Local and Global Evaluation Functions for Computational Evolution

Jing Han*
The Santa Fe Institute,
Sante Fe, NM, 87501
and
Institute of Systems Science,
Academy of Mathematics and Systems Science,
Beijing, China, 100080

This paper compares computational evolution using local and global evaluation functions in the context of solving two classical combinatorial problems on graphs: the k-coloring and minimum coloring problems. It is shown that the essential difference between traditional algorithms using local search (such as simulated annealing) and distributed algorithms (such as the Alife&AER model) lies in the evaluation function. Simulated annealing uses global information to evaluate the whole system state, which is called the global evaluation function (GEF) method. The Alife&AER model uses local information to evaluate the state of a single agent, which is called the local evaluation function (LEF) method. Computer experiment results show that the LEF methods are comparable to GEF methods (simulated annealing and greedy), and in some problem instances the LEF beats GEF methods. We present the consistency theorem which shows that a Nash equilibrium of an LEF is identical to a local optimum of a GEF when they are "consistent." This theorem explains why some LEFs can lead the system to a global goal. Some rules for constructing a consistent LEF are proposed.

1. Introduction

The process of solving a combinatorial problem is often like an evolution of a system which is guided by evaluation functions. The purpose of this paper is to explore some of the differences between local and global evaluation functions in the context of graph coloring problems. We limit our discussion to the evolution of systems that are composed of homogeneous agents in limited state space. Note that these agents do not evolve their strategies during evolution.

Evolution is not pure random processing. There always exists an explicit or implicit evaluation function which directs the evolution. In

*Electronic mail address: hanjing@amss.ac.cn.

different contexts, the evaluation function is also called the objective function, penalty function, fitness function, cost function, or energy E. It has the form $f : S \in \Omega \to \mathbb{R}$, where Ω is the state (configuration) space. The evaluation function calculates how good a state of the whole system (or a single agent) is, then selection for the next step is based on its value. The evaluation function plays a key role in evolution and is one of the fundamental problems of evolution.

Generally speaking, the system state (configuration) related to the minimal evaluation value is the optimum of the system. For example, the Hamiltonian is used to evaluate the energy of a configuration of the spin glass system in statistical physics. So in this case the Hamiltonian is the evaluation function and the ground-state is related to the minimum of the Hamiltonian. In combinatorial problems, such as the SAT problem [1], the evaluation function is the total number of unsatisfied clauses of the current configuration. So the evaluation value for a solution of the SAT problem is zero. In artificial molecular evolution systems, the evaluation function is the fitness function, and evolution favors configurations that are related to the maximal fitness value.

A system may evolve in different paths if it is guided by different evaluation functions. But there always exists pairs of different evolution functions that lead to the same final state although the paths may be different. Therefore, for a specified system, people are able to construct different evaluation functions to reach the same goal state. As the main theme of this paper, we investigate the following two ways to construct evaluation functions.

1. **Global evaluation function (GEF).** GEF methods use global information to evaluate the whole system state. This is the traditional way of constructing the evaluation function and is widely used in many systems. The GEF embodies the idea of centralized control.

2. **Local evaluation function (LEF).** LEF methods use local (limited) information to evaluate a single agent's state and guide its evolution.

The idea of using local rules based on local information is one of the features of complex systems such as cellular automata [2], Boid [3], game of life [4], escape panic model [5], and sandpile model [6]. The private utility function in game theory [7] and the multiagent learning system COIN [8] are very close to the LEF idea except that the utility function is allowed to use global information.

It is a simple and natural idea to use a GEF to guide the evolution of a whole system to a global goal. But can we use an LEF to guide each agent so that the whole system attains the global goal in the end? In other words, if each agent makes its own decisions according to local information, can the agents get to the same system state reached by evolution guided by a GEF? Some emergent global complex behaviors from

local interactions have been found in natural and man-made systems [3–5, 9].

Most traditional algorithms used for solving combinatorial problems use a GEF, such as simulated annealing [10], and some new ones use an LEF, such as extremal optimization [11] and the Alife&AER model [12, 13]. Our earlier paper [14] investigated emergent intelligence in solving constraint satisfaction problems (CSPs, also called decision problems) [15] by modeling a CSP as a multiagent system and then using LEF methods to find the solution.

This paper studies the relationship and essential differences between local and global evaluation functions for the evolution of combinatorial problems, including decision and optimization problems. We use the coloring problem to demonstrate our ideas because it has two forms: the k-coloring problem, which is a decision problem; and the minimum coloring problem, which is an optimization problem. We found that some LEFs are consistent with some GEFs while some are not. So LEF and GEF methods differ in two layers: consistency and exploration. We also compare the performance of LEF and GEF methods using computer experiments on 36 benchmarks for the k-coloring and minimum coloring problems.

This paper is organized as follows. In section 2, computer algorithms for solving coloring problems are reviewed, and then a way to model a coloring problem as a multiagent system is described. Section 3 is about GEFs and LEFs. We first show how to construct a GEF and an LEF for solving the coloring problem (section 3.1), and then show their differences from aspects of consistency (section 3.2.1) and exploration heuristics (section 3.2.2). Computer experiments (section 3.3) show that the LEF (Alife&AER model) and GEF (simulated annealing and greedy) are good at solving different instances of the coloring problem. Conclusions are presented in section 4.

2. Coloring problems

The coloring problem is an important combinatorial problem. It is NP-complete and is useful in a variety of applications, such as time-tabling and scheduling, frequency assignment, and register allocation [15]. It is simple and can be mapped directly onto a model of an antiferromagnetic system [16]. Moreover, it has obvious network structure, making it a good example for studying network dynamics [17]. The coloring problem can be treated in either decision problem (CSPs) or optimization problem form.

Definition 1 (k-coloring problem) Given a graph $G = (V, e)$, where V is the set of n vertices and e is the set of edges, we need to color each vertex using a color from a set of k colors. A "conflict" occurs if a pair of

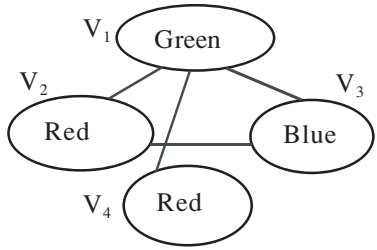


Figure 1. A solution for a 3-coloring problem.

linked vertices has the same color. The objective of the problem is either to find a coloring configuration with no conflicts for all vertices, or to prove that it is impossible to color the graph without any conflict using k colors. Therefore, the k-coloring problem is a decision problem. There are n variables for the n vertices where each variable is one of k colors. For each pair of nodes linked by an edge, there is a binary constraint between the corresponding variables that disallows identical assignments. Figure 1 is an example:

$$\begin{aligned} \text{Variable set } X &= \{X_1, X_2, X_3, X_4\}, \\ \text{Variable domain } D_1 &= D_2 = D_3 = D_4 = \{\text{green}, \text{red}, \text{blue}\}, \\ \text{Constraint set } R &= \{X_1 \neq X_2, X_1 \neq X_3, X_1 \neq X_4, X_2 \neq X_3\}. \end{aligned}$$

The k-coloring problem ($k \geq 3$) is NP-complete. It might be impossible to find an algorithm to solve the k-coloring problem in polynomial time.

Definition 2 (Minimum coloring problem) This definition is almost the same as the k-coloring problem except that k is unknown. It requires finding the minimum number of colors that can color the graph without any conflict. The minimum possible number of colors of G is called the *chromatic number* and denoted by $\chi(G)$. So the domain size of each variable is unknown. For the graph shown in Figure 1, $\chi(G) = 3$.

The minimum coloring problem is also NP-complete.

2.1 Algorithms

There are several basic general algorithms for solving combinatorial problems. Of course they can also be used to solve the coloring problem. Although we focus on the coloring problem, the purpose of this paper is to demonstrate the ideas of LEF and GEF methods for all combinatorial problems. What follows is a brief review of all general algorithms used for combinatorial problems, even though some of them might not have been applied to the coloring problem. The focus is only on the basic ones, so variants of the basic algorithms will not be discussed.

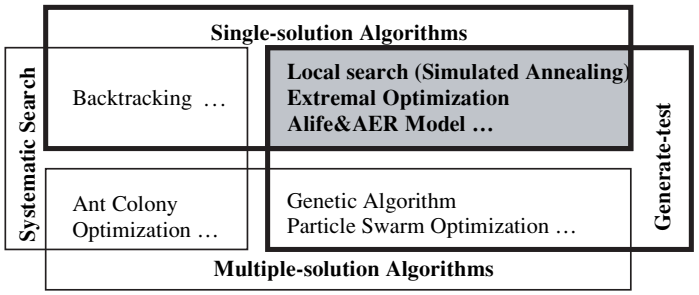


Figure 2. Classifications of basic algorithms for combinatorial problems. This paper focuses on GT-SS algorithms, the intersection of single-solution and generate-test (the gray area).

