Iterated game based local search to the maximum independent set problem

Jianshe Wu. *Member, IEEE*, and Xing Shen

改成the maximum independent set problem

Abstract—The maximum independent set is a well-known NP -hard problem in combinatorial optimization. A one-to-one corresponding relation between the maximum independent sets and the Nash equilibriums of the prisoner's dilemma game is established firstly in this paper. The cooperators of the Nash equilibrium constitute the local maximum independent set. Based on the relation, a distributed algorithm for maximum independent sets is proposed by game, which is much better than previous algorithms with a control center. Then an iterated game based local search is provided.

We treat each vertex as a rational agent playing the prisoner's dilemma game asynchronously with its neighbors. Artificially reversing an agent's strategy in a Nash equilibrium from cooperator to defector will lead to the strategy changes of its neighbors, and converge to a new Nash equilibrium again. This process may find a better solution and we prove that at least it will not degenerate the solution, which is the proposed game-based local search (GLS). In order to escape from the local optimal, we further propose iterated game-based local search (IGLS) by employing the perturbation skill. Extensive experiment results on various networks show that the proposed IGLS outperforms recent state of the art algorithms in performance.

the可以改成this

后面加上algorithm

策略表达用cooperation、defection

Index Terms—maximum independent set, prisoner's dilemma game, local search, Nash equilibrium, graph theory.

I. Introduction

The maximum independent set (MIS) problem is a well known combinatorial optimization problem in the graph theory [1],[2]. Given an undirected network described by $G=(V, E)$ where V is the set of vertices and E is the set of edges of the network. The MIS problem is to find a set of vertices $V_{MIS} \subseteq V$ with the maximum cardinality, in which all vertices are independent with each other in the sense that there is not an edge among them.

The MIS problem has many important applications in a wide range of fields such as classification theory [3], information retrieval, computer vision [4], computer graphics [5], map labeling [6], and routing [7]. The MIS problem is related with two other combinatorial optimization problems named the maximum clique (MC) problem and the minimum vertex cover (MVC) problem [8,9]. To find the MC of a graph is equivalent to find the MIS of the complement graph. In the same way, if V_{MIS} is the maximum independent set then $V \backslash V_{MIS}$ is the minimum vertex cover of the network. The V_{MIS} together with the V_{MVC} makes up the vertex set V of the network [10]. So a progress in solving one of the three problems also means a progress to the other two problems.

The three are all NP-hard problems [11], and there are many heuristics algorithms for the problems [1]-[2], [12]-[18]. For convenience in representation, an independent set (IS) is called a local maximal IS (LMIS) if there is no vertex can be directly inserted into the set with keeping its independence. Among all the LMISs, the one which has the most vertices is the global maximal IS (GMIS). Since the MIS problem is NP-hard, an algorithm always attempts to find a LMIS with more vertices as possible.

改成obtaining

前面需要加上一个as

Most of these heuristics algorithms solve the problems by obtain a LMIS with random insertion firstly, and then using various local search (LS) techniques to improve the solution. Those LS techniques usually based on (k,l)-swaps by removing k and adding l vertices $(k,l \geqslant 0,\ l>k)$ [2],[10]. For example, the (1,2)-swap and (2,3)-swap which are the famous 2-improvement and 3-improement LS respectively [2].

不需要用复数

Besides the LS techniques, there are some exact algorithms for the MIS problem as well. These exact algorithms mainly apply reduction operation during recursion [19].

Evolutionary games on graphs have been studied for many years [20],[21], where each vertex plays game with his neighbors. Evolutionary snowdrift game-based algorithms for the MVC problem are also developed [22]-[24]. Comparing with the swap-based algorithms, a merit of the game-based algorithms is that they always run in a distributive manner and can be easily realized in a parallel way. In this paper, we attempt at developing a game-based algorithm for the MIS problem. Instead of the snowdrift game, we find that the prisoner's dilemma game (PDG) is suitable for the MIS problem. So we treat each vertex as a rational agent and each vertex play the PDG with its neighbors on the graph. It is surprised that there exists a perfect equivalence between the LMISs of the graph and the Nash equilibriums (NEs). A NE of the PDG corresponds to a LMIS, which indicates than we can obtain a LMIS by the PDG. Experiments show that PDG-based method always obtains a better LMIS with less runtime than the random insertion (see Section VII.*B*).

his its 应统一使用一种称谓？

that

改成when

A NE state of the graph (corresponding to a LMIS) is usually a local optimal state. If the strategy of an agent in a NE is artificially changed, the strategies of its neighbors may be sequentially changed by the best response rule,⋯, and a new NE will arrive again. Based on this method, we can obtain a new LMIS with increased cardinality. Thus, based on the PDG, we come up with an efficient LS technique, named as game-based local search (GLS). In the swap-based LS for the MIS problem, it is obvious that a (2,3)-swap cannot be achieved

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)..

by a (1,2)-swap, and vice versus. The merit of the GLS is that it can realize a (k,l)-swap for various number of k and l (k,l⩾0). In other words, the GLS provides an uniform implementation scheme of the (k,l)-swap for various number of k and l.

When LS is trapped in LMISs, perturbation is a commonly used method. A game-based perturbation method is also designed for the MIS problem in this paper. Iteratively using the GLS and the perturbation, the iterative game-based local search (IGLS) algorithm is provided. The IGLS can find the optimal solution in a reasonable time.

The rest of this paper is structured as follows. Section Ⅱ is the preliminaries, detailed description of the MIS problem, the PDG, and the NE are given. Section Ⅲ introduces related algorithms and the motivations of our IGLS. Section IV introduces how to obtain a LMIS by the PDG. The GLS is described in Section V. The detailed process of the IGLS is provided in Section VI. Section VII is the experiments. We conclude this paper in Section VIII.

II. Preliminaries

A. The MIS Problem

Given an undirected graph denoted by G=(V, E), where $V = \{1, 2, \dots, N\}$ is the set of vertices and $E = \{e_{ij} | \ i, j \in V\}$ is the set of edges. $N = |V|$ is the number of vertices and $m = |E|$ is the number of edges of the given graph, where $|\cdot|$ denotes the cardinality of the element. Let $V_{IS} \subseteq V$ denote an IS in which each vertex has no edges with other vertices. There are many forms of IS of a graph. Particularly, a single vertex can be seen as an IS with the cardinality one. A GMIS denoted by V_{GMIS} is an IS of a graph with the global maximum cardinality. So the goal of the MIS problem is to find an IS with the global maximum cardinality among all the ISs.

An IS is called a LMIS (denoted by V_{LMIS}) if no a vertex can be inserted without removing a vertex from the set. A non-LMIS is denoted as $V_{\overline{LMIS}}$ with at least a new vertex can be inserted directly. A GMIS is always a LMIS, which is a special LMIS with the global maximum cardinality. On the contrary, the cardinality of a V_{LMIS} is not the global maximum usually. A V_{LMIS} is easily obtained in practice. However, to find a V_{GMIS} is NP-hard. Fig.1 shows a LMIS and a GMIS of the graph. Fig.1 (a) is a LMIS with cardinality 2 and Fig.1 (b) is a GMIS with cardinality 3.

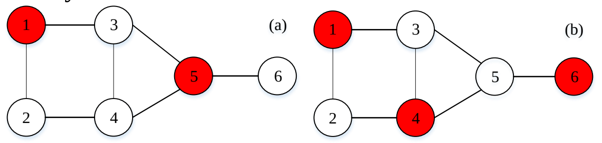


Fig.1 A LMIS and a GMIS. (a) $V_{LMIS} = \{1,5\}$; (b) $V_{GMIS} = \{1,4,6\}$.

B. The Prisoner's Dilemma Game

The prisoner's dilemma is a classical game scheme in game theory [25]. Imagine that two prisoners are arrested by the police. However, the police have no evidence to charge the two criminals. So the police interrogate them respectively. The two prisoners have two choices: cooperation (C) and defection (D). C means that saying nothing to the police while D stands for betraying the other one. If one of them chooses C and the other chooses D. The cooperator will be put into prison for ten years with enough evidence, while the defector will be acquitted of a charge because of telling the truth to the police. If both of them choose D, there will be enough evidence to charge all of them. Both of them will be put into prison for five years. If they all choose C and the police don't have enough evidence. They will be punished by one year. In general, the payoffs of two prisoners are shown in Table I.

TABLE I
The Payoffs of The PDG

	C	D
C	(R, R)	(S, T)
D	(T, S)	(P, P)

In Table I, T, R, P and S satisfy T>R>P≥S. From the global perspective, when they both choose C, they will obtain the maximum payoff totally: 2R>T+S or 2R>2P. Usually, the values of T, R, P and S are set as follows: $T = b$, R = 1, P = 0 and S = 0. The parameter b is called the temptation to defection and satisfies $1 < b < 2$ [26]. Hence, obtaining the one-parameter payoff matrix of the PDG as follows:

$$\begin{array}{ccc} & C & D \\ C & 1 & 0 \\ D & b & 0 \end{array}$$

The payoff matrix can be simply recorded as [27]:

$$P_{PDG} = \begin{pmatrix} 1 & 0 \\ b & 0 \end{pmatrix}, \quad 1 < b < 2. \tag{1}$$

Given a graph $G = (V, E)$, the vertices can be treated as players of the PDG and the edges stand for the interactions among those vertices. Each vertex plays PDG with all its neighbors and gains a total cumulated payoff. Suppose all players are completely rational.

C. Nash Equilibrium

Consider an N-player PDG with a finite space $S = \prod S_i$, where S_i is the strategy set of vertex i. In the PDG, $S_i = \{C, D\}$. Let $s_i \in S_i$ denotes the strategy that player i chooses, thus $s_i = C$ or D. The strategies of all the N players are $X = \{s_1, \ s_2, \ \dots \ , \ s_N\}$ and the vector $s_{-i} = \{s_1, \ s_2, \ \dots \ , s_{i-1}, \ s_{i+1}, \ \dots, \ s_N\}$ describes the strategies of the $N-1$ players except i. Let $U_i(X) = U_i(s_i, \ s_{-i})$ be the utility function of player i. It stands for the payoff of player i with strategy s_i and the rest players' strategies are s_{-i}.

An NE $X^* = \{s_1^*, \ s_2^*, \ \dots \ , \ s_n^*\}$ is such a state in which every player cannot make its payoff increased by changing its strategy [23]. In formula:

$$\forall i, \ U_i(s_i^*, \ s_{-i}^*) \geq U_i\left(s_i^{'}, \ s_{-i}^*\right) \tag{2}$$

where $s_i^* \in X^*$ and $s_i^{'} \notin X^*$. If (2) holds strictly for every $s_i^{'} \neq s_i^*$: $U_i(s_i^*, \ s_{-i}^*) > U_i\left(s_i^{'}, \ s_{-i}^*\right)$, X^* is a strict Nash equilibrium (SNE).

Best response rule. The best response rule is as follows [25]-[26]. If $U_i(D, \ s_{-i}) > U_i(C, \ s_{-i})$, then strategy D is the best response for the player in the next step. Otherwise, if $U_i(C, \ s_{-i}) \geq U_i(D, \ s_{-i})$, then strategy C is the best. Please note that if $U_i(C, \ s_{-i}) = U_i(D, \ s_{-i})$, then C is the best response in this paper.

D. Asynchronous Game

In an N-player game, if the players update their strategies synchronously in one time step, that is a synchronous game [28]. If the players update their strategies one by one, after a player updated its strategy the next player begin to calculate its payoff and make a decision, the game is called asynchronous [28],[29]. In a synchronous game, there is no sequence among the players. In an asynchronous game, the strategy choice of earlier players will have an influence on the strategy choice of latter players. In our algorithm, vertices in the network play the PDG asynchronously.

