vol. 46, pp. 59–68, Jun. 2014.
[47] C. Solnon and S. Fenet, "A study of ACO capabilities for solving the maximum clique problem," *J. Heuristics*, vol. 12, no. 3, pp. 155–180, May 2006.
[48] G. Szabo and G. Fath, "Evolutionary games on graphs," *Physics Reports*, vol. 446, no. 4, pp. 97-216, 2006.
[49] R. Chiong and M. Kirley, "Effects of Iterated Interactions in Multiplayer Spatial Evolutionary Games," *IEEE Trans. Evol. Comput.*, vol. 16, no. 4, pp. 537-555, Aug. 2012.
[50] C. H. Papadimitriou and K. Steiglitz, "Algorithms and Complexity," in *Combinatorial Optimization,* New York: Dover, ch. 15, sec. 6, 1998, pp. 360-363.
[51] Y. Jin and J.-K. Hao "General swap-based multiple neighborhood tabu search for the maximum independent set problem," Engineering Applications of Artificial Intelligence, vol. 37, pp. 20–33, 2015.
[52] A. S. Azad, M. M. Islam, and S. Chakraborty, "A Heuristic Initialized Stochastic Memetic Algorithm for MDPVRP With Interdependent Depot Operations," *IEEE Trans. on Cybe.*,doi: 10.1109/TCYB.2016.2607220, accepted paper.
[53] S. Khuri and T. Bäck, "An evolutionary heuristic for the minimum vertex cover problem," *Genetic Algorithms within the Framework of Evolutionary Computation*, pp. 86-90, 1994.
[54] A. L. Barabási and R. Albert, "Emergence of scaling in random networks," *Science*, vol. 286, no. 5439, pp. 509-512, Oct. 1999.
[55] P. Erdös and A. Rényi, "On random graphs," *Publ. Math. Debrecen*, vol. 6, no. 290, pp. 290-297, 1959.
[56] D. J. Watts and S. H. Strogatz, "Collective dynamics of 'small-world' networks," *Nature*, vol. 393, no. 6684, pp. 440-442, Jun. 1998.
[57] A. Lancichinetti, S. Fortunato, and F. Radicchi, "Benchmark graphs for testing community detection algorithms," *Phys. Rev. E*, vol. 78, no. 4, p. 046110, 2008.
[58] D. Lusseau, K. Schneider, O.J. Boisseau, P. Haase, E. Slooten, S.M. Dawson, "The Bottlenose Dolphin Community of Doubtful Sound Features a Large Proportion of Long-Lasting Associations: Can Geographic Isolation Explain This Unique Trait?" *Behav. Ecol. Sociobiol.*, vol. 54, pp. 396-405, 2003.
[59] K. Xu, BHOSLIB: Benchmarks with Hidden Optimum Solutions for Graph Problems Maximum Clique (Maximum Independent Set, Minimum Vertex Cover and Vertex Coloring), http://www.nlsde.buaa. edu.cn/~kexu/benchmarks/graph-benchmarks. Htm.

Jianshe Wu (M'09) received his Ph.D. degrees in pattern recognition and intelligence system in 2007 from the Xidian University, Xi'an, China. He currently works as a professor at the Key Laboratory of Intelligent Perception and Image Understanding of Ministry of Education of China, School of Electronic Engineering (SEE), Xidian University, China. His main research interests lie in the areas of complex networks, computational intelligence, and pattern recognition.

Kui Jiao received his B.S. and M.S. degree from Xidian University, Xi'an, China, in 2013 and 2016 respectively. Currently, he is a software engineer in Baidu Times (Beijing) Co., Ltd.

Xing Shen received her B.S. degree from the School of Electronic Engineering, Xidian University, Xi'an, China, in 2015. Currently, she is working towards her master degree at the School of Electronic Engineering.