Figure 2 is our classification of current algorithms. According to how many (approximate) solutions the system is seeking concurrently, they can be divided into single-solution and multiple-solution algorithms. There are two styles of search: systematic search and generate-test. This paper will focus on GT-SS algorithms: the intersection of single-solution and the generate-test algorithms (the gray area in Figure 2), because the process of finding a solution using GT-SS algorithms can be regarded as the evolution of a multiagent system.

Single-solution algorithms (SS). The system is searching for only one configuration at a time. Some examples are backtracking [18], local search [10, 19–26], extremal optimization (EO) [18], and the Alife&AER model [12, 13].

Multiple-solution algorithms (MS). The system is concurrently searching for several configurations, which means that there are several copies of the system. Examples of this are genetic algorithms [27], ant colony optimization [9], and particle swarm optimization [28]. They are all population-based optimization paradigms. The system is initialized with a population of random configuration and searches for optima by updating generations based on the interaction between configurations.

Backtracking. The system assigns colors to vertices sequentially and then checks for conflicts in each colored vertex. If conflicts exist, it will go back to the most recently colored vertex and perform the process again. Many heuristics have been developed to improve performance. For the coloring problem, several important algorithms are SEQ [29], RLF [29], and DSATUR [30].

Generate-test (GT). This is a much bigger and more popular family than the backtracking paradigm. GT generates a possible configuration (colors all vertices) and then checks for conflicts (i.e., checks if it is a so-

lution). If it is not a solution, it will modify the current configuration and check it again until a solution is found. The simplest but inefficient way to generate a configuration is to select a value for each variable randomly. Smarter configuration generators have been proposed, such as hill climbing [25] or min-conflict [20]. They move to a new configuration with a better evaluation value chosen from the current configuration's neighborhood (see details in section 3.1) until a solution is found. This style of searching is called *local* (or *neighborhood*) search. For many large-scale combinatorial problems, local search always gives better results than the systematic backtracking paradigm. To avoid being trapped in the local-optima, it sometimes performs stop-and-restart, random walk [26], and Tabu search [23, 24, 31]. Simulated annealing [10] is a famous heuristic of local search inspired by the roughly analogous physical process of heating and then slowly cooling a substance to obtain a strong crystalline structure. People proposed simulated annealing schedules to solve the coloring problem and showed that it can dominate backtracking diagrams on certain types of graphs [29].

Some algorithms that use the complex systems ideas are genetic algorithms, particle swarm optimization, extremal optimization, and the Alife&AER model.

Extremal optimization and the Alife&AER model also generate a random initial configuration, and then change one variable's value in each improvement until a solution is found or the maximum number of trials is reached. Note that extremal optimization and the Alife&AER model improve the variable assignment based on LEFs, which is the main difference from simulated annealing.

Both genetic algorithms and particle swarm optimization have many different configurations at a time. Each configuration can be regarded as an agent (called a "chromosome" in genetic algorithms). The fitness function for each agent evaluates how good the configuration represented by the agent is, so the evolution of the system is based on a GEF.[1]

All of the given algorithms have their advantages and drawbacks, and no algorithm can beat all other algorithms on all combinatorial problems. Although GT algorithms always beat backtracking on large-scale problems, they are not complete algorithms. They cannot prove there is no solution for decision problems and sometimes miss solutions that do exist.

[1]Some implementations of particle swarm optimization can use either a GEF or an LEF. Since particle swarm optimization is not a single-solution algorithm, it is not discussed in this paper.

2.2 Multiagent modeling

We proposed modeling a CSP in the multiagent framework in [14]. Here we translate the coloring problem into the language of multiagent systems.

The concept of *agent* (denoted by a) is a computational entity. It is autonomous. It is able to act with other agents and get information from the system. It is driven by some objectives and has some behavioral strategies based on information it collects.

A multiagent system (MAS) is a computational system that consists of a set of agents $A = \{a_1, a_2, \ldots, a_n\}$, in which agents interact or work together to reach goals [32]. Agents in these systems may be homogeneous or heterogeneous, and may have common or different goals. MAS focus on the complexity of a group of agents, especially the local interactions among them (Figure 3).

It is straightforward to construct a MAS for a coloring problem:

- n vertices $\{V_1, V_2, \ldots, V_n\} \rightarrow n$ agents $\{a_1, a_2, \ldots, a_n\}$
- Possible colors of $V_i \rightarrow$ possible states of agent a_i
- Color of V_i is red $\Leftrightarrow X_i$ = red $\rightarrow$ state of agent a_i is a_i.state = red
- Link between two vertices $\rightarrow$ interaction between two agents
- Graph $\rightarrow$ interaction network in MAS
- Configuration $S = \langle X_1, X_2, \ldots, X_n \rangle \rightarrow$ a system state
- Solution $S_{\text{sol}} \rightarrow$ goal system state
- Process of finding a solution by GT-SS algorithms $\rightarrow$ evolution of MAS
- Evaluation function $\rightarrow$ evolution direction of MAS

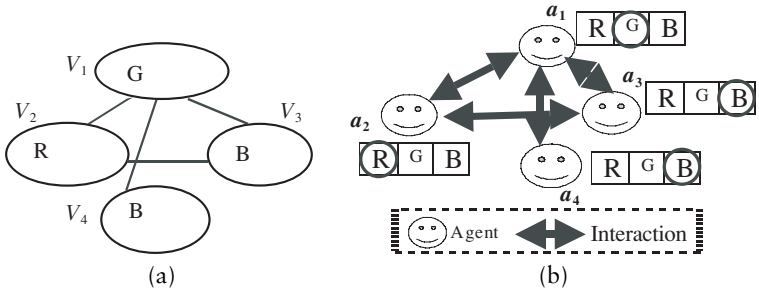
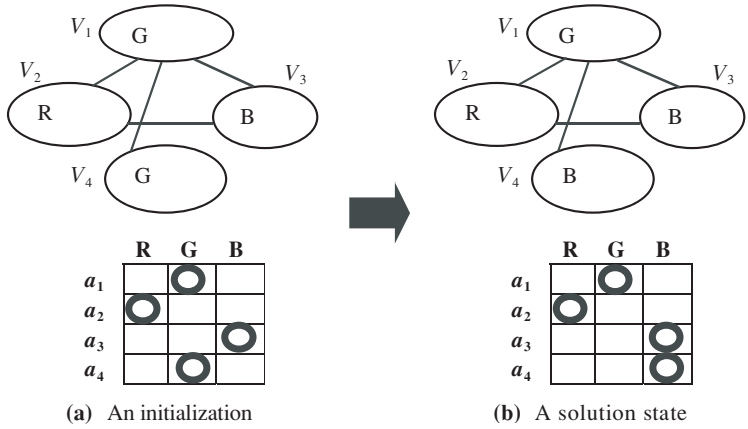


Figure 3. A MAS (right) for a 3-coloring problem (left). The table with three lattices beside each agent represents all possible states (R-red, G-green, B-blue), and the circle indicates the current agent state (color). The current configuration $S = \langle \text{G}, \text{R}, \text{B}, \text{B} \rangle$ is a solution.



(a) An initialization **(b)** A solution state

Figure 4. The MAS representation for a 3-coloring problem.

3. Global and local evaluation functions

3.1 Evolution

We now need to design a set of rules (or algorithm) to evolve configurations of the coloring problems. The core of the algorithm is how the whole system (configuration) changes from one state to another state, that is, how variables change their assigned values.

For example, if the system is initialized as shown in Figure 4, $S = \langle \mathrm{G}, \mathrm{R}, \mathrm{B}, \mathrm{G} \rangle$ is obviously not a solution. There is a conflict between V_1 and V_4 which are linked to each other and have the same color. Thus, the system needs to evolve to reach a solution state, for example, $S = \langle \mathrm{G}, \mathrm{R}, \mathrm{B}, \mathrm{B} \rangle$ (see Figure 4(b)).

How does the system evolve to the solution state? As mentioned earlier, GT-SS algorithms (e.g., simulated annealing, extremal optimization, and the Alife&AER model) are schedules for evolution of the MAS. In the language of MAS, those algorithms all share the same following framework.

1. Initialization ($t = 0$): each agent picks a random state, get $S(0)$.
2. For each time step (t): according to some schedule and heuristics, one or several agents change their state, so that $S(t + 1) = \mathrm{Evolution}(S(t))$; $t \leftarrow t + 1$.
3. Repeat step 2 until a solution is found or the maximum number of trials is reached.
4. Output the current state $S(t)$ as the (approximate) solution.