III. Related LS Algorithms

A. Related Algorithms

A special form of the (k,l)-swap is the $(k,k+1)$-swap, which attempts to find k vertices whose removal from the current solution set will allow at least k+1 vertices to be added back to the set. The $(k,k+1)$-swap is also called k-exchange, which first came up in Ref. [4]. In Ref.[2], fast LSs for the (1,2)-swap and (2,3)-swap are proposed, which are named as 2-improvement and 3-improvement respectively. Based on the LSs, the iterated local search (ILS) algorithm to solve the MIS problem is provided and the results are well promising in practice [2].

There are some useful definitions in Ref.[2]. Suppose a V_{LMIS} of the graph is obtained. The tightness of a vertex $i \notin V_{LMIS}$, denoted by $t(i)$, is the number of neighbors of vertex i that are included in V_{LMIS}. If the tightness of vertex i is k, the vertex is *k-tight*. Particularly, vertices with tightness 0 are called *free* vertices. A free vertex can be directly inserted into the IS without removing any vertex.

The 2-improvement algorithm process each vertex $i \in V_{LMIS}$ in turn and updates the solution V_{LMIS}. The detailed process is as follows [2]: First, remove vertex i from the current solution V_{LMIS} temporarily, obtaining a new solution V_{IS}. If the number of free vertices in solution V_{IS} is less than two, stop the LS process. There is no 2-improvement after removing vertex i. Otherwise, for each neighbor v of vertex i that are free in V_{IS}, insert v into V_{IS} and obtain a new solution S'. Then check if there exists a free vertex w in the solution S'. If there exists, insert vertex w to accomplish a 2-improvement; if it doesn't, remove vertex v and check the next neighbor of vertex i. Finally, if vertex i cannot lead to a 2-improvement, reinsert vertex i into the solution and look for the next vertex in the V_{LMIS}. Andrade et al. make the process of finding a valid 2-improvement faster by obtaining and simplifying a set of candidates. In addition, a 3-improvement LS technique attempts to remove 2 vertices from the current solution and insert 3 vertices, the cardinality of the solution plus by one.

这样表达是否会出现：有tightness参数的节点在集合V_{IS}中的歧义？

加参考文献

改成is increased by

In Ref.[10], the swap-based tabu search (SBTS) is proposed by employing the $(k,1)$-swap, where k can be 0, 1, 2 or a number more than 2. The main operation is to insert one vertex that are not in the V_{IS} into V_{IS}, and remove vertices in V_{IS} that are adjacent to the vertex. If k=0, the solution is improved; if k=1, the solution is not improved but the structure of the solution is changed; if k>1, the solution is degenerated and the $(k,1)$-swap is only a diversification for escaping from the local optima. 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.

Another recent algorithm is the breakout local search (BLS) for the MC problem [31]. Since finding the MC of a graph is equal to find the MIS of the complement graph, the BLS algorithm can be used to the MIS problem. The BLS is essentially an iterated LS algorithm which uses the tabu list for its directed diversification [32]. The BLS uses LS to find local optima and continually move from one local optimum to another in the search space. The continual moving in search areas is achieved by alternating between random or directed, and weak or strong perturbations [31].

In summary of previous algorithms, three key skills are: (i) a candidate set to the LS technique which limits the search space and speed up the process; (ii) a scheme preventing removed vertex come back to the IS set again soon; (iii) a perturbation scheme that provide various (week and strong) perturbation and help the LS escaping from local optima.

prevent... from...用法，改为from coming back to

改为to escape

B. Motivations

Motivation 1. The relationship between the vertex cover and game has been researched [22]-[24]. It is shown in Ref.[23]: the cooperators in a strict NE of the spatial snowdrift game constitute a vertex cover set. The first motivation of this paper is to reveal the relationship between the LMISs and the NEs of a game. It is proved in this paper that the cooperators in an NE of the PDG constitute a LMIS (see Section IV for details).

As mentioned above, obtaining an initial LMIS is the first step for the MIS problem and it is usually obtained by the random insertion. Based on this finding, an easy method to obtain a LMIS is established by the PDG (see Section IV), experiments show that it always obtain a better LMIS than the random insertion method with less runtime.

Motivation 2. A (1,2)-swap LS process cannot be implemented by a (2,3)-swap usually, and vise versus; A (2,3)-swap LS process cannot be implemented by a (3,4)-swap usually, and vise versus; ⋯, etc. That is the inherent limitation of the (k,l)-swap-based LS technique. In an iterated LS scheme, we may chose one or two best LS techniques, but we can not implement the $(k$-$l)$-swap for all number of k and l $(k<l)$. So the second motivation of this paper is to provide a uniform LS scheme that can implement (k,l)-swap for all number of k and l $(k<l)$.

去掉-

这里用we是否合适？

In this paper, the GLS is provided which implement the uniform LS by the PDG (see Section V for details). Experiments show that the GLS always obtains better solutions than the 2-improvement LS technique.

implements

改为playing the role of

Motivation 3. With the iterated LS scheme [2],[31], the third motivation is to provide an iterated LS algorithm with the GLS as its LS technique. For this purpose, PDG-based perturbation should be designed. As analyzed in Section III.*A*, the diversity of the perturbation is a key. In previous iterated LS algorithms, the perturbation is added after the LS stage, and is iterated until a condition to stop. Besides this kind of perturbation, it is surprised that we can naturally combine perturbation with the GLS without adding any cost, thus provide a diverse perturbation scheme for the IGLS.

IV. The Relation between an NE and a LMIS

The first step for the MIS problem is finding an initial LMIS by the random insertion [2],[10],[33], then use a LS technique for improvement. The random insertion algorithm finds a LMIS by inserting vertices into the IS one by one. Initially, the solution set is empty. In each step, a vertex is randomly chosen

目前还不能得到这一条结论。C节点只有D邻居，这样得到的应该是一个IS。

and adds into the IS if it can. The algorithm stops when there is no vertex that can be inserted into the IS.

In this section, we propose a game-based algorithm for finding a LMIS. Experiments in Section VII show that the game-based algorithm always obtains a better LMIS with less runtime than the random insertion.

Another method for obtaining a LMIS is the greedy algorithm [2]. The greedy algorithm chooses a feasible vertex greedily to insert into the IS, which is conducted as follows. Suppose vertex v can be added into V_{IS}. Let $n(v)$ stands for the number of vertices that can be further inserted into the IS after vertex v is inserted. The greedy algorithm will search all the feasible vertices and compute $n(v)$ of every vertex and then choose the vertex with the maximum $n(v)$. This algorithm tries to preserve as many feasible vertices as possible after each insertion. The algorithm stops when there is no vertex that can be inserted. The ILS takes greedy algorithm to generate the initial LMIS.

In this section, we introduce a new method to obtain a LMIS by the PDG. Each vertex plays the PDG with its neighbors and changes its strategy according to its total utility, and eventually the network will converge to an NE. In such a scenario, we show that the vertices of C of an NE form a V_{LMIS}. Fig.1 shows two NEs of the network, where the red vertices are C and the white vertices are D. The red vertices in Fig.1(a) and Fig.1(b) are two LMISs respectively.

Define

$$V_{NE} = \{i \mid s_i^* = C,\ \ s_i^* \in X^*\}, \tag{3}$$

where $X^* = \{s_1^*,\ s_2^*,\ \dots\ ,\ s_n^*\}$ is the NE of the PDG with $s_i^* = C\ or\ D$.

Theorem 1: Let $\{V_{NE}\}$ denote the set of V_{NE} of a graph and $\{V_{LMIS}\}$ for the set of V_{LMIS}, we have $\{V_{NE}\} = \{V_{LMIS}\}$.

Proof: The proof has two parts:

(i) $\{V_{LMIS}\} \subseteq \{V_{NE}\}$: vertices of a V_{LMIS} are cooperators of an NE.

By corresponding vertices in a V_{LMIS} to cooperators and vertices in $V\backslash V_{LMIS}$ to defectors, $\forall i \in V_{LMIS}$, there is $s_i = C$ and vertex i has only defective neighbors. Let k_i is the degree of vertex i. From the payoff matrix, $U_i(C, s_{-i}) = k_i \times 0 = 0$ and $U_i(D, s_{-i}) = k_i \times 0 = 0$. From the best response rule, vertex i will choose C in the next step. Thus, vertex i will not change its strategy.

$\forall i \notin V_{LMIS}$, there is $s_i = D$, then $\exists j \in \mathcal{N}_i,\ s_j = C$, where $\mathcal{N}_i$ is the neighbor set of i. Suppose there are c cooperators and d defectors in $\mathcal{N}_i$, $c \geq 1,\ c + d = k_i$. From the payoff matrix, $U_i(C, s_{-i}) = c \times 1 + d \times 0 = c$ and $U_i(D, s_{-i}) = c \times b + d \times 0 = c \times b$. Because $1 < b < 2$, $U_i(D, s_{-i}) > U_i(C, s_{-i})$ holds strictly. So vertex i will not change its strategy in the next step. The game state is an NE, and the vertices of the V_{LMIS} are cooperators of the NE.

(ii) $\{V_{NE}\} \subseteq \{V_{LMIS}\}$: Cooperators of an NE constitute a V_{LMIS}.

$\forall i \in V_{NE}$, form (3), $s_i = C$. Suppose there are c_1 cooperators and d_1 defectors in $\mathcal{N}_i$, where $c_1 + d_1 = k_i$. From the payoff matrix, we have

$$U_i(C, s_{-i}) = 1 \times c_1 + 0 \times d_1 = c_1, \tag{4}$$

$$U_i(D, s_{-i}) = b \times c_1 + 0 \times d_1 = b \times c_1. \tag{5}$$

Since $i \in V_{NE}$ and $s_i = C$,

$$U_i(C, s_{-i}) \geq U_i(D, s_{-i}). \tag{6}$$

From Eqs.(4)-(6), we have $c_1 \geq c_1 \times b$ with $1 < b < 2$, then $c_1 = 0$. That means vertex i has only defective neighbors if $i \in V_{NE}$, thus V_{NE} is a V_{LMIS}.

$\forall i \in V\backslash V_{NE}$, $s_i = D$ in the NE state, suppose vertex i has c_2 cooperative and d_2 defective neighbors, where $c_2 + d_2 = k_i$. From the payoff matrix,

$$U_i(C, s_{-i}) = 1 \times c_2 + 0 \times d_2 = c_2, \tag{7}$$

$$U_i(D, s_{-i}) = b \times c_2 + 0 \times d_2 = b \times c_2. \tag{8}$$

Since $s_i = D$ and it is an NE state, from the best response rule we have

$$U_i(D, s_{-i}) > U_i(C, s_{-i}). \tag{9}$$

D节点至少有一个C邻居，也就是说这个D节点不能转化为C，也就是一个局部最

From Eqs.(7)-(9), we have $c_2 \times b > c_2$ with $1 < b < 2$, then $c_2 \geq 1$. That means at least one neighbor of vertex i is C in the NE state, thus V_{NE} is a V_{IS}: $\{V_{NE}\} \subseteq \{V_{IS}\}$.

Assuming that there is a vertex j that can be directly inserted into to V_{NE} with keeping its independence, which is equivalent to say that vertex $j \in V\backslash V_{NE}$ can change its strategy from D to C directly. That is a contradiction to the definition of an NE, so V_{NE} is a V_{LMIS}: $\{V_{NE}\} \subseteq \{V_{LMIS}\}$.