The essential difference between the local and global functions lies in the process of Evolution(S) in step 2. GEF methods are described in section 3.1.1 and LEF methods follow in section 3.1.2.

3.1.1 Traditional methods: Global evaluation functions

Evolution(S) of step 2 for GEF methods can be described as:

- Compute the GEF values of all (or some, or one) neighbors of the current system state $S(t)$, and then select and return a neighboring state $S(t)'$ to update the system.

A *neighborhood* in GEF methods is the search scope for the current system state. For example, we can define the neighborhood structure of a configuration $S \in \Omega$ based on the concept of *Hamming distance*:

$$N_S^d = \{S' \mid \text{Hamming-distance}(S, S') = d\},$$

where

$$\text{Hamming-distance}(S, S') = \sum_i (1 - \delta(X_i, X_i')),$$

with δ denoting the *Kronecker* function $\delta(x, y) = 1$ if $x = y$ and otherwise $\delta(x, y) = 0$.

For example, if $d = 1$ in the k-coloring problem, the GEF search space is N_S^1 and there are $n(k-1)$ neighbors of each system state. Figure 4(b) is a neighbor of Figure 4(a). In Figure 6, (b) and (c) are neighbors of (a). If d is larger, the neighborhood size is bigger. This will enlarge the search scope and increase the cost of searching the neighborhood. But there will probably be greater improvement in each time step. So the neighborhood structure will affect the efficiency of the algorithm [33]. In the following discussions, we always use the neighborhood structure N_S^1 since the neighborhood structure is not in the scope of this paper. Note that simulated annealing just randomly generates a neighbor and accepts it if it is a better state, or with a specified probability to accept a worse state. Here "better" and "worse" are defined by the evaluation value.

A GEF evaluates how good the current system state S is. In simulated annealing, the GEF is the energy or cost function of the current system state and guides the search. There are a variety of evaluation functions for a specified problem, and a suitable choice can get good performance.

A GEF for the k-coloring problem. One naïve GEF for the k-coloring problem is to count the number of conflicts in the current configuration. For simplicity, equation (1) counts twice the number of conflicts:

$$E_{\text{GEF-k}}(S) = \sum_{i=1}^{n} \sum_{X_j \in S_{-i}} \delta(X_i, X_j) \tag{1}$$

where $S_{-i} = \langle X_j \mid \langle i, j \rangle \in e \rangle$ denotes the set of variables that link to V_i.

Equation (1) is a GEF because it considers the whole system including all agents. For the system state shown in Figure 4(a), the evaluation value is $1 \times 2 = 2$. Figure 4(b) shows a solution state, so $E_{\text{GEF}}(S_{\text{sol}}) = 0$.

Minimizing the evaluation value is the goal of the solving process, although a state with a better evaluation value does not always mean that it is closer to the solution state.

A GEF for the minimum coloring problem. There are several evaluation functions [29, 34, 35] for finding the chromatic number $\chi(G)$. We modified the version given by Johnson [29] to be:

$$E_{\text{GEF-O}}(S) = \underbrace{-\sum_{i=1}^{m} |C_i|^2}_{\text{(a)}} + \underbrace{n \sum_{i=1}^{n} \sum_{X_j \in S_{-i}} \delta(X_i, X_j)}_{\text{(b)}} \tag{2}$$

where m is the current total number of colors being used. The set C_i includes variables that are colored by the i-th color. We assign each color s an index number Index(s). Say red is the first color, Index(red) = 1; green is the second color, Index(green) = 2; blue is the third color, Index(blue) = 3; and so on. So

$$C_i = \{X_j \,|\, \text{Index}(X_j) = i,\ j \in 1 \ldots n\}.$$

Part (b) of equation (2) counts twice the number of conflicts in the current configuration. Part (a) favors using fewer colors. n in part (b) is a weight value to balance parts (a) and (b), which makes all the local optima in equation (2) correspond to a nonconflicting solution, that is, a feasible solution.

For example, using equation (2) again, in Figure 5(a): $C_1 = \{X_2, X_4\}$, $C_2 = \{\emptyset\}$, and $C_3 = \{X_1, X_3\}$, so $E_{\text{GEF-O}}(S_a) = -(2^2 + 2^2) + 4 \times 1 \times 2 = 0$. We can get the evaluation value of its two neighbors: $E_{\text{GEF-O}}(S_b) = 2$ and $E_{\text{GEF-O}}(S_c) = -6$. $E_{\text{GEF-O}}(S_b) > E_{\text{GEF-O}}(S_a)$ shows that S_b is a worse

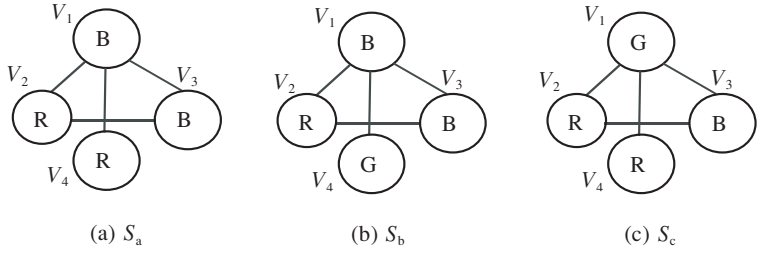


Figure 5. An example of using GEFs for a coloring problem. S_a is the initial state. S_b and S_c are two neighbors of S_a. As a 3-coloring problem, based on equation (1), $E_{\text{GEF-k}}(S_a) = 2$, $E_{\text{GEF-k}}(S_b) = 2$, $E_{\text{GEF-k}}(S_c) = 0$. So S_c is the solution. As a minimum coloring problem based on equation (2), $E_{\text{GEF-O}}(S_a) = 0$, $E_{\text{GEF-O}}(S_b) = 2$, $E_{\text{GEF-O}}(S_c) = -6$, so S_c is the best one (also an optimal solution).

state than S_a, because it uses one more color, but it still has one conflict as does S_a. For $E_{\text{GEF-O}}(S_c) < E_{\text{GEF-O}}(S_a)$, S_c is a better state because it can remove the conflict in S_a although it uses one more color. Actually S_c is an optimal solution.

So we can see that the GEF methods have two main features: (1) they evaluate the whole system state, not a single agent's state; (2) they use all information of the system, equations (1) and (2) consider all agents' states. Therefore, centralized control in GEF methods is obvious: in each step, the system checks all (or one) neighbors and selects the one that has the best evaluation value as the new system state. No agents here are autonomous. They all obey the choice of the centralized controller.

3.1.2 Local evaluation function methods

Evolution(S) of step 2 using LEF methods can be described as:

- For each agent a_i, do the following sequentially:[2] Compute the evaluation values of all/some states by a_i's LEF, and then select a state for a_i to update itself.

After obtaining the evaluation values for all states of an agent, it will update its state using some strategies, such as least-move, better-move, and random-move (random walk). Least-move means the agent will select the best state and change to that state, and random-move means the agent will change its state randomly (for details, see [13]). These strategies are part of the exploration heuristics (see section 3.2.2).

An LEF for the k-coloring problem. Like the GEF, there are a variety of functions which can serve as the LEF for a problem. Equation (3) is one possible LEF for each agent a_i in the k-coloring problem [13]. It calculates how many conflicts each state (s) of agent a_i receives based on the current assignments of a_i's linked agents. Note that if for all $i \in 1 \ldots n$ we have $E^i_{\text{LEF-k}}(S_{-i}, X_i) = 0$, then S is a solution (Figure 6(c)):

$$E^i_{\text{LEF-k}}(S_{-i}, s) = \sum_{X_j \in S_{-i}} \delta(X_j, s). \tag{3}$$

For example, in Figure 5(a), agent a_1 links to a_2, a_3, and a_4, so we get:

$$E^1_{\text{LEF-k}}(S_{a,-1}, \text{R}) = (\delta(X_2, \text{R}) + \delta(X_3, \text{R}) + \delta(X_4, \text{R})) = 2$$
$$E^1_{\text{LEF-k}}(S_{a,-1}, \text{G}) = (\delta(X_2, \text{G}) + \delta(X_3, \text{G}) + \delta(X_4, \text{G})) = 0$$
$$E^1_{\text{LEF-k}}(S_{a,-1}, \text{B}) = (\delta(X_2, \text{B}) + \delta(X_3, \text{B}) + \delta(X_4, \text{B})) = 1.$$

[2] Or simultaneously in some cases, as in cellular automata. This is one of our future projects.