In summary of (i) and (ii), we have $\{V_{NE}\} = \{V_{LMIS}\}$.

From the proof of theorem 1, **Corollary 1** can be obtained.

Corollary 1. The sufficient and necessary condition for an NE of asynchronous PDG are: (a) a cooperator has no cooperative neighbor and (b) a defector has at least one cooperative neighbor.

Proof: (i) Sufficient part. For any a vertex i, if it is a cooperator and has no cooperative neighbor, from the payoff matrix we have $U_i(C, s_{-i})$=0 and $U_i(D, s_{-i}) = 0$. Vertex i will not change its strategy. If vertex i is a defector and has c_1 cooperative neighbors ($c_1 \geqslant 1$), from the payoff matrix we have $U_i(C, s_{-i})$= c_1 and $U_i(D, s_{-i}) = b \times c_1$. Thus $U_i(D, s_{-i}) > U_i(C, s_{-i})$ and vertex i will not change its strategy by the best response rule. Both the cooperators and defectors will not change their states, the state is an NE.

(i) Necessary part. Since $\{V_{NE}\} \subseteq \{V_{LMIS}\}$, V_{NE} is really a V_{LMIS}, this completes the proof.

Remark 1. From **Theorem 1**, a LMIS corresponds to an NE and vice versus. The relation is one-to-one correspondence. In the following we directly say: $V_{NE} = V_{LMIS}$.

Theorem 2. From any an initial state, the asynchronous PDG will converge to an NE with probability one.

Proof: Suppose the initial state $X(0) = \{s_1(0),\ s_2(0),\ \dots\ ,\ s_N(0)\}$ is not an NE. From **Corollary 1,** there exist one/both of the following two cases:

(a) A *C-C* edge: a cooperator i has cooperative neighbor.

(b) A defector j has no cooperative neighbors (a defector j surrounded by defectors).

If (a) exits, assuming vertex i has c cooperative neighbors and d defective neighbors, from the payoff matrix: $U_i(C, s_{-i}) = c \times 1 + d \times 0 = c$ and $U_i(D, s_{-i}) = c \times b + d \times 0 = c \times b$. $U_i(D, s_{-i}) > U_i(C, s_{-i})$, vertex i will change its strategy from C to D in $X(1)$. There is a positive probability that the *C-C* edge disappear in $X(1)$.

If (b) exits, assuming vertex j has only d defective neighbors, from the payoff matrix: $U_j(C, s_{-i}) = 0$ and $U_i(D, s_{-i}) = 0$. Vertex j will change its strategy from D to C in $X(1)$. There is a positive probability that case (b) disappear in $X(1)$.

In summary, if $X(0)$ is not an NE, there is a positive probability that $X(1)$ is an NE. If $X(1)$ is not an NE, there is a positive probability that $X(2)$ is an NE. In general, if $X(k)$ is not an NE, there is a positive probability that $X(k+1)$ is an NE. Thus the network will converge with probability one to an NE.

From **Theorem 1** and **Theorem 2**, a new method for a LMIS is obtained. All vertices as agents play the asynchronous PDG, then an NE is arrived. The cooperated vertices are elements in the V_{LMIS}.

If an NE is arrived after one round of game**,** a LMIS is obtained when each vertex has a chance to update its strategy by the PDG. The time complexity is $O(m)$. In general, if an NE is arrived after k round of game. The time complexity for a LMIS is $O(km)$. In the random insertion algorithm and the greedy algorithm, determining whether a solution is local maximal can take as much as $O(m)$ time. Experiments show that the proposed PDG-based method is faster than both the random and greedy algorithms.

在and前加上逗号，并加上 *conrespondingly, vertex i changes its strategy from C to D ,*

少了G

V. DESCRIPTION OF THE GLS

A. Theoretical Preparing

Theorem 3: Let $\bar{\mathcal{N}}_\mathrm{i}$=$\mathcal{N}_\mathrm{i} \cup i$. For a given NE of an undirected graph $= (V, E)$, suppose vertex $i \in V_{NE}$ is removed from V_{NE} and obtain V' of the graph, where $V' = V_{NE} \backslash \{i\}$ is not an NE. Let only the vertices in $\bar{\mathcal{N}}_\mathrm{i}$ update their strategies asynchronously, the game will arrive a new NE again.

改成 *asynchronously until NE. Then the graph will arrive at a new NE state again.*

Proof: Let V_{NE}^n denote the set of cooperators of the new NE.

(a) $\forall v \in \mathcal{N}_i \cup i$, if vertex v has neighbors that choose strategy C, then vertex v will choose strategy D by *Corollary 1*. So $\forall u \in V'$, $s_u = C$, $\nexists j \in V_{NE}^n$ and $j \in \mathcal{N}_i \cup i$, $e_{uj} \in E$. So the strategies of vertices in $\mathcal{N}_i \cup i$ cannot influence the strategy choice of vertex v in V'. v改成u

(b) $\forall v \in V \backslash V_{NE}$, $s_v = D$, then there must be at least one vertex that choose strategy C among the neighbors of vertex v. So the strategies of vertices in $\mathcal{N}_i \cup i$ cannot influence the strategy choice of vertex v in $V \backslash V_{NE}$.

(c) There is at least one vertex choosing C in $\mathcal{N}_i \cup i$. Vertices in $\bar{\mathcal{N}}_i$ that has no neighbor with strategy C will choose strategy C in the NE by Corollary 1. Particularly, if no vertex in $\mathcal{N}_i$ choose C, then vertex i will choose, the network will go back to the original NE again. This completes the proof.

Remark 2. In theorem 3, if vertex i updates its strategy firstly, then the new NE is the same as the original and the operation is nonsense. In the following, we say the verities in $\bar{\mathcal{N}}_\mathrm{i}$ update their strategies asynchronously, which connotatively indicates that vertex i updates its strategy in the last.

Theorem 4: Given an NE of an N-player PDG on an undirected graph, corresponds to a solution V_{NE}. Randomly remove vertex $i \in V_{NE}$ from V_{NE}, the strategy of vertex i turn to D from C. Let vertices in $\bar{\mathcal{N}}_\mathrm{i}$update their strategies asynchronously and obtain a new solution V_{NE}^n, then $|V_{NE}^n| \geq |V_{NE}|$.

写错了

Proof: There are four possibilities for V_{NE}^n: 改成situations较好

(i) $\exists\, w \in \mathcal{N}_\mathrm{i}$, w is added into V'_{NE} satisfying $|V_{NE}^n| = |V_{NE}|$ with different vertices. In this case, the cardinality of the solution is not changed, but the solution structure is changed. So the operation increases the diversity, which facilities for a better solution.

(ii) $\exists\, w, v \in \mathcal{N}_\mathrm{i}$, both w and v are added into V_{NE}^n satisfying $|V_{NE}^n| = |V_{NE}| + 1$. This is a (1,2)-swap or 2-improvement in the language of Ref.[2].

改为*and all cooperative vertices in the graph form a new solution*

这里加上：*vertices in Ni that choose D cannot influence the strategy choice of vertex v . In the meantime, vertices in Ni that choose C can only make the number of cooperative neighbors of vertex v increased.Particually, this situation cannot make vertex v change its strategy by the best response rule.*

(iii) $\exists\, \Omega,\ \Omega \subseteq \mathcal{N}_\mathrm{i}$ and $|\Omega| \geq 3$, all vertices in Ω are added into V_{NE}^n satisfying $|V_{NE}^n| \geq |V_{NE}| + 2$. This is a (1, $|\Omega|$)-swap or $|\Omega|$-improvement, including the 3-impromemt.

(iv) No vertex in $\mathcal{N}_\mathrm{i}$ is added into V_{NE}^n, at last vertex i is added into the set again and go back to solution V_{NE}.

This completes the proof.

The second and third situations are clearly wonderful results. The first situation can be considered as a perturbation, which leads to a solution structural change and is helpful to escape from local optima. So it is reasonable to accept the first situation to some degree. In summary, the proposed GLS will improve the solution or change the structure of the solution.

改成*vertex's strategy*

B. Game-Based LS

From an NE ($V_{NE} = V_{LMIS}$), randomly remove a vertex, which is equivalent to artificially change a vertex from C to D. Then the verities in $\bar{\mathcal{N}}_i$ update their strategies asynchronously. In algorithmic operation, we keep a set of candidates that play the game. In other words, only vertices in the candidate set are selected to change the strategy from C to D. Initially, all vertices in V_{LMIS} make up the candidates set. We choose a vertex i from the candidates set randomly and change it from C to D, if the resulted NE brings no change to the solution, remove i from the candidate set. The set of candidates is updated whenever the solution changes.

最后加上一句话：*The scheme of updating the candidate set is as follows.*

Sometimes it may appear a phenomenon that the first situation in **Theorem 4** occurs many times, a vertex is added to the solution again after its removing, and the solution is not changed. To avoid this phenomenon, we use PDG with memory for the LS. In other words, each vertex (agent) has a memory of length ml, which stores the strategies of the vertex in the past ml steps. If a vertex has strategy D in memory, the vertex should be removed from the set of candidates.

A recommended choice for the memory length is to set it a value various with the scale of the initial V_{LMIS}. In our GLS, the memory length ml=$r \times |V_{LMIS}|$, where r is the ratio of the memory length to the cardinality of the initial LMIS generated by the PDG. With a constant r, the memory length varies with the cardinality of the LMIS and thus varies with the network scale. In the experiments of this paper, r=0.9. Since there is no need to calculate the tightness of the neighbors in each step, the candidate set of the GLS is easier to be operated than the 2-improvemtn [2].

We introduce two slightly different GLS routines: random GLS (RGLS) and greedy GLS (GGLS). The detailed steps of the RGLS routine are as follows:

Step 1. Obtain a V_{LMIS} by the PDG and initialize the set of candidates: $V_C = V_{LMIS}$.

an

Step 2. Randomly choose a vertex (e.g., i) from V_C and obtain $\bar{\mathcal{N}}_i$. The strategy of vertex i changes from C to D.

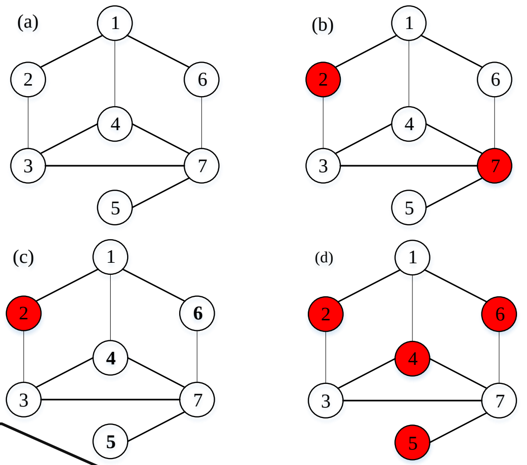
Step 3. Let the vertices in $\bar{\mathcal{N}}_i$ play the PDG asynchronously, until a NE arises again and a new solution V_{LMIS}^n obtains.

Step 4. Update the candidate set V_C. Repeat from step 2 again.

改为*step 2 and step 3*

The above steps stopped in two cases: (i) The candidate set become empty, the GLS stops naturally; (ii) If the above steps repeat for t_{GLS} times. A larger number of t_{GLS} will make most of the GLS stops naturally.

Since the solution structure is invariable before finding a 2-improvement in the ILS[2]. The candidate set will become empty if the LMIS has no 2-improvement. The solution structure is usually changed even if there is no improvement in our GLS, which increases its search ability. On the other hand, the candidate set is not guaranteed to become empty to stop the GLS.



改为the strategy of vertex 7

Fig 2. A detailed process of the RGLS. (a) A 7-vertex graph. (b) A LMIS: $V_{LMIS}=\{2,7\}$ is obtained by the PDG, which is an NE. (c) Choose a vertex from $\{2,7\}$ randomly, e.g., 7, change vertex 7 from C to D. (d) Beginning from the neighbor vertices of 7, play the PDG asynchronously. Vertices 4, 5, and 6 are changed from D to C, a new NE is obtained and the improved solution is $V_{LMIS}^{n}=\{2,4,5,6\}$.

The pseudo code of the RGLS is as follow. Fig. 2 shows a detailed process of the RGLS.

```
Function RGLS (a LMIS $V_{LMIS}$)
initialize
    the memory of the vertices are set empty;
    set a value to $r$ and a number to $t_{GLS}$;
    $V_C \leftarrow V_{LMIS}$;
    $t \leftarrow 0$;
while $V_C \neq \emptyset$ && $t < t_{GLS}$ do
    select $i \in V_C$ at random and obtain $\mathcal{N}_i$;
    change $i$ from $C$ to $D$;
    vertices in $\overline{\mathcal{N}}_i$ play the PDG until an NE;
    update $V_{LMIS}$: all vertices of $C$ of the NE;
    update the memories;
    update $V_C$;
    $t \leftarrow t+1$;
return $V_{LMIS}$
```

change vertex i's strategy

加上asynchronously

all cooperative vertices in the NE

vertices' memories

In step 2 of above procedure, if we choose a vertex from V_C in the decreasing order of the degrees of the vertices, the RGLS changed to GGLS. Similarly, the pseudo code of the GGLS is as follow.

```
Function GGLS (a LMIS $V_{LMIS}$)
initialize
    the memory of the vertices are set empty;
    set a value to $r$ and a number to $t_{GLS}$
    $V_C \leftarrow V_{LMIS}$;
    $t \leftarrow 0$;
while $V_C \neq \emptyset$ && $t < t_{GLS}$ do
    select $i \in V_C$ in decreasing order of the degrees
    and obtain $\mathcal{N}_i$; change $i$ from $C$ to $D$;
    vertices in $\overline{\mathcal{N}}_i$ play the PDG until an NE;
    update $V_{LMIS}$: all vertices of $C$ of the NE;
    update the memories;
    update $V_C$;
    $t \leftarrow t+1$;
return $V_{LMIS}$
```

The GGLS updates the vertex with the highest degree firstly, which will be assigned D if it has a cooperative neighbor. Thus, GGLS tends to assign D to vertices with higher degree than the RGLS. So it is expected that the GGLS has better performance than the RGLS. This is consistent with the experiments in Section VII.*C*. So it is GGLS that be used in our IGLS.

C. Time Complexity of the GLS

From Theorem 3, in a GLS procedure, if a vertex i is removed from the solution set, only the vertices in $\overline{\mathcal{N}}_i$ update their strategies asynchronously, a new NE will be arrived. So the time complexity is $O(k_i+1)$, where k_i is the degree of vertex i. If remove each vertex in V_{LMIS} one by one for a improvement, the time complexity is $O(\sum_{i\in V_{LMIS}}(k_i+1))$, which is easier than the 2-improvement with $O(m)$ time complexity.

VI. Iterated Game-Based Local Search

A. Perturbation

In iterated LS algorithms, perturbation of the solution is very useful for escaping from the local optima. The "force insertions" perturbation is used in [2]. The adaptive perturbation is employed in [22]. The perturbation used in our IGLS can be seen as a counterpart of the GLS. In the GLS, a vertex is removed from the V_{LMIS} firstly, and then start the game for a better solution. In our perturbation, k vertices in $V\backslash V_{LMIS}$ are inserted (changing from D to C) into the solution set, remove their neighbors of the k vertices from the solution set (changing from C to D), and then start the game for a better solution. We use two kinds of perturbation by probability: strong perturbation (k=2) with probability $1/2|V_{mis}|$ and week perturbation (k=1) otherwise. The pseudo code of the proposed perturbation function is shown in **Function Perturb**.

The game-based perturbation is very simple but is suitable to the GLS. The GLS accepts the solution structure change without a cardinality increasing, which is also a kind of perturbation of the solution.

```
Function Perturb ($V_{LMIS}$)
    $r1 \leftarrow$ random integer within the interval $[1, 2\cdot|V_{LMIS}|]$;
    $V_{LMIS}{}' \leftarrow V_{LMIS}$;
if $r1 \neq 1$ then $k=1$;
else $k=2$;
if $k=1$ then insert a randomly selected vertex $i \in$
    $V\backslash V_{LMIS}$ into $V_{LMIS}$;
else if $k=2$ then
    $K \leftarrow$ select $k$ vertices from $V\backslash V_{LMIS}$ at random;
    insert vertices in $K$ into $V_{LMIS}$;
play the PDG until a new NE, $V_{LMIS}=V_{NE}$;
return $V_{LMIS}$
```

将每个操作的描述位置，加括号说明一下。比如 the perturbation(Section VI. A)

B. Iterated local search

Iterated LS is a common scheme for the MIS problem [2]. Based on the proposed GLS, IGLS can be designed. In the IGLS, perturbation is added to the GLS for escaping from local optima. Starting from a random solution, obtains the V_{LMIS} by the PDG on the network. Then apply the perturbation, GGLS, and check repeatedly until the stopping criterion is met. The pseudo code of the IGLS is shown in **Algorithm 1**. The flowchart of IGLS is shown in Fig. 3.

Algorithm 1: Iterated game-based local search (IGLS)

```
Input: Graph $G = (V, E)$
Input: parameters
    $b$: the temptation to defection of the PDG;
    $T$: the initial temperature;
    $k$: the decline ration for the temperature;
    $r$: ratio of the memory length to the cardinality of $V_{mis}$;
    $t_{iter}$: the maximum iteration times;
    $t_{GLS}$: the maximum GLS times in one iteration;
Output: The $V_{LMIS}$
  $V_{LMIS}$ ← obtain a $V_{LMIS}$ by the PDG
  $t$=0
while $t<t_{iter}$ do
  $V_{LMIS} \leftarrow perturb(V_{LMIS})$;
  $V_{LMIS} \leftarrow GGLS(V_{LMIS})$;
  $V_{LMIS} \leftarrow Check(V_{LMIS})$;
return $V_{LMIS}$
```

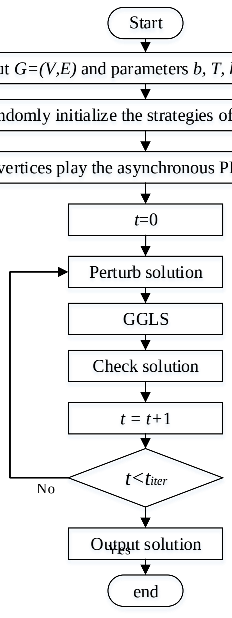

Start
Input G=(V,E) and parameters b, T, k, r, titer, tGLS
Randomly initialize the strategies of all vertices
All vertices play the asynchronous PDG to an NE
t=0
Perturb solution
GGLS
Check solution
t = t+1
t<titer
No
Yes
Output solution
end

Fig 3. The flowchart of the IGLS

C. Check

In IGLS, after reaching a new solution through the GLS, we need to check whether the new solution should be accepted to the next iteration or not. A direct method is to only accept the one with greater cardinality. However, this may increase the difficulty to escape from a local optimum. If the new solution is worse than the last one, we adopt the simulated annealing algorithm for accepting the worse solution by a small probability calculated by

需要加参考文献

$$p = e^{\frac{\Delta}{k\times T}},\ \Delta = |V_{LMIS}^{n}| - |V_{LMIS}|\,, \tag{10}$$

where T is the initial temperature and k is the parameter to be set. V_{LMIS}^{n} is the new solution and it is worse than V_{LMIS}. So $|V_{LMIS}^{n}| - |V_{LMIS}| < 0$ and $0 < p < 1$. In the experiments of this paper, $k = 0.75$ and $T = 100$. This method makes an effect on searching the global optimum when the times of iteration become greater. The pseudo code of the check function is shown in **Function Check.**

修饰times次数，用becomes

From the pseudo code of **Algorithm 1**, the effect of the parameter t_{GLS} can be seen. For a larger number of t_{GLS}, the GLS stops naturally in most cases (candidate set become empty), and the perturbation is the structural change embedded in the GGLS. For a smaller number of t_{GLS}, the times that the GLS stops by reaching t_{GLS} will be increased. As a result, the proportion using the perturbation function will increase. So t_{GLS} is not a critical parameter.

Function Check (solution V_{LMIS}^{n})

```
if $|V_{LMIS}^{n}| \geq |V_{LMIS}|$ then $V_{LMIS} \leftarrow V_{LMIS}^{n}$;
else
    $V_{LMIS} \leftarrow V_{LMIS}^{n}$ with probability $p$ compute by (10);
return $V_{LMIS}$
```

VII. Experiments

In this section, extensive experiments are done to test the proposed algorithms. First, experiments are done to compare four algorithms on obtaining a LMIS. Second, we compare the proposed GLS with the famous 2-improvement LS. Third, we compare the proposed IGLS with the ILS of 2-improvement [2], the SBTS[10], and the BLS [31]. The authors implement the algorithms in *C* and all experiments are done on a computer with 2.10 GHz CPU, 2GB of RAM running and Window 7, 32-bit Edition. The experimental time doesn't include operations for reading graph information and outputting the results because these are common for all algorithms and cannot reflect the characteristics of algorithms.

A. Networks for Experiments

Experiments are done on many networks for the comparison. Those networks can be divided into two kinds. The first kind networks with the known optimal solution such as the DIMACS networks [34], the CODE networks [35], and the BHOSLIB networks [36]. The merit of the first kind networks is that the cardinality of the GMIS ($|V_{GMIS}|$) is known, which can test whether the optimal solution is found or not by the algorithms. The defect of the first kind networks is that they usually have special topologies and are not common in real world. All parameters of the first kind networks in experiments are shown in Table II.

The second kind networks include the BA scale-free networks [37], ER random networks [38], WS small world networks [39], and the LFR medullar networks [40]. The merit of the second kind networks is that they are popular in real world and usually is difficult than the first kind networks. The defect of the second kind networks is that the cardinality of the GMIS is unknown.

The degree distribution of the BA scale-free networks is power law. To generate a BA scale-free network, two

这句话再加上参考文献

parameters are needed. One is the number of vertices *N* and the other one is the number of edges m [37]. The parameter m can be equivalently given by the average degree <*k*> of the network. For an ER random network, the m edges are uniformly distributed among the *N* vertices. The WS small world network has three parameters: the number of vertices *N*, the average degree <*k*> and the rewiring probability p[39]. Starting from an *N* vertices regular ring in which each vertex has <*k*> edges with $<k>/2$ vertices on each side, rewiring each edge by probability p, then a WS small world network is generated. There are six parameters for the LFR modular benchmark network [40]: the number of vertices *N*, the average degree <*k*>, the maximum degree k_{max}, the exponent for the degree distribution γ, the exponent for the distribution of the community size β, and the mixing parameter μ.