	R	G	B
a_1	2	0	1
a_2	0	0	2
a_3	1	0	1
a_4	0	0	1

(a) Evaluation values in S_a

	R	G	B
a_1	1	1	1
a_2	0	0	2
a_3	1	0	1
a_4	0	0	1

(b) Evaluation values in S_b

	R	G	B
a_1	2	0	1
a_2	0	1	1
a_3	1	1	0
a_4	0	1	0

(c) Evaluation values in S_c

Figure 6. An example of using an LEF for the 3-coloring problem shown in Figure 5. Numbers on each row of the lattices represent the evaluation values based on equation (3) for an agent's three possible states (colors): R, G, or B. Circles refer to the current state of the agent. (c) shows that S_c is a solution state because all agents are in nonconflicting states. From S_a, if a_1 moves first, it will most probably change its state from B to G and get to S_c. If a_4 moves first, it will stay, or with small probability, it will change to G and get to S_b.

Since the evaluation value of the current state ($X_1 = B$) is higher than state G, a_1 will most probably change to state G.

Agent a_4 only links to a_1, so it only considers a_1's state:

$$E^4_{\text{LEF-k}}(S_{a,-4}, \text{R}) = \delta(X_1, \text{R}) = 0$$
$$E^4_{\text{LEF-k}}(S_{a,-4}, \text{G}) = \delta(X_1, \text{G}) = 0$$
$$E^4_{\text{LEF-k}}(S_{a,-4}, \text{B}) = \delta(X_1, \text{B}) = 1.$$

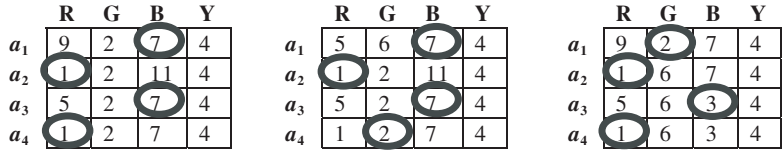
Similarly, $E^2_{\text{LEF-k}}$ for agent a_2 only considers a_1 and a_3. $E^3_{\text{LEF-k}}$ for agent a_3 only considers a_1 and a_2. Continuing this way, we can get evaluation values (Figure 6) for all states of each agent.

An LEF for the minimum coloring problem. If each agent only knows its linked agents, how can they use their limited knowledge to find the minimal colors for a graph? Each agent has to make an approximation of the global requirement. Let us try the following LEF for each agent a_i:

$$E^i_{\text{LEF-O}}(S_{-i}, s) = \underbrace{\text{Index}(s)}_{\text{(a)}} + \underbrace{n \sum_{X_j \in S_{-i}} \delta(X_j, s)}_{\text{(b)}} \tag{4}$$

where Index(s) is described in section 3.1.1 and returns the index number of color s.

Part (b) of equation (4) counts how many conflicts a_i will receive from its linked agents if a_i takes the color s. Part (a) of equation (4) favors using fewer colors by putting pressure on each agent to try to get a color with a smaller index number. n in part (b) is a weight value to balance parts (a) and (b), which makes all the local optima in equation (4) correspond to a nonconflicting state. Note that if for



(a) Evaluation values in S_a (b) Evaluation values in S_b (c) Evaluation values in S_c

Figure 7. An example of using an LEF for the minimum coloring problem shown in Figure 5. Numbers on each row of the lattices represent the evaluation values based on equation (4) for all possible states (colors) of an agent. Circles refer to the current state of the agent. (c) shows that S_c is a feasible solution state because all agents are in nonconflicting states and it is also an optimal solution. From S_a, if a_1 moves first, it will most probably change its state from B to G and get to S_c; if a_4 moves first, it will stay, or with small probability it will change to G and get to S_b.

all $i \in n$ we have $E^i_{\text{LEF-O}}(S_{-i}, X_i) \leq n$, $S = \langle X_1, X_2, \ldots, X_n \rangle$ is a feasible solution, but not always the optimal solution.

We can evaluate each state of a_1 in Figure 7(a):

$$\begin{aligned} E^1_{\text{LEF-O}}(S_{a,-1}, \text{R}) &= \text{Index(R)} + n \times (\delta(X_2, \text{R}) + \delta(X_3, \text{R}) + \delta(X_4, \text{R})) \\ &= 1 + 4 \times 2 = 9 \\ E^1_{\text{LEF-O}}(S_{a,-1}, \text{G}) &= \text{Index(G)} + n \times (\delta(X_2, \text{G}) + \delta(X_3, \text{G}) + \delta(X_4, \text{G})) \\ &= 2 + 4 \times 0 = 2 \\ E^1_{\text{LEF-O}}(S_{a,-1}, \text{B}) &= \text{Index(B)} + n \times (\delta(X_2, \text{B}) + \delta(X_3, \text{B}) + \delta(X_4, \text{B})) \\ &= 3 + 4 \times 1 = 7. \end{aligned}$$

Since the evaluation value of the current state ($X_1 = B$) is higher than state G, a_1 will most probably perform a least-move to change from B to G, and get to an optimal solution S_c. Or, with lower probability a_1 will perform a random-move to change to state Y and get to another feasible solution (but not an optimal one, since it uses four colors).

So we can see that the LEF methods have two main features: (1) they evaluate the state of a single agent, not the whole system; (2) they use local information of the system. Equations (3) and (4) only consider its linked agents' states (i.e., only knows the information of its related subgraph). Decentralized control in LEF methods is also obvious: in each step, the system dispatches all agents sequentially (e.g., in the extremal optimization algorithm, the system dispatches the agent who is in the worst state according to the LEF). All agents are autonomous and decide their behaviors based on the LEF that maximizes their own profit. Therefore, if the whole system can achieve a global solution based on all selfish agents which are using the LEF, we call this *emergence* and the system self-organizes towards a global goal.

3.2 Essential differences between local and global evaluation functions

3.2.1 Consistency and inconsistency

In section 3.1 we studied the local and global evaluation functions using the coloring problem. Now we know the following two aspects should be considered while constructing an evaluation function.

1. What object is being evaluated? With a GEF, the whole system is considered, while the LEF considers a single agent.

2. How much information is used in the evaluation? The GEF considers the whole system state, while the LEF only considers the agent's linked agents.

This fits the concepts we mentioned in the Introduction. But what if the evaluation functions consider the whole system and only use local information? What is the relationship between the local and global evaluation functions?

Consistency between LEF and GEF. In the following, we use

$$S' = \text{replace}(S, i, s, s'), s, s' \in D_i$$

to denote that S' is achieved by replacing the state s of X_i in S to be s'. So, all variables share the same state in S and S' except X_i. So S' belongs to N_S^1.

Definition 3. An LEF is *consistent* with a GEF if it is true for $\forall S \in \Omega$ and $\forall S' \in N_S^1$ (i.e., $S' = \text{replace}(S, i, s, s')$) that

$$\text{sgn}(E_{\text{GEF}}(S') - E_{\text{GEF}}(S)) = \text{sgn}(E^i_{\text{LEF}}(S_{-i}, s') - E^i_{\text{LEF}}(S_{-i}, s)), \qquad (5)$$

where $\text{sgn}(x)$ is defined as: $\text{sgn}(x) = 1$ when $x > 0$, $\text{sgn}(x) = -1$ when $x < 0$, and $\text{sgn}(0) = 0$ when $x = 0$.

For convenience, sometimes we just simply say $\text{sgn}(\Delta E_{\text{GEF}}) = \text{sgn}(\Delta E^i_{\text{LEF}})$ for equation (5).

Definition 4. An LEF is *restrict-consistent* with a GEF if
(1) the LEF is consistent with the GEF and
(2) $\exists \alpha \in R^+$ such that it is true for all $S \in \Omega$ that

$$E_{\text{GEF}}(S) = \alpha \sum_{i=1}^{n} D^i_{\text{LEF}}(S_{-i}, X_i). \qquad (6)$$

So restrict-consistent is the subset concept of consistent.