TABLE II
INFORMATION OF THE NETWORKS IN THE EXPERIMENTS.

network familiy	network name	*N*	<*k*>	$\|V_{GMIS}\|$
The BHOSLIB family	Frb30-15-1	450	79	30
	Frb35-17-1	595	94	35
	Frb40-19-1	760	109	40
	Frb45-21-1	945	125	45
	Frb50-23-1	1150	139	50
	Frb100-40	4000	286	100
CODE family	1dc.1024	1024	47	94
	1tc.1024	1024	16	196
	1et.2048	2048	22	316
	2dc.2048	2048	493	24
The DIMACS family	brock200_2	200	99	12
	brock200_4	200	131	17
	C1000.9	1000	900	68
	C125.9	125	112	34
	C250.9	250	224	44
	C500.9	500	450	57
	DSIC1000-5	1000	1000	15
	MANN a_27	378	374	126
	hamming8-4	256	163	16
	hamming10_4	1024	848	40
	keller4 171	171	110	11
	keller5 776	776	582	27
	gen200_p0.9_44	200	179	44
	gen200_p0.9_55	200	179	55
	gen400_p0.9_55	400	359	55
	gen400_p0.9_65	400	359	65
	gen400_p0.9_75	400	359	75
	p-hat300-1	300	73	8
	p-hat300-2	300	146	25
	p-hat300-3	300	223	36
	p-hat700-2	700	348	44
	p-hat700-3	700	523	62
	p-hat1500-1	1500	380	12
	p_hat1500-2	1500	759	65
	p_hat1500-3	1500	1130	94

The parameters of the second kind networks can be seen in their names. The BA scale-free networks are named as BA-*N*<*k*>, where *N* is the number of vertices and <*k*> is the average vertex degree of the network. Similarly, ER-*N*$<k>$ stands for an ER network and the meaning of *N* and <*k*> are the same as that of a BA network. The WS small world network is named as WS-*N*<*k*> *p*= p, where p is the rewiring probability. We generate the LFR benchmark networks with different *N*, <*k*> and k_{max}. The other parameters are $\gamma = 2$, $\beta = 1$ and $\mu = 0.2$. A LFR benchmark network is recorded by LFR-*N*<*k*>(k_{max}) to show the number of the vertices, the average vertex degree and the maximum vertex degree of the network.

B. Finding a LMIS

The game based algorithm for finding a LMIS is proposed is Section IV. In this subsection, experiments are done to compare with the random, greedy, and the greedy game algorithm.

不需要with

TABLE III
AVERAGE SIZE OF THE SOLUTION/STANDARD DEVIATIONS /AVERAGE RUNTIME OF THE GAME, GGAME, RANDOM, AND THE GREEDY ALGORITHMS.

networks	best	game	ggame	random	greedy
frb30-15	30	21.33/0.52 / 0.004	21.33/1.18 /0.013	20.00/1.21 / 0.016	24.47/1.03 / 0.029
frb100-40	100	68.40/1.37 / 0.258	67.20/3.29 /1.068	66.27/1.93 / 1.626	80.40/1.24 / 3.73
1dc.1024	94	70.67/0.61 / 0.016	66.00/0.82 /0.070	59.73/1.98 / 0.065	74.67/1.19 /0.221
2dc.2048	24	19.93/0.25 / 0.080	18.400.21 /0.325	15.67/1.14 / 0.338	21.13/0.49 /0.639
brock200_2	12	7.27/0.68 / 0.001	7.27/0.52 /0.002	7.20/0.75 / 0.003	9.2/0.40 /0.006
C1000.9	68	45.87/1.75 / 0.019	47.06/2.43 /0.070	44.53/2.25 / 0.093	58.40/1.25 /0.194
hamming10_4	40	26.40/1.96 / 0.019	26.40/1.79 /0.075	20.00/2.68 / 0.076	35.73/1.00 /0.185
p-hat1500-1	12	7.13/0.81 / 0.067	7.00/0.63 /0.183	6.73/0.93 / 0.235	9.07//0.69 /0.360
BA500 <4>	-	273.47/1.36 / 0.003	274.93/2.94 /0.016	253.4/7.79 / 0.015	289.27/1.18 /0.158
BA1000 <8>	-	415.67/5.64 / 0.013	416.20/6.09 /0.062	373.87/8.47 / 0.076	456.00/4.13 / 0.905
BA2000 <10>	-	737.87/6.51 / 0.052	737.80/7.15 /0.252	644.80/9.64 / 0.311	817.73/5.37 / 5.96
ER500 <4>	-	210.73/3.73 / 0.003	221.60/3.43 /0.017	206.33/4.60 / 0.012	243.27/1.18 / 0.134
ER1000 <8>	-	282.27/6.82 / 0.013	303.20/5.23 /0.063	275.47/7.70 / 0.056	336.67/2.15 /0.659
ER2000 <10>	-	486.07/7.10 / 0.051	526.33/5.59 /0.250	477.47/7.44 / 0.288	595.20/3.17 / 4.195
WS500<10> p=0.1	-	96.13/1.93 / 0.003	**96.80**/2.11 /0.017	83.00/3.24 / 0.010	100.67/0.79 / 0.060
WS1000 <10> p=0.5	-	329.27/4.27 / 0.013	329.33/5.88 /0.062	308.47/5.92 / 0.076	368.00/0.52 / 0.792
WS2000 <10> p=0.5	-	670.93/7.79 / 0.050	671.33/7.29 /0.25	622.73/9.04 / 0.319	744.80/0.40 / 5.952
LFR500 <15>(50)	-	125.20/4.38 / 0.004	150.33/3.39 /0.017	117.20/7.62 / 0.021	157.67/3.42 /0.109
LFR1000 <20>(60)	-	226.27/11.08 / 0.013	276.80/4.00 /0.062	215.73/12.32 / 0.093	294.07/2.98 / 0.696
LFR2000 <25>(70)	-	358.00/8.93 / 0.052	449.00/5.05 /0.253	347.00/12.62 / 0.379	485.87/6.34 / 3.826

Table III shows the experiment results. The parameter of the PDG is *b*=1.5. The initial states of the game are generated randomly. Each algorithm runs 15 times independently on every network. The experimental results shown in Table III include the average size of the obtained LMISs, the standard deviations, and the average runtime.

The greedy game (labeled as ggame in Table III) is also a game based algorithm, in which the vertices update their strategies is the decreasing order of their degrees. Comparing the game and the greedy game algorithms, the game algorithm always has smaller values of standard deviations and runtimes, the sizes of the solutions of the game may be less or larger than the greedy game. The order of the vertices updating their strategies makes uncertain effects to the size of solution but increase the runtime certainly. So we do not recommend greedy game in finding a LMIS.

Comparing the game algorithm with the random algorithm, the game algorithm always obtains a better solution with less runtime and smaller standard deviations in all the experiments. So we can conclude that the game algorithm is better than the random algorithm. Comparing the game algorithm with the greedy algorithm, the greedy algorithm always obtains a better solution but with much longer runtime. In particular, the game algorithm is the fastest algorithm among the four algorithms; it always obtains a LMIS with the shortest runtime.

The random algorithm is the most popular in finding a LMIS [2],[10],[22],[33], we recommend using the game algorithm instead of the random algorithm.

C. Comparing the GLS with the 2-improvement

The 2-improvement is an effective LS technique [2], which attempts to iteratively remove one vertex from and add two into the solution set. The 2-improvement is a representative of the (*k*,*l*)-swap based LS and it is really a fast (1,2)-swap. The 2-improvement LS has no parameter for stop the search process, which maintains a list of candidates that is updated whenever the solution changes. When the list of candidates becomes empty, the search process stops.

TABLE IV
COMPARING THE GLS AND THE 2-IMPROVEMENT: AVERAGE SOLUTION SIZE/STANDARD DEVIATION/AVERAGE RUNTIME (S)

name	$\|V_{GMIS}\|$	RGLS	GGLS	2-improvement
Frb30-15	30	23.80/0.98 /0.010	**24.73**/1.15 /0.010	23.07/0.77 / 0.023
Frb50-23	50	41.20/1.19 /0.053	**41.23**/1.31 /0.051	39.33/1.34 / 0.176
1dc.1024	94	**72.53**/0.99 /0.024	72.00/1.00 /0.026	72.00/0 / 0.032
C1000.9	68	57.47/1.52 /0.073	57.66/1.46 /0.070	**59.93**/1.67 / 0.198
DSIC1000-5	15	12.27/0.49 /0.068	**12.43**/0.66 /0.079	11.33/0.58 / 0.205
MANN a_27	126	117.60/0.75 /0.004	117.90/0.87 /0.006	**119.20**/1.00 / 0.002
p-hat1500-1	12	9.63/0.55 /0.123	**9.93**/0.25 /0.191	8.20/0.73 / 0.630
BA1000 <10>	-	382.63/4.95 /0.018	393.00/4.62 /0.058	**401.47**/2.50 / 0.081
ER500 <8>	-	145.27/3.60 /0.007	**159.47**/3.53 /0.015	153.27/2.62 / 0.020
ER1000 <8>	-	316.00/5.15 /0.063	**322.63**/4.83 /0.050	321.53/3.79 / 0.094
ER2000 <8>	-	610.33/0.38 /0.228	**634.37**/5.11 /0.221	624.07/9.48 / 0.400
WS500 p=0.1	-	100.90/2.62 /0.009	**102.90**/2.93 /0.012	100.33/2.11 / 0.009
WS1000 p=0.1	-	205.70/3.72 /0.047	**206.13**/2.91 /0.045	204.80/2.29 / 0.039
WS2000 p=0.1	-	405.67/4.45 /0.075	**412.90**/4.66 /0.167	410.67/4.22 / 0.138
LFR500 <15>(50)	-	145.17/5.00 /0.020	**151.40**/2.60 /0.017	150.33/1.73 / 0.048
LFR1000 <25>(70)	-	248.00/5.87 /0.085	**260.40**/3.97 /0.082	254.40/4.90 / 0.362
LFR2000 <25>(70)	-	474.13/8.78 /0.332	**489.13**/5.01 /0.280	485.33/6.21 / 1.500

becomes

The GLS stops when the candidate set become empty or the iterative times reaches to t_{GLS}. In the experiments, we set $t_{GLS} = N/8$ and $r = 0.90$. As mentioned in Section V, a larger number of t_{GLS} may increase the runtime of the GLS. With$t_{GLS} = N/8$, the overall runtime of the GLS is about less than or equivalent to that of the 2-improvemtn, so we can only compare their average solution size.

Table IV shows the experiments about the RGLS, GGLS, and the 2-improvement. Each algorithm runs 30 times independently on each network. The experimental results include the average size of the solutions, the standard deviations, and the average runtime. The initial LMIS of both the GLS and the 2-improvement is found by the game algorithm with random initial states. The best results are highlighted in bold in Table IV.

From Table IV, the GGLS obtains better solutions than the 2-improvement in most networks except C1000.9, MANN a_27, and BA1000<10>, which indicates that the GGLS has a better search ability than the 2-improvement because it accepts both 2-improvement, 3-improvement, ⋯, and the structural change to the solution. Furthermore, in all networks, GGLS obtains solution better than or equal to the RGLS, which indicates that the GGLS has a better search ability than the RGLS because it tends to assign D to vertices with higher degree.

TABLE V
DIMACS NETWORKS: MAXIMUM SIZE (TIMES APPEAR) /AVERAGE SIZE /STANDARD DEVIATION /AVERAGE RUNTIME (S)

network name (best solution)	IGLS	ILS	SBTS	BLS
brock200_4 (17)	17(30)/**17.00** /0/0.672	17(30)/**17.00** /0/4.000	17(15)/16.50 /0.50/28.258	17(3)/16.07 /0.37/0.471
C125.9 (34)	34(30)/**34.00** /0/0.003	34(30)/**34.00** /0/0.287	34(30)/**34.00** /0/0.002	34(30)/**34.00** /0/0.162
C250.9 (44)	44(30)/**44.00** /0/0.054	44(12)/43.27 /0.73/1.381	44(30)/**44.00** /0/0.093	44(29)/43.93 /0.37/0.555
C500.9 (57)	57(8)/56.03 /0.71/34.482	57(5)/54.40 /0.95/5.690	57(4)/**56.13** /0.34/79.706	57(2)/54.00 /0.67/2.341
DSJC1000_5 (15)	15(2)/14.07 /0.25/123.769	15(30)/**15.00** /0/103.00	15(27)/14.90 /0.30/559.221	15(2)/14.07 /0.25/6.216
gen200_p0.9_44 (44)	44(30)/**44.00** /0/0.465	44(9)/40.93 /2.06/0.827	44(30)/**44.00** /0/0.088	44(28)/43.70 /1.15/0.342
gen200_p0.9_55 (55)	55(30)/**55.00** /0/0.263	55(16)/48.63 /6.81/0.864	55(30)/**55.00** /0/0.027	55(30)/**55.00** /0/0.331
gen400_p0.9_55 55	55(3)/52.97 /0.80/25.048	52(13)/51.13 /0.92/3.404	55(30)/**55.00** /0/1.231	55(6)/51.80 /1.69/1.466
gen400_p0.9_65 (65)	65(30)/**65.00** /0/0.763	65(15)/57.70 /7.33/3.474	65(30)/**65.00** /0/0.102	65(30)/**65.00** /0/1.296
gen400_p0.9_75 (75)	75(30)/**75.00** /0/0.318	75(30)/**75.00** /0/3.489	75(30)/**75.00** /0/0.263	75(30)/**75.00** /0/1.230
hamming8-4 (16)	16(30)/**16.00** /0/0.133	16(30)/**16.00** /0/3.970	16(30)/**16.00** /0/0.008	16(30)/**16.00** /0/0.388
hamming10-4 (40)	40(30)/**40.00** /0/4.969	40(30)/**40.00** /0/39.00	40(40)/**40.00** /0/0.111	40(30)/**40.00** /0/6.471
keller4 171 (11)	11(30)/**11.00** /0/0.067	11(30)/**11.00** /0/1.655	11(30)/**11.00** /0/0.003	11(30)/**11.00** /0/0.145
keller5 776 (27)	27(29)/**26.97** /0.18/19.687	27(6)/25.27 /1.18/33.18	27(29)/**26.97** /0.18/50.019	27(2)/25.17 /0.65/3.270
MANN_a27 (126)	126(28)/125.93 /0.25/2.198	126(30)/**126.00** /0/1.002	122(2)/120.17 /0.90/17.189	125(9)/124.07 /0.74/2.255
p-hat300-1 (8)	8(30)/**8.00** /0/1.654	8(9)/7.20 /0.60/8.920	8(30)/**8.00** /0/0.127	8(27)/7.90 /0.31/0.386
p-hat300-2 (25)	25(30)/**25.00** /25/0.186	25(30)/**25.00** /0/16.487	25(30)/**25.00** /0/0.107	25(30)/**25.00** /0/5.314
p-hat300-3 (36)	36(30)/**36.00** /0/0.339	36(27)/35.83 /0.52/6.241	36(30)/**36.00** /0/0.443	36(29)/35.97 /0.18/0.615
p-hat700-2 (44)	44(30)/**44.00** /0/1.254	44(25)43.83 /0.37/134.72	44(30)/**44.00** /0/2.887	44(30)/**44.00** /0/3.475
p-hat700-3 62	62(30)/**62.00** /0/0.426	62(21)/61.67 /0.54/45.143	62(30)/**62.00** /0/4.557	62(29)/61.97 /0.18/4.401
p_hat1500-2 (65)	65(24)/**65.00** /0/19.074	65(22)/64.70 /0.53/150.00	65(30)/**65.00** /0/31.349	65(29)/64.97 /0.18/18.622
p_hat1500-3 (94)	94(30)/**94.00** /0/13.041	94(30)/**94.00** /0/88.00	94(30)/**94.00** /0/35.245	94(16)/93.53 /0.51/21.571

D. Comprehensive Comparison

In this section, experiments are done to comprehensively compare the proposed IGLS with three recent algorithms: ILS[2], SBTS[10], and BLS[31]. For the first kind networks whose optimal solutions are known, the algorithms stop when the optimal solution is found, or it will stop when iterated t_{iter} times.

不加very

The parameters of IGLS are: r=0.90 and t_{GLS}=1000 for most networks, for several very simple networks t_{GLS}=100 and we mark these networks with* after their name. For the first kind of networks, t_{iter}=1000; for the second kind of networks, t_{iter}=100. Other parameters have stated before (b=1.5, T=100, k=0.75).

For the ILS, the maximum iteration times is also set as 1000. The scheme of the SBTS and BLS is different from that of IGLS and ILS. For the SBTS, the maximum iteration times are 10000. For the BLS, the parameters are: $L_0 = 0.02N$, $\mathrm{T} = 100$, $L_{Max} = 0.1N, \alpha_s = 0.70$, $\emptyset = 5$ for all networks. If $N \geqslant 1000$, $P_0 = 0.75$ and $\alpha_r = 0.92$; else if $N < 1000$, $P_0 = 1$ and α_r is useless.

Each algorithm run 30 times on each network, the maximum size of the solution, the number of times it appears among the 30 runs, the average size of the 30 solutions, the standard

修改了SBTS的实验，现在采用的结果是迭代100000次。

改成the experimental results of the

deviation of 30 solutions, and the average runtime are shown in Table V to Table XI. The seven tables correspond to the DIMACS, CODE, BHOSLIB, BA, ER, WS, and LFR networks respectively.

On the second kind networks (BA, ER, WS, and LFR), it is obvious that the IGLS performs better than the others. In overall, we do the *t* test to compare the IGLS with ILS, SBTS, and BLS on all the results from Table V to Table XI.

Statistical tests between the IGLS and the ILS. There are 74 different networks in Table V to Table XI. We do the statistical tests on results in the seven tables. The average size of the solutions computed by the IGLS and the ILS are used for the test. Their differences on the 74 networks are calculated:

$D_{IGLS-ILS} = |V_{IGLS}| - |V_{ILS}| = \{0,\ 0,\ 0.73,\ 1.63,\ -0.93,\ 3.07,$ 6.37, 1.84, 7.30, 0, 0, 0, 0, 1.70, -0.07, 0.80, 0, 0.17, 0.17, 0.33, 0.30, 0, 2.73, -2.67, 3.00, -0.33, -0.30, -1.06, -0.73, 1.17, 0, 0, 0, 0, 0.20, 0.54, 0, 0, 1.07, 0, 1.33, 1.86, 0, 0.10, 0.53, 0, 2.17, 1.33, 2.20, 3.77, 3.60, 4.47, 7.73, 8.43, 0.13, 0.03, -0.10, 0.70, 1.30, 1.26, 3.57, 3.37, 0.10, 0, 0, 0.27, 0.07, 0.04, 1.13, -0.23, 0.63, 1.20, 1.80, 1.77}.

The elements in $D_{IGLS-ILS}$ are independent from each other and obey the same distribution. Without loss of generality, suppose the elements in D obey the normal distribution $D_i \sim N(\mu_D, \sigma_D^2)$ $(i = 1,2,\ldots,74)$ where μ_D and σ_D^2 are unknown. In other word, $D_1, D_2, \ldots, D_{74}$ 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 size of the solution obtained by the IGLS is larger than that of the ILS;

H_1: The size of the solution obtained by the IGLS is not larger than that of the ILS.

The hypotheses are equivalent to the following hypothetical format:

H_0: $\mu_D > 0$;

H_1: $\mu_D \le 0$.

The critical region of this test problem is calculated by:

改成the t distribution

$$t = \frac{\bar{d}}{s_D/\sqrt{n}} \le t_\alpha(n-1), \tag{11}$$

where $\bar{d}$ is the sample average, s_D is the sample variance, *n*=74 is the number of the samples, and $t_\alpha(n-1)$ is the α quantile of the $t(n-1)$ distribution with the degree of the freedom $n-1$.

TABLE VI

CODE NETWORKS: MAXIMUM SIZE (TIMES APPEAR) / AVERAGE SIZE / STANDARD DEVIATION/ AVERAGE RUNTIME (S)

network name (best solution)	IGLS	ILS	SBTS	BLS
1dc.1024 (94)	94(10)/93.33 /0.47/23.576	94(4)/80.60 /3.70/14.00	94(11)/93.37 /0.48/109.31	94(16)/**93.53** /0.51/13.222
1et.2048 (316)	316(1)/314.33 /0.65/26.265	316(30)/**316.00** /0/16.00	315(4)/308.73 /4.85/346.134	316(9)/315.07 /0.74/20.019
1tc.1024 (196)	196(30)/**196.00** /0/1.581	196(30)/193.00 /0/8.323	196(28)/195.93 /0.25/36.276	196(29)/195.97 /0.18/10.022
2dc.2048 (24)	24(20)/**23.67** /0.47/174.320	24(30)/24.00 /0/165.00	24(8)/23.27 /0.44/679.91	24(18)/23.60 /0.55/21.929

From D, it is easy to obtain that $\bar{d} = 1.1074$ and $s_D = 1.969$. As a result, $t = \frac{\bar{d}}{s_D/\sqrt{n}} = 4.8378$. Given the significance level $\alpha = 0.005$, from the critical table of the *t* distribution, we obtain that $t_\alpha(n-1) = t_{0.005}(73) = 2.645$.

It's clear that $t = 4.8378 > t_{0.005}(73)$, which means that the value of the *t* is not in the critical region. So we accept H_0: The size of the solution obtained by the IGLS is larger than that of the ILS.

TABLE VII

BHOSLIB NETWORKS: MAXIMUM SIZE (TIMES APPEAR) /AVERAGE SIZE /STANDARD DEVIATION /AVERAGE RUNTIME (S)

network name (best solution)	IGLS	ILS	SBTS	BLS
frb30-15-1 (30)	30(21)/29.70 /0.46/42.383	30(30)/**30.00** /0/9.00	30(29)/29.97 /0.18/36.529	30(30)/**30.00** /0/1.338
frb35-17-1 (35)	35(2)/33.87 /0.50/102.36	35(28)/**34.93** /0.18/13.00	35(1)/33.53 /0.56/182.555	35(15)/34.40 /0.70/2.564
frb40-19-1 (40)	40(13)/39.27 /0.73/99.530	40(30)/**40.00** /0/19.00	40(6)/38.80 /0.75/225.593	40(30)/**40.00** /0/4.018
frb45-21-1 (45)	45(1)/43.30 /0.53/89.761	45(22)/**44.47** /0.89/27.00	45(1)/43.10 /0.47/366.299	45(3)/43.60 /0.70/6.694

TABLE VIII

BA NETWORKS: MAXIMUM SIZE (TIMES APPEAR) / AVERAGE SIZE / STANDARD DEVIATION / AVERAGE RUNTIME (S)