It is easy to prove that LEF equation (3) is restrict-consistent with GEF equation (1) in the k-coloring problem. For example, in the 3-coloring problem shown in Figure 5, agent a_1 uses LEF equation (3) to change

to G which can decrease its valuation value from 1 to 0. This change is also a good change for the whole system, because it decreases the evaluation value of the whole system based on GEF equation (1) (from $E_{\text{GEF-k}}(S_a) = 2$ to $E_{\text{GEF-k}}(S_c) = 0$). Suppose we use simulated annealing. The current system state is S and we randomly generate a neighbor $S' = \text{replace}(S, j, s, s')$. Then we need to calculate $\Delta = E_{\text{GEF-k}}(S') - E_{\text{GEF-k}}(S)$ to decide whether S' is a better system state or not. In the consistent case, it is not necessary to consider all of the agents' states and $\Delta E_{\text{GEF-k}}$ can be calculated by $\Delta E^j_{\text{LEF-k}}$:

$$E_{\text{GEF-k}}(S') - E_{\text{GEF-k}}(S) = 2\Delta E^j_{\text{LEF-k}}.$$

What does *consistency* mean? A good agent decision based on the LEF that can decrease the value of $E^j_{\text{LEF-k}}$ is also a good decision for the whole system based on the GEF that can decrease the value of $E_{\text{GEF-k}}$. In the k-coloring problem, while all agents get to a state that is evaluated by equation (3) of zero, the evaluation value of equation (1) for the whole system is also zero, which means it is a solution. This makes it easier for us to understand why the LEF can guide agents to a global solution. When the LEF is consistent with the GEF, they both indicate the same equilibrium points. For a certain configuration S, if it is the local optimum of the GEF, then it is the Nash equilibrium [36] of the LEF, and *vice versa*.

Definition 5. If $E_{\text{GEF}}(S') \geq E_{\text{GEF}}(S)$ is true for $S \in \Omega$ and $\forall S' \in N^1_S$, then S is called the *local optimum* of the GEF using a N^1_S neighborhood structure in problem P.

The set of all local optima is denoted as $L(P, E_{\text{GEF}})$.

Optimal solutions of optimization problems and feasible solutions in decision problems belong to $L(P)$ and have the smallest evaluation value.

Definition 6. A *Nash equilibrium* of an LEF in a problem P is defined as $S = \langle s_1, s_2, \ldots, s_n \rangle$, $S \in \Omega$, such that

$$E^i_{\text{LEF}}(S_{-i}, s_i) \leq E^i_{\text{LEF}}(S_{-i}, s'_i) \text{ for } \forall s'_i \in D_i$$

holds for all $i = 1, 2, \ldots, n$.

The set of all Nash equilibria of the P and the LEF is denoted as $N(P, E_{\text{LEF}})$.

Theorem 1 (Consistency) Given a problem P, a GEF, and an LEF: if the LEF is consistent with the GEF (i.e., equation (5) holds between them) then the following is true:

$$(\forall S \in \Omega)\ S \in L(P, E_{\text{GEF}}) \Leftrightarrow S \in N(P, E_{\text{LEF}}). \tag{7}$$

It follows that $L(P, E_{\text{GEF}}) = N(P, E_{\text{LEF}})$.

Proof. If $S \in L(P, E_{\text{GEF}})$, S is a local optimum, that is, $E_{\text{GEF}}(S') \geq E_{\text{GEF}}(S)$ is true for $\forall S' \in N_S^1$

$$\Rightarrow \forall i = 1, 2, \ldots, n, \forall x \in D_i, E_{\text{GEF}}(\text{replace}(S, i, s_i, x)) - E_{\text{GEF}}(S) \geq 0.$$

Because $\text{sgn}(E_{\text{GEF}}(\text{replace}(S, i, s_i, x)) - E_{\text{GEF}}(S)) = \text{sgn}(E^i_{\text{LEF}}(S_{-i}, x) - E^i_{\text{LEF}}(S_{-i}, s))$

$$\Rightarrow \forall i = 1, 2, \ldots, n, \forall x \in D_i, E^i_{\text{LEF}}(S_{-i}, x) - E^i_{\text{LEF}}(S_{-i}, s) \geq 0$$

$$\Rightarrow \forall i = 1, 2, \ldots, n, \forall x \in D_i, E^i_{\text{LEF}}(S_{-i}, x) \geq E^i_{\text{LEF}}(S_{-i}, s)$$

$$\Rightarrow S \text{ is a Nash equilibrium}, S \in N(P, E_{\text{LEF}}).$$

If $S \in N(P, E_{\text{LEF}})$ and S is a Nash equilibrium, then we have

$$\forall i = 1, 2, \ldots, n, \forall x \in D_i, E^i_{\text{LEF}}(S_{-i}, x) \geq E^i_{\text{LEF}}(S_{-i}, s).$$

Because $\text{sgn}(E^i_{\text{LEF}}(S_{-i}, x) - E^i_{\text{LEF}}(S_{-i}, s)) = \text{sgn}(E_{\text{GEF}}(\text{replace}(S, i, s_i, x)) - E_{\text{GEF}}(S))$

$$\Rightarrow \forall i = 1, 2, \ldots, n, \forall x \in D_i, E_{\text{GEF}}(\text{replace}(S, i, s, x)) - E_{\text{GEF}}(S) \geq 0$$
$$\Rightarrow \forall S' \in N_S^1, E_{\text{GEF}}(S') \geq E_{\text{GEF}}(S) \rightarrow S \text{ is a local optimum,}$$
$$S \in L(P, E_{\text{GEF}}). \blacksquare$$

Inference 1. Given a problem P, a GEF, and an LEF: if the LEF is consistent with the GEF, S is a solution and S is a local optimum of the GEF, then S is also a Nash equilibrium of the LEF.

Inference 1 helps us to understand why some LEF can solve problems that the GEF can solve: because their equilibrium points are the same! They have the same phase transition, solution numbers, and structure according to spin glass theory [37]. The main difference between a consistent pair of LEF and GEF lies in their exploration heuristics, such as dispatch methods and strategies of agents. We discuss this in section 3.2.2.

So can we construct any other LEF or GEF that can make an essential difference? Is there any case for an LEF that is not consistent with a GEF?

Inconsistency between local and global evaluation functions. The LEF equation (4) is designed for minimum coloring problems, but it can also be used to solve the k-coloring problem (the experimental result in section 3.3.1 shows that equation (4) works for the k-coloring problem). LEF equation (4) and GEF equation (1) are not consistent with each other. Suppose the current configuration is S and its neighboring configuration $S' = \text{replace}(S, i, s, s')$, then

$$\Delta E^i_{\text{LEF-k}} = \frac{n}{2} \Delta E_{\text{GEF-k}} + \text{Index}(s') - \text{Index}(s). \tag{8}$$

ΔE_{GEF} \ ΔE_{LEF}	>0	=0	<0
>0	Y	N	N
=0	Y	Y	Y
<0	N	N	Y

(a)

ΔE_{GEF} \ ΔE_{LEF}	>0	=0	<0
>0	Y	Y	N
=0	N	Y	N
<0	N	Y	Y

(b)

Table 1. Weak-inconsistency relations between LEF and GEF. (a) LEF is weak-inconsistent with GEF and (b) GEF is weak-inconsistent with LEF. The tables show all possible relation combinations of ΔE_{GEF} and ΔE_{LEF} for any $S \in \Omega$ and any $S' \in N_s^1$. Y means allowed in the weak-inconsistency, N means not allowed.

If $\Delta E_{\mathrm{GEF}} \neq 0$: Because Index($s$) $\leq n$ for all s, so $|n/2\Delta E_{\mathrm{GEF}}| > |\mathrm{Index}(s') - \mathrm{Index}(s)|$ holds for all s' and s, we will get $\mathrm{sgn}(\Delta E_{\mathrm{GEF}}) = \mathrm{sgn}(\Delta E^i_{\mathrm{LEF\text{-}k}})$.

If $\Delta E_{\mathrm{GEF}} = 0$: $\Delta E^i_{\mathrm{LEF\text{-}k}}$ can be larger than, equal to, or smaller than zero depending on Index(s') – Index(s). There exists S and $S' \in N_S^1$ such that $\mathrm{sgn}(\Delta E_{\mathrm{GEF}}) \neq \mathrm{sgn}(\Delta E^i_{\mathrm{LEF\text{-}k}})$.

So equation (5) is true only when $\Delta E_{\mathrm{GEF}} \neq 0$. LEF equation (4) is inconsistent with GEF equation (1) in the k-coloring problem.

Definition 7. An LEF is *inconsistent* with a GEF if the LEF is not consistent with the GEF.

Definition 8. An LEF is *weak-inconsistent* with a GEF if $\forall S \in \Omega$ and $\forall S' \in N_S^1$, equation (5) is not true only when $\Delta E_{\mathrm{GEF}} \neq 0$. A GEF is *weak-inconsistent* with an LEF if $\forall S \in \Omega$ and $\forall S' \in N_S^1$, equation (5) is not true only when $\Delta E^i_{\mathrm{LEF}} \neq 0$ (see Table 1).

So weak-inconsistency is an asymmetrical relation and is a subset concept of inconsistency. LEF equation (4) is weak-inconsistent with GEF equation (1) in the k-coloring problem.