BA network	IGLS	ILS	SBTS	BLS
BA100 <4>*	59(30)/**59.00** /0/0.038	59(30)/**59.00** /0/0.823	59(30)/**59.00** /0/0.003	59(30)/**59.00** /0/0.104
BA100 <8>*	43(30)/**43.00** /0/0.057	43(30)/**43.00** /0/0.136	43(30)/**43.00** /0/0.005	43(30)/**43.00** /0/0.106
BA100 <10>*	41(30)/**41.00** /0/0.061	41(30)/**41.00** /0/0.185	41(30)/**41.00** /0/0.006	41(30)/**41.00** /0/0.104
BA500 <4>*	290(30)/**290.00** /0/0.066	290(30)/**290.00** /0/0.874	290(30)/**290.00** /0/18.145	290(30)/**290.00** /0/3.843
BA500 <8>	222(30)/**222.00** /0/0.983	222(24)/221.80 /0.40/1.114	222(30)/**222.00** /0/25.001	222(18)/221.60 /0.50/3.957
BA500 <10>	205(29)/**204.97** /0.18/1.059	205(14)/204.43 /0.56/1.345	205(20)/204.67 /0.47/27.339	205(1)/203.03 /0.89/5.791
BA1000 <4>	574(30)/**574.00** /0/1.229	574(30)/**574.00** /0/2.716	574(4)/573.00 /0.52/46.308	574(24)/573.80 /0.45/5.926
BA1000 <8>	461(30)/461.00 /0/1.432	462(9)/**461.10** /0.70/3.494	461(6)/459.80 /0.88/59.582	461(9)/459.80 /1.23/5.932
BA1000 <10>	420(30)/**420.00** /0/1.520	420(11)/418.93 /0.93/3.293	419(3)/417.23 /0.99/54.095	420(12)/419.00 /0.94/5.974
BA2000 <4>	1165(30)/**1165.00** /0/2.172	1165(30)/**1165.00** /0/9.532	1165(2)/1162.77 /1.36/71.361	1165(30)/**1165.00** /0/17.216
BA2000 <8>	902(9)/**901.20** /0.65/2.475	903(1)/899.87 /1.45/8.254	899(2)/895.83 /1.69/111.177	900(6)/899.00 /0.71/17.659
BA2000 <10>	837(1)/**834.03** /0.98/2.467	835(1)/832.17 /1.95/8.551	834(1)/827.87 /2.63/106.638	834(18)/833.60 /0.55/17.826

Statistical tests between the IGLS and the SBTS. Similar as the last test, we obtain:

$D_{IGLS-SBTS} = |V_{IGLS}| - |V_{SBTS}| = \{0.5,\ 0,\ 0,\ -0.1,\ -0.83,\ 0,\ 0,$ -2.03, 0, 0, 0, 0, 0, 0, 5.76, 0, 0, 0, 0, 0, 0, 0, -0.04, 5.60, 0.07, 0.40, 0, 0.34, 0.47, 0.20, 0, 0, 0, 0, 0, 0.30, 1.00, 0.20, 2.77, 2.23, 5.37, 6.16, 0, 1, 1, 0, -0.03, -0.27, 3.7, 1.77, 2.06, 6.77, 12.30, 8.96, 0, 0, 1.93, 2.17, 5.20, 4.3, 14.7, 14.7, 0, 0, 0, 0.43, 0, 0.2, 2.1, 1.77, 0.73, 4.30, 5.34, 5.67}.

The hypotheses are:

H_0: The size of the solution obtained by the IGLS is larger than that of the SBTS;

H_1: The size of the solution obtained by the IGLS is not larger than that of the SBTS.

The hypotheses are equivalent to:

H_0: $\mu_D > 0$;

H_1: $\mu_D \le 0$.

The critical region of this test problem is also calculated by (11). In this test, *n*=74, $\bar{d} = 1.7455$, and $s_D = 3.2867$. As a result, $t = \frac{\bar{d}}{s_D/\sqrt{n}} = 4.5686$. Given the significance level $\alpha = 0.005$, from the critical table of the *t* distribution, we obtain that $t_\alpha(n-1) = t_{0.005}(73) = 2.645$. It's clear that $t = 4.5686 > t_{0.005}(73)$, we accept H_0 again.

Statistical tests between the IGLS and the BLS. Similar as the last two tests, we obtain:

$D_{IGLS-BLS} = |V_{IGLS}| - |V_{BLS}| =$ {0.93, 0, 0.07, 2.03, 0, 0.30, 0, 1.17, 0, 0, 0, 0, 0, 1.80, 1.86, 0.10, 0, 0.03, 0, 0.03, 0.03, 0.47,

-0.20, -0.74, 0.03, 0.07, -0.30, -0.53, -0.73, -0.30, 0, 0, 0, 0, 0.40, 1.94, 0.20, 0.20, 1.00, 0, 2.20, 0.43, 0, 0.03, 0.60, 3.27, 2.20, 1.13, 2.15, 2.07, 3.53, 2.90, 2.03, 6.13, 0.07, 0, 0.17, 0.23, 1.20, 1.10, 2.60, 2.90, 0, 0, 0, 0.20, 0, 0.20, 0.23, 1.40, 0.40, 1.47, 0.07, 0}.

TABLE IX
ER NETWORKS: MAXIMUM SIZE (TIMES APPEAR) / AVERAGE SIZE / STANDARD DEVIATION / AVERAGE RUNTIME (S)

ER network	IGLS	ILS	SBTS	BLS
ER100 <4>*	47(30)/**47.00** /0/0.041	47(30)/**47.00** /0/0.135	47(30)/**47.00** /0/0.004	47(30)/**47.00** /0/0.090
ER100 <8>*	35(30)/**35.00** /0/0.062	35(27)/34.90 /0.30/0.139	34(30)/34.00 /0/0.005	35(29)/34.97 /0.18/0.087
ER100 <10>*	31(30)/**31.00** /0/0.076	31(14)/30.47 /0.50/0.179	30(30)/30.00 /0/0.006	31(12)/30.40 /0.50/0.092
ER500 <4>*	245(30)/**245.00** /0/0.145	245(30)/**245.00** /0/1.570	245(30)/**245.00** /0/20.069	245(3)/241.73 /1.89/3.372
ER500 <8>	177(29)/176.97 /0.18/4.970	177(3)/174.80 /1.47/1.024	177(30)/**177.00** /0/26.204	177(3)/174.77 /1.38/2.630
ER500 <10>	155(23)/154.73 /0.51/0.450	155(4)/153.40 /1.08/1.186	155(30)/**155.00** /028.271/	155(2)/153.60 /0.81/4.069
ER1000 <4>	479(12)/**478.40** /0.49/6.087	479(1)/476.20 /1.62/2.210	478(3)/474.70 /2.04/49.699	479(1)/476.25 /1.37/6.682
ER1000 <8>	355(4)/**353.20** /1.14/7.055	352(3)/349.43 /1.71/2.443	355(2)/351.43 /2.09/49.358	353(8)/351.13 /1.46/8.002
ER1000 <10>	313(14)/**312.03** /1.05/7.505	313(1)/308.43 /1.93/2.693	313(8)/309.97 /3.07/56.752	312(6)/308.50 /2.27/8.286
ER2000 <4>	953(23)/**952.70** /0.59/10.940	951(1)/948.23 /1.36/7.046	950(1)/945.93 /2.53/81.049	952(6)/949.80 /1.48/17.288
ER2000 <8>	696(2)/**692.03** /2.07/11.633	691(2)/684.30 /3.03/6.372	688(1)/679.73 /4.73/102.082	691(18)/690.00 /1.41/18.104
ER2000 <10>	629(2)/**625.33** /2.15/12.103	624(1)/616.90 /2.95/6.544	623(2)/616.37 /3.59/118.952	620(24)/619.20 /1.79/14.018

TABLE X
WS NETWORKS: MAXIMUM SIZE (TIMES APPEAR) / AVERAGE SIZE / STANDARD DEVIATION / AVERAGE RUNTIME (S)

WS network	IGLS	ILS	SBTS	BLS
WS100 (p=0.1)*	21(30)/**21.00** /0/0.071	21(26)/20.87 /0.34 /0.167	21(30)/**21.00** /0/0.009	21(28)/20.93 /0.25/0.092
WS100 (p=0.5)*	38(30)/**38.00** /0/0.087	38(29)/37.97 /0.18/0.248	38(30)/**38.00** /0/0.101	38(30)/**38.00** /0/0.095
WS500 (p=0.1)	112(30)/**112.00** /0/6.198	112(4)/**112.10** /1.16/0.946	112(1)/110.07 /1.15/30.242	112(25)/111.83 /0.38/3.448
WS500 (p=0.5)	187(30)/**187.00** /0/13.559	187(17)/186.30 /0.97/1.828	187(3)/184.83 /1.49/32.046	187(23)/186.77 /0.43/3.913
WS1000 (p=0.1)	224(23)/**223.77** /0.42/9.546	224(23)/222.47 /1.36/2.080	222(4)/218.57 /2.01/54.635	224(3)/222.57 /0.73/6.689
WS1000 (p=0.5)	379(22)/**378.73** /0.44/25.243	379(21)/377.47 /1.18/7.156	377(5)/374.43 /1.71/65.256	379(6)/377.63 /0.96/7.424
WS2000 (p=0.1)	453(3)/**451.20** /1.11/15.500	452(1)/447.63 /7.40/4.560	447(1)/436.50 /4.66/107.519	451(6)/448.60 /1.95/16.689
WS2000 (p=0.5)	763(10)/**762.10** /0.79/48.756	763(2)/758.37 /6.08/53.524	754(1)/747.40 /3.16/116.397	760(18)/759.20 /1.30/15.945

可以把上面两个表格放在这段话下面。

The hypotheses are:

H_0: The size of the solution by the IGLS is larger than that of the BLS;

H_1: The size of the solution by the IGLS is not larger than that of the BLS.

The hypotheses are equivalent to:

H_0: $\mu_D > 0$;

H_1: $\mu_D \leq 0$.

The critical region of this test problem is also calculated by (11). In this test, n=74, $\bar{d} = 0.6861$, and $s_D = 1.1691$. As a result, $t = \frac{\bar{d}}{s_D/\sqrt{n}} = 5.0484$. At the significance level $\alpha = 0.005$, we obtain that $t_\alpha(n-1) = t_{0.005}(73) = 2.645$ again. It's clearly that $t = 5.0484 > t_{0.005}(73)$, so we accept H_0 again.

VIII. CONCLUSION

It is proved that a local maximal independent set corresponds to a Nash equilibrium of the prisoners' dilemma and vice versus. The relation is one-to-one correspondence. Then game based method for obtaining a local maximal independent set is established, which is much better than the usually used random insertion algorithm in both the quality of the solution and the runtime. Furthermore, greedy game-base local search (GGLS) is provided, which can realize (k,l)-swaps for various number of k and l. The GGLS can be seen as an integrated (k,l)-swap, which is its merit. As a result, iterative game-based local search (IGLS) is proposed for the maximum independent set problem. Experiments on various networks show that the IGLS performs better than state of the art algorithms, especially on the general (BA, WS, ER, and LFR) networks.

改为the set of the cooperative vertices in a Nash equilibrium state

If we establish a relation between the game and the problem to be solved, for example the Nash Equilibriums and the independent sets, a game-based algorithm for the problem can be established. Game-based algorithms have their own merits, which are problem-dependent.