Inference 2. Given a problem P, a GEF, and an LEF we have $N(P, E_{\mathrm{LEF}}) \subset L(P, E_{\mathrm{GEF}})$ if the LEF is weak-inconsistent with the GEF. We have $L(P, E_{\mathrm{GEF}}) \subset N(P, E_{\mathrm{LEF}})$ if the GEF is weak-inconsistent with the LEF.

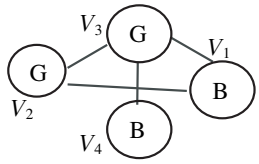
GEF equation (2) and LEF equation (4) in the minimum coloring problem is an example of inconsistency but not weak-inconsistency. The minimal number of colors $\chi(G)$ is a global variable. We can see that the linear relationship between equations (2) and (4) only exists between part (b), not between part (a). It is easy to prove the inconsistency: suppose a variable X_k changes from the l-th color to the h-th color. If this does not cause a change in conflict numbers (i.e., conflicts between

X_k and its linked vertices keep the same number, part (b) remains the same), then

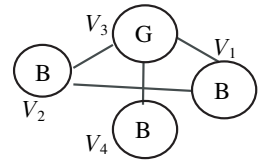
$$\Delta E_{\text{GEF-O}} = 2(|C_h| - |C_l| - 1) \text{ and } \Delta E^i_{\text{LEF-O}} = h - l.$$

If $h > l$ and $|C_l| > |C_h| + 1$, we will get $\Delta E_{\text{GEF-O}} < 0$ and $\Delta E^i_{\text{LEF-O}} > 0$. If $h < l$ and $|C_l| < |C_h| + 1$, we will get $\Delta E_{\text{GEF-O}} > 0$ and $\Delta E^i_{\text{LEF-O}} < 0$. In the LEF, it is more likely that $h < l$ because the agent favors colors which have smaller index numbers according to equation (4). So when $h < l$ and C_h has no fewer vertices than C_l, a good choice for an agent (from color l to color h) based on the LEF will be a bad one for the whole system based on the GEF. This occurs because part (a) of equation (2) favors the unbalanced distribution of colors.

The minimum coloring problem example in Figure 8 shows that, when using an LEF, a_2 will not change from G to B, while if using a GEF it will. This indicates that LEF equation (4) and GEF equation (2) will lead the system to different states.



(a) S_a



(b) S_b

	R	G	B
a_1	1	14	3
a_2	1	**8**	9
a_3	1	8	15
a_4	1	8	3

(c) LEF values for S_a

	R	G	B
a_1	1	8	9
a_2	1	8	**9**
a_3	1	2	21
a_4	1	8	3

(d) LEF values for S_b

GEF for (a) and (b):
$E_{\text{GEF-O}}(S_a)=4$
$E_{\text{GEF-O}}(S_b)=2$
$\Delta E_{\text{GEF-O}} = -2 < 0$

(e) GEF values for S_a, S_b

Figure 8. Inconsistency between LEF equation (4) and GEF equation (2) in a minimum coloring problem. (a) and (b) show two possible configurations which are neighbors and differ only in agent a_2. (c) and (d) separately show LEF values for all states of each agent of S_a and S_b, so $E^2_{\text{LEF-O}}(S_b, \text{G}) - E^2_{\text{LEF-O}}(S_a, \text{B}) = 9 - 8 = 1 > 0$. (e) shows the GEF values for S_a and S_b, $E_{\text{GEF-O}}(S_b) - E_{\text{GEF-O}}(S_b) = 2 - 4 = -2 < 0$. So equations (4) and (2) are inconsistent with each other because the LEF prefers S_a while the GEF prefers S_b.

X_i \ X_j	s_l	s_h
s_l	x / x	r / r
s_h	z / z	y / y

Table 2. The symmetrical penalty of two linked nodes i and j for two colors s_l and s_h. Upper values are of $g_{ij}(X_i, X_j)$, and lower values are of $g_{ji}(X_j, X_i)$. It has the attribute of $g_{ij}(X_i, X_j) = g_{ji}(X_j, X_i)$.

Consistency between $E_{\mathbf{LEF}}$ and $E_{\mathbf{GEF}} = \sum E_{\mathbf{LEF}}$. Inconsistency means the individual's profit (defined by an LEF) will (sometimes) conflict with the profit (defined by a GEF) of the whole system. Does this mean that if an LEF is inconsistent with a GEF, the LEF is also inconsistent with other GEFs? Actually, we can find a consistent GEF for equation (4) by simply defining it as the sum of the entire formula:

$$E_{\text{GEF-O}}(S) = \sum_{i=1}^{n} \text{Index}(X_i) + n \sum_{i=1}^{n} \sum_{X_j \in S_{-i}} \delta(X_i, X_j). \tag{9}$$

It is easy to prove that LEF equation (4) is consistent with GEF equation (9).[3]

Definition 9. If an LEF E_{LEF} is consistent with the GEF $E_{\text{GEF}} = \sum E_{\text{LEF}}$, E_{LEF} is called a *consistent* LEF, otherwise it is called an *inconsistent* LEF.

Is any LEF E_{LEF} always consistent with $E_{\text{GEF}} = \sum E_{\text{LEF}}$? Are all LEFs consistent? Our answer to both is no. For example, in the 2-coloring problem for a graph that only consists of two linked nodes i and j, we have a total of four coloring configurations. If we define the evaluation value of the LEF of each node in each configuration as in Table 2, we will find an inconsistency between the LEF and GEF $= \sum E_{\text{LEF}}$.

We want to give some rules for constructing a consistent LEF under the following limitations.

1. We limit our discussion to binary constraint problems with the same domains for all variables. This means all constraints are only between two variables, and $D_1 = D_2 = \cdots = D_n = D$. The coloring problem is a binary-constraint problem and all domains are the same.

[3]We can also say that GEF equation (9) is inconsistent with GEF equation (2) if we extend the consistency concept to be: evaluation function E_1 is consistent with evaluation function E_2 if for any two neighboring configurations S and S', $\text{sgn}(\Delta E_1) = \text{sgn}(\Delta E_2)$.

2. We only discuss LEFs which have the form

$$E^i_{\mathrm{LEF}}(S_{-i}, s) = f(s) + \beta \sum_{X_j \in S_{-i}} g_{ij}(s, X_j),$$
$$\beta > 0 \text{ and } \beta > \max_{s',s}\{|f(s') - f(s)|\}. \tag{10}$$

So the form of the GEF $E_{\mathrm{GEF}} = \sum E_{\mathrm{LEF}}$ is:

$$E_{\mathrm{GEF}}(S) = \sum_i f(X_i) + \beta \sum_i \sum_{X_j \in S_{-i}} g_{ij}(X_i, X_j). \tag{11}$$

$f(s)$ indicates the preference for color (value) s of the node (agent or variable). The second part is about constraints between two linked nodes, so $g_{ij}(X_i, X_j)$: $D_i \times D_j \to \mathbb{R}$ is the static penalty function of X_i incurred by the constraint between X_i and X_j. β balances these two parts. Equation (11) has the same form as a Hamiltonian in spin glasses: the first part relates to the external field, and the second part relates to the interaction between two spins. Therefore, the form we defined here can cover a big class of LEFs for combinatorial problems, and the related GEFs have the same form as the Hamiltonian in spin glass models. Equations (3) and (4) are of the form given by equation (10); equations (1) and (9) are of the form of equation (11).

3. Only one form of static penalty function $g_{ij}(X_i, X_j)$ is considered here which satisfies $g_{ij}(X_i, X_j) = g_{ji}(X_j, X_i)$. We call this $g_{ij}(X_i, X_j)$ a *symmetrical penalty* (see Table 2): in equations (1) through (4) and (9), $g_{ij}(X_i, X_j) = \delta(X_i, X_j)$ which is a symmetrical penalty with $x = y = 1$ and $r = z = 0$. Obviously, the penalty function shown in Figure 9(a) is not a symmetrical penalty.

Given these limitations, an LEF can be consistent, as stated in Theorem 2.

Theorem 2. If $\forall \langle i, j \rangle \in e$, $g_{ij}(X_i, X_j) = g_{ji}(X_j, X_i)$, then an E_{LEF} which has the same form as equation (10) is consistent with $E_{\mathrm{GEF}} = \sum E_{\mathrm{LEF}}$. That is, E_{LEF} is a consistent LEF.

Proof. For $\forall S = \langle X_1, X_2, \ldots, X_n \rangle \in \Omega$ and $\forall S' = \langle X'_1, X'_2, \ldots, X'_n \rangle \in N^1_S$,

$$\begin{aligned}
&\mathrm{sgn}(E_{\mathrm{GEF}}(S') - E_{\mathrm{GEF}}(S)) \\
&\quad = \mathrm{sgn}\left(\sum_k f(X'_k) + \beta \sum_k \sum_{X'_j \in S'_{-k}} g_{kj}(X'_k, X'_j) \right. \\
&\qquad \left. - \left(\sum_k f(X_k) + \beta \sum_k \sum_{X_j \in S_{-k}} g_{kj}(X_k, X_j) \right) \right)
\end{aligned}$$

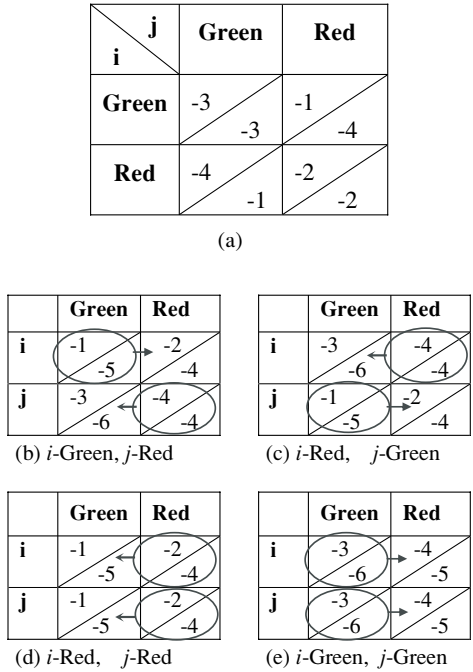


Figure 9. The LEF is inconsistent with the GEF = $\sum E_{\mathrm{LEF}}$ in a 2-node-linked graph of a 2-coloring problem. (a) LEF values for node i (upper) and j (lower). For example, if i is Green and j is Red, the LEF value of i is -1, and the LEF value of j is -4. (b) through (e) are LEF values (upper) and GEF values (lower) of each configuration for nodes i and j. $\rightarrow$ is the tendency to change color given the other node's color according to the LEF judgment. $\longrightarrow$ is the tendency to change color given the other node's color according to the GEF judgment. (d) is the Nash equilibrium of the LEF methods, and (e) is the local optimum of the GEF methods. In (b) and (c), the LEF and the GEF lead the system to evolve to different configurations. So they are inconsistent even though the GEF is the sum of the LEFs of all agents. If we see "Green" as "Cooperative" and "Red" as "Defective" (a) is actually the payoff matrix of the prisoner dilemma problem [7] which is one of the most interesting problems in game theory, and the LEF of each agent is the utility function of each player.

$$= \operatorname{sgn}\left(f(s') - f(s) + \beta \sum_{X_j \in S_{-i}} \left(g_{ij}(s', X_j) - g_{ij}(s, X_j) + g_{ji}(X_j, s') - g_{ji}(X_j, s) \right) \right)$$

$$= \operatorname{sgn}\left(f(s') - f(s) + \beta \sum_{X_j \in S_{-i}} (g_{ij}(s', X_j) - g_{ij}(s, X_j) + g_{ij}(s', X_j) - g_{ij}(s, X_j))\right)$$

$$= \operatorname{sgn}\left(f(s') - f(s) + 2\beta \sum_{X_j \in S_{-i}} (g_{ij}(s', X_j) - g_{ij}(s, X_j))\right).$$

And

$$\operatorname{sgn}(E^i_{\mathrm{LEF}}(S_{-i}, s') - E^i_{\mathrm{LEF}}(S_{-i}, s)) = \operatorname{sgn}\left(f(s') - f(s) + \beta \sum_{X_j \in S_{-i}} (g_{ij}(s', X_j) - g_{ij}(s, X_j))\right).$$

Because $\beta > \max_{s',s}\{|f(s') - f(s)|\}$, we have

$$\operatorname{sgn}\left(f(s') - f(s) + 2\beta \sum_{X_j \in S_{-i}} (g_{ij}(s', X_j) - g_{ij}(s, X_j))\right) = \operatorname{sgn}\left(f(s') - f(s) + \beta \sum_{X_j \in S_{-i}} (g_{ij}(s', X_j) - g_{ij}(s, X_j))\right)$$

$$\rightarrow \operatorname{sgn}(E_{\mathrm{GEF}}(S') - E_{\mathrm{GEF}}(S)) = \operatorname{sgn}(E^i_{\mathrm{LEF}}(S_{-i}, s') - E^i_{\mathrm{LEF}}(S_{-i}, s)). \blacksquare$$

Theorem 2 only gives a sufficient condition but not the necessary condition for a consistent LEF. Finding theorems for the necessary condition will be our future study because some asymmetrical penalties might be good evaluation functions for efficient search.

So far, we know we can always find a GEF = $\sum E_{\mathrm{LEF}}$ for any LEF. Not all GEFs constructed in this way are consistent with the LEF, which means they will direct the system to evolve to different configurations. However, on the other hand, not all GEFs can be decomposed to be n LEFs, such as in minimum coloring problems. If we construct a GEF as:

$$E_{\text{GEF-O}}(S) = m + n \sum_{i=1}^{n} \sum_{X_j \in S_{-i}} \delta(X_i, X_j) \tag{12}$$

where m is the current number of colors, we get a naïve evaluation function that reflects the nature of the minimum coloring problem. Since the LEF is not allowed to use global information, an agent only knows its linked agent and does not know the others, so it cannot know how many colors have been used to color the graph. It seems difficult for a single

agent to make a good decision for the whole system. One approximate solution here is LEF equation (4). If we compare its consistent GEF equation (9) with GEF equation (12), we will see equation (12) is more "rational" (rationality is discussed in section 3.2.2) than equation (9) because equation (12) gives a "fair" evaluation on all solution states while equation (9) does not. We will discuss "rational" and "partial rational" evaluation functions in our next paper. The experiments in section 3.3.2 show that the rational GEF equation (12) works much worse than the partial rational GEF equation (12).

3.2.2 Exploration of local and global evaluation functions

If an LEF is consistent with a GEF, what is the main difference between them? Is there any picture that can help us understand how LEF and GEF explore the state space?

To understand the behavior of the LEF and GEF, first of all we should understand the role of an evaluation function in searching. An important notion here is that the evaluation function is not a part of the problem, it is a part of the algorithm:

GT-SS Algorithms = Evaluation Function + Exploration Heuristics.

Therefore, the evaluation function values (EF values) are not built-in attributes for configurations of a given combinatorial problem. In the backtracking algorithm, there is no evaluation function and only two possible judgments for current partial configurations: "no conflict" or "conflict" (i.e., the evaluation function in backtracking has only two EF values). In the GT paradigm, since the system needs to decide whether it accepts a new state or not, it has to compare the new and current states—which one might be better? Which one might be closer to the solution state? For this purpose, the evaluation function is introduced to make approximate evaluations for all possible states and lead the system to evolve to a solution state. Except for solution states, it is always difficult to compare which configuration is better than the other. So the evaluation function is an internal "model" for the actual problem. As we know, models do not always exactly reflect everything of the reality, depending on their purpose. Therefore, different models can be built for a problem. For example, there are several different GEFs for the coloring problem.

A *rational* evaluation function satisfies: (1) all solution states should have smaller EF values than all nonsolution states; (2) all solution states should have the same EF value. A *good* evaluation function will lead the system to evolve to a solution state more directly. For example, evaluation Function 1 is more efficient than Evaluation Function 2 in Figure 10 because Evaluation Function 2 has many local optima that will trap the evolution. However, it is still impossible to construct an evaluation function that has no local optima for many combinatorial problems.

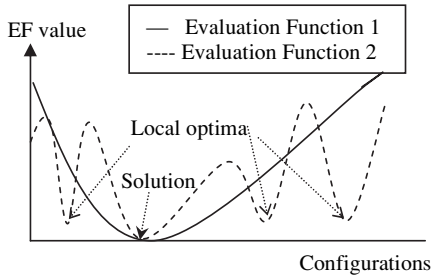


Figure 10. Different evaluation functions for a problem.

This is why they are so difficult to solve and people have developed so many heuristics to explore the configuration space. Otherwise, for Evaluation Function 1, we can simply use the greedy heuristic.

Searching (as in the GT-SS style) for solutions is just like walking in a dark night using a map. The map is the model for the configuration space, so the evaluation function is the generator of the map. Heuristics are for exploration in the space based on the map. So an algorithm includes what map it uses and how it uses the map to explore.

It is impossible to compute evaluation values for all configurations at a time. It is only possible to get evaluation values for a small part of the configuration space at each time step. This is like using a flashlight to look at a map on a dark night. But since the map is very big, only part of the map can be seen unless the flashlight is very "powerful."