TABLE XI
LFR NETWORKS: MAXIMUM SIZE (TIMES APPEAR) / AVERAGE SIZE / STANDARD DEVIATION / AVERAGE RUNTIME (S)

LFR network	IGLS	ILS	SBTS	BLS
LFR100 <15> 50*	32(30)/**32.00** /0/0.098	32(27)/31.90 /0.30/0.412	32(30)/**32.00** /0/0.007	32(30)/32.00 /0/0.974
LFR100 <20> 60*	35(30)/**35.00** /0/0.118	35(30)/**35.00** /0/0.616	35(30)/**35.00** /0/0.008	35(30)/**35.00** /0/0.126
LFR100 <25> 70*	30(30)/**30.00** /0/0.145	30(30)/**30.00** /0/0.848	30(30)/**30.00** /0/0.010	30(30)/**30.00** /0/0.092
LFR500 <15> 50	159(30)/**159.00** /0/6.626	159(1)/158.73 /0.73/2.061	159(17)/158.57 /0.50/36.568	159(24)/158.80 /0.41/3.536
LFR500 <20> 60	162(30)/**162.00** /0/1.601	162(1)/161.93 /0.36/2.707	162(30)/**162.00** /0/42.003	162(30)/**162.00** /0/3.562
LFR500 <25>70	129(29)/**128.97** /0.18/1.824	130(1)/128.93 /0.36/3.460	129(23)/128.77 /0.42/50.274	129(23)/128.77 /0.43/3.326
LFR1000 <15>50	354(30)/**354.00** /0/8.810	354(8)/352.87 /0.92/4.826	354(3)/351.90 /1.58/81.474	354(23)/353.77 /0.43/7.841
LFR1000 <20>60	289(27)/288.90 /0.30/9.528	289 (21)/**289.13** /0.81/4.996	289(5)/287.13 /1.61/100.532	288(15)/287.50 /0.53/7.666
LFR1000 <25>70	272(30)/**272.00** /0/11.534	273(2)/271.37 /0.84/6.709	272(16)/271.27 /0.89/102.483	272(18)/271.60 /0.52/7.414
LFR2000 <15>50	670(26)/**669.87** /0.34/13.740	670(20)/668.67 /1.27/10.045	668(6)/665.57 /1.82/159.678	670(3)/668.40 /0.97/14.946
LFR2000 <20>60	600(11)/**599.27** /0.68/14.375	600(1)/597.47 /1.335/11.760	598(3)/593.93 /2.50/164.633	600(15)/599.20 /1.03/16.678
LFR2000 <25>70	520(23)/**519.77** /0.42/15.882	520(4)/518.00 /1.34/14.171	518(6)/514.10 /3.37/198.214	520(23)/**519.77** /0.43/16.763

REFERENCES

[1] J.-C. Lin and T. C. Huang, "An Efficient Fault-Containing Self-Stabilizing Algorithm for Finding a Maximal Independent Set," IEEE Trans. on Para. and Dist. Sys., vol. 14, no. 8, pp: 742-754, Aug. 2003.

[2] D. V. Andrade, M. G. C. Resende, and R. F. Werneck, "Fast local search for the maximum independent set problem," J. Heuristics, vol. 18, pp:525–547, 2012.

[3] J. Dahium, S. Lamm, P. Sanders, C. Schulz, D. Strash, R. F. Werneck, "Accelerating local search for the maximum independent set problem," Experimental Algorithms, SEA 2016, Lecture Notes in Computer Science, A. Goldberg and A. Kulikov, Eds., vol. 9685, Springer, 2016, pp.118-133.

[4] T. A. Feo, M. G. C. Resende, S. H. Smith, "A greedy randomized adaptive search procedure for maximum independent set," Operations Research, vol. 42, no. 5, pp. 860-878, 1994.

[5] P.V. Sanser, D. Nehad, E. Chlamtac, and H. Hoppe "Efficient traversal of mesh edges using adjacency primitives," Proceedings of ACM SIGGRAPH Asia 2008, ACM Trans. Graph., vol. 27, no. 5, pp: 1-144:9 2008.

[6] A. Gemsa, M. Nöllenburg, and I. Rutter, "Evaluation of labeling strategies for rotating maps," Journal of Experimental Algorithmics , vol. 21, no. 1, pp: 235-246, 2014.

[7] T. Kieritz, D. Luxen , P. Sanders , and C. Vetter, "Distributed time-dependent contraction hierarchies," SEA 2010, LNCS, P. Festa, Eds., vol. 6049, pp. 83–93, 2010, Springer, Heidelberg.

[8] A.P. Ambler, H.G. Barrow, C.M. Brown, R.M. Burstall, R.J. Popplestone. A versatile computer-controlled assembly, in: Proc. Third Int. J. Conf. Artif.Intell.Standford, CA, 1982, pp.298-307.

[9] R.C. Bolles, R.A. Cain, "Recognizing and locating partially visible objects: the local-feature-focus method," Int.J. Robot. Res. 1(3)(1982)57-82.

[10] 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.

[11] M. R. Garey and D. S. Johnson, Computers and Intractability: A Guide to the Theory of NP-completeness, in Freeman, W. H. 1979.

[12] R. Battiti and M. Protasi, "Reactive local search for the maximum clique problem," Algorithmica, vol. 29, no. 4, pp: 610–637, 2001.

[13] A. Grosso, M. Locatelli, and F. D. Croce, "Combining swaps and node weights in an adaptive greedy approach for the maximum clique problem," J. Heuristics, vol. 10, pp:135–152, 2004.

[14] A. Grosso, M. Locatelli, and W. Pullan, "Simple ingredients leading to very efficient heuristics for the maximum clique problem," J. Heuristics, vol. 14, no. 6, pp: 587–612, 2008.

[15] P. Hansen, N. Mladenović, and D. Urošević, "Variable neighborhood search for the maximum clique," Discrete Applied Mathematics, vol. 145, no. 1, pp: 117–125, 2004.

[16] K. Katayama, A. Hamamoto, and H. Narihisa, "An effective local search for the maximum clique problem," Information Processing Letters, vol. 95, pp: 503–511, 2005.

[17] W. J. Pullan and H. H. Hoos, "Dynamic local search for the maximum clique problem," J. Artificial Intelligence Research, vol. 25, pp: 159–185, 2006.

[18] S. Richter, M. Helmert, and C. Gretton, "A stochastic local search approach to vertex cover," In Proceedings of the 30th German Conference on Artificial Intelligence (KI), pp: 412–426, 2007.

[19] F. V. Fomin and D. Kratsch, Exact Exponential Algorithms, Springer, 2010.

[20] G. Szabo and G. Fath, "Evolutionary games on graphs," *Physics Reports*, vol. 446, no. 4, pp. 97-216, 2006.

[21] C. Hauert and G. Szabo, "Prisoner's Dilemma and Public Goods Games in Different Geometries: Compulsory Versus Voluntary Interactions," Complexity, vol. 8, no. 1, pp: 31-38, 2003.

[22] Q. Z. Fang and L. Kong, "Core stability of vertex cover games," in 2007 *Proc. of Int. and Netw. Econ.*, pp. 482-490.

[23] Y. Yang and X. Li， "Towards a Snowdrift Game Optimization to Vertex Cover of Networks," *IEEE Trans. on Cybe.* vol. 43， no. 3， Jun. 2013.

[24] 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.

[25] C. P. Roca, J. A. Cuesta, and A. Sanchez, "Evolutionary game theory: Temporal and spatial effects beyond replicator dynamics," *Phys. Life Rev.*,vol. 6, no. 4, pp. 208–249, Dec. 2009.

[26] M. Perc and A. Szolnoki, "Coevolutionary games--A mini review," *BioSystems*, vol. 99, pp.109-125, 2010.

[27] W.-X. Wang, Y.-C. Lai, C. Grebogi, and J. Ye, "Network Reconstruction Based on Evolutionary-Game Data via Compressive Sensing," *Phys. Rev. X*, vol. 1, p. 021021, 2011.

[28] B. A. Huberman and N. S. Glance, "Evolutionary games and computer simulations," *Proc. Natl. Acad. Sci.* USA, vol. 90, p.7716-7718, 1993.

[29] B. Genest, H. Gimbert, A. Muscholl, and I. Walukiewicz, " Asynchronous Games over Tree Architectures," International Colloquium on Automata, Languages, and Programming, pp: 275-286, 2013.

[30] J. W. Weibull, Evolutionary Game Theory, Southern Economic Journal , vol. 98, no. 3, pp: 847-858, 1997.

[31] U. Benlic and J.-K. Hao, "Breakout Local Search for maximum clique problems," Computers & Operations Research, vol. 40, pp: 192–206. 2013.

[32] H. R, Lourenco, O, Martin, and T. Stützle, Iterated local search, hand book of meta- heuristics, Berlin, Heidelberg: Springer-Verlag; 2003.

[33] M. Hifi, "A Genetic Algorithm-Based Heuristic for Solving the Weighted Maximum Independent Set and Some Equivalent Problems," Journal of the Operational Research Society , vol. 48, no. 6, pp: 612-622, 1997.

[34] http://cs.hbg.psu.edu/txn131/clique.html.

[35] http:// neilsloane.com/doc/graphs.html.

[36] http://www.nlsde.buaa.edu.cn/_kexu/benchmarks/graph-benchmarks.htm.

[37] A. L. Barabási and R. Albert, "Emergence of scaling in random networks," Science, vol. 286, no. 5439, pp. 509-512, Oct. 1999.

[38] P. Erdös and A. Rényi, "On random graphs," Publ. Math. Debrecen, vol. 6, no. 290, pp. 290-297, 1959.

[39] D. J. Watts and S. H. Strogatz, "Collective dynamics of 'small-world' networks," Nature, vol. 393, no. 6684, pp. 440-442, Jun. 1998.

[40] A. Lancichinetti, S. Fortunato, and F. Radicchi, "Benchmark graphs for testing community detection algorithms," Phys. Rev. E, vol. 78, no. 4, p. 046110, 2008.

[41] B. An and V. Lesser, "Characterizing contract-based multiagent resource allocation in networks," IEEE Trans. Syst., Man, Cybern. B, Cybern., vol. 40, no. 3, pp. 575–586, Jun. 2010.

[42] B. Oommen and M. Hashem, "Modeling a student's behavior in a tutoriallike system using learning automata," IEEE Trans. Syst., Man, Cybern. B, Cybern., vol. 40, no. 2, pp. 481–492, Apr. 2010.

[43] J. F. Nash, "Equilibrium points in n-person games," Proc. Nat. Acad. Sci.U.S.A., vol. 36, no. 1, pp. 48–49, Jan. 1950.

[44] D. S. Johnson and M. Trick, editors. Cliques, Coloring and Satisfiability, volume 26 of DIMACS Series in Discrete Mathematics and Theoretical Computer Science. AMS, 1996.

[45] K. Xu. BHOSLIB: Benchmarks with hidden optimum solutions for graph problems, 2004. http://www.nlsde.buaa.edu.cn/~kexu/benchmarks/graph-benchmarks.htm. Last visited on March 15, 2008.

[46] A. Grosso, M. Locatelli, and W. Pullan. Simple ingredients leading to very efficient heuristics for the maximum clique problem. J. Heuristics, 14(6):587–612, 2008.

[47] S. Richter, M. Helmert, and C. Gretton. A stochastic local search approach to vertex cover. In Proceedings of the 30th German Conference on Artificial Intelligence (KI), pages 412–426, 2007.

[48] N. J. A. Sloane. Challenge problems: Independent sets in graphs, 2000. http://www. research.att.com/njas/doc/graphs.html. Last visited on March 15, 2008.

[49] S. Butenko, P. M. Pardalos, I. Sergienko, V. Shylo, and P. Stetsyuk. Finding maximum independent sets in graphs arising from coding theory. In Proceedings of the 2002 ACM Symposium on Applied Computing, pages 542–546, 2002.

[50] S. Butenko, P. M. Pardalos, I. Sergienko, V. Shylo, and P. Stetsyuk. Estimating the size of correcting codes using extremal graph problems. In C. Pearce, editor, Optimization: Structure and Applications. Springer, 2008.

[51] 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.

[52] C. H. Papadimitriou and K. Steiglitz, "Algorithms and Complexity," in Combinatorial Optimization, New York: Dover, ch. 15, sec. 6, 1998, pp. 360-363.

Jianshe Wu (M'07) 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.

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.