Using the metaphors of "map" and "flashlight," we now illustrate the exploration of LEF and GEF.

GEF: One *n*-dimensional map and neighborhood flashlight. Traditional GEF methods use only one map, which is n-dimensional for n-variable problems. For each time step, the system computes the evaluation values of neighbors in N_S^1, that is, it uses a flashlight to look at the neighborhood of the current position on the map and make a decision about where to go (see Figure 11). There are several strategies for making decisions based on the highlighted part of the map: greedy, always selects the best one (Figure 11); greedy-random, most of the time selects the best, but with a small probability will go to a random place (GEF-GR, see section 3.3); totally random, and others. In simulated annealing, the flashlight only highlights one point (configuration) each time in the neighborhood, and will move to that point if it is better than the current one, or with a small probability ($e^{-\Delta E/T}$) move to that point even though it is worse. There can be many variations in types of flashlight and strategies for selection on the highlighted part of the map. In section 3.3, we show that both the evaluation function (map) and the exploration methods will affect performance.

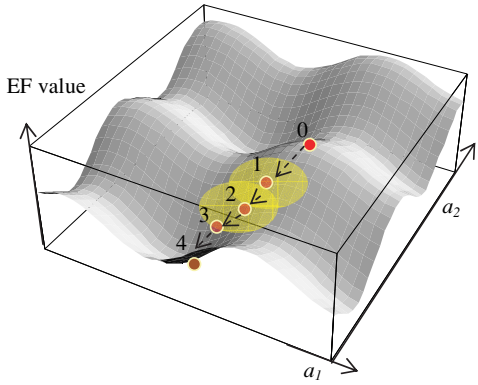


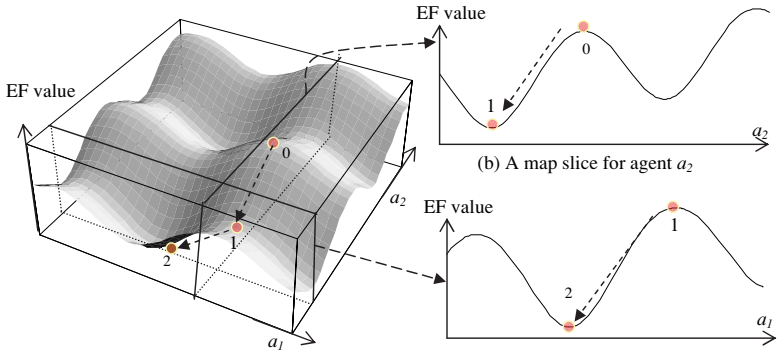
Figure 11. The GEF methods for a system consisting of two agents (a_1 and a_2). The evolution path is $0 \to 1 \to 2 \to 3 \to 4$. The light circle around 1 is the part highlighted by the flashlight when the system is in state 1. 2 is the best neighbor inside this circle. So the system accepts 2, highlights its neighborhood, gets to 3, and so on.

LEF: n d_i-dimensional maps and slice flashlight. Using the distributive LEF methods, agent a_i has a d_i-dimensional map if a_i has d_i constrained agents (in the coloring problem, d_i is the degree of a_i). In every time step each agent uses its one-dimensional slice of its d_i-dimensional map to guide its evolution.

Consistent LEF and inconsistent LEF are very different. For consistent LEF a map of a_i is actually a d_i-dimensional slice of the GEF map of $E_{\mathrm{GEF}} = \sum E_{\mathrm{LEF}}$. In other words, all LEF maps can make up a GEF map. At every time step, each agent uses a one-dimensional slice of the n-dimensional GEF map of $E_{\mathrm{GEF}} = \sum E_{\mathrm{LEF}}$. Since the system dispatches all agents sequentially, its exploration is from slice to slice. As Figure 12 shows, the two-agent system evolves as $0 \to 1 \to 2$ using slices (b) and (c) of the GEF map (a). The flashlight illuminates a one-dimensional slice each time and the agent makes a decision according to its strategies: least-move, always select the best one (Figure 12); better-move, select a random one and accept it if it is better than the current assignment; random-move, which is totally random selection, or others. So here the LEF and GEF use the same map and the difference lies in how they use the map: what flashlights and strategies are used for selecting from the highlighted part of the map.

For inconsistent LEF a map of a_i is no longer a d_i-dimensional slice of the GEF map of $E_{\mathrm{GEF}} = \sum E_{\mathrm{LEF}}$. Each agent uses its own map. So here the differences between the LEF and GEF lie not only in how they use the map, but also on which map they are using.

To conclude this section, we summarize the main differences between global and local evaluation functions in Table 3 from two layers: eval-



(a) The GEF map of a system consisting of agents a_1 and a_2

Figure 12. Consistent LEF: map slices of GEF and exploration. The system starts from "0". Suppose it is the turn of a_2 to move, then a_2 gets a slice (b) through "0" and selects the best move "1". Then it is the turn of a_1 to move, a_1 gets a slice (c) through "1" and selects the best move "2" and then repeats. So the exploration is from one slice (dimension) to another: $0 \to 1 \to 2$.

		GEF	LEF	
			Consistent	Inconsistent
Evaluation Function	Generator	E_{GEF}	$\sum E_{\text{LEF}}$	E^i_{LEF}, $(i = 1..n)$
	Dimension	n (number of nodes)	d_i (degree of each node V_i)	
	Equilibria	Local optima $L(E_{\text{GEF}})$	Nash equilibria $N(E_{\text{LEF}})$	
			$N(E_{\text{LEF}})=L(\sum E_{\text{LEF}})$	$N(E_{\text{LEF}})\neq L(\sum E_{\text{LEF}})$
Exploration Heuristics	Flashlight (search scope)	Neighborhood	one-dimensional slice	
	Strategies	Greedy, Greedy-random, SA, Random, …	Least-move, better-move, random, …	

Table 3. Differences between the GEF and consistent/inconsistent LEF.

uation function (map) and exploration heuristics. This paper focuses on the evaluation function layer. We believe that using a different evaluation function will result in different performance. The exploration heuristics also affect performance. This is discussed in the next section.

3.3 Experimental results

In the previous sections we proposed LEF methods and compared them to traditional GEF methods. We now ask the following questions about

performance: Can the LEF work for coloring problems? Can an LEF (GEF) solve problems that a GEF (LEF) can solve if they are inconsistent? If an LEF and a GEF are consistent, is their performance similar? In this section we present some preliminary experimental results on a set of coloring-problem benchmarks from the Center for Discrete Mathematics and Theoretical Computer Science.[4] Sections 3.3.1 and 3.3.2 separately show results on the k-coloring and minimum coloring problems. Some observations are given in section 3.3.3.

Note that comparisons between local and global evaluation functions are difficult for two reasons. First, the LEF and GEF are concepts for evaluation functions. One specified LEF/GEF includes several algorithms according to their exploration heuristics and different parameter settings. Second, it is difficult to avoid embodying some preliminary knowledge in the algorithm. So the following experimental comparisons are limited to the specified settings of algorithms and problem instances.

We now compare the performance of the following three algorithms using different evaluation functions and specified parameters.

1. **LEF method:** Alife&AER [12, 13].
2. **GEF method:** Greedy-random (GEF-GR) always looks for the best configuration in a N_S^1 neighborhood and has a small random walk probability.
3. **GEF method:** Simulated annealing (GEF-SA) [29].

Here are details of the Evolution($S(t)$) of the three algorithms in the framework shown in section 3.1.

- **LEF**

  Evolution(S)

  {

  1. For each agent $i = 1$ to n do
  2. Choose a random number r in [0, 1]
  3. If $r < P_{\text{least-move}}$ // perform least-move strategy
  4. Find one color c that for all other possible colors b, $E^i_{\text{LEF}}(S_{-i}, c) \le E^i_{\text{LEF}}(S_{-i}, b)$
  5. Else // perform random walk
  6. c = random-select (current possible colors)
  7. S' = replace(S, i, X_i, c)
  8. If S' is a feasible solution, call ProcessSolution(S')
  9. Return S'

  }

[4] Available at: http://mat.gsia.cmu.edu/COLOR/instances.html.

- **GEF-GR**

Evolution(S)
{
1. Choose a random number r in [0, 1]
2. If $r < P_{\text{greedy}}$ // perform greedy strategy
3. Find one neighbor $S' \in N_S^1$, that for all other $S'' \in N_S^1$, $E_{\text{GEF}}(S') \le E_{\text{GEF}}(S'')$
4. Else // perform random walk
5. S' = random-select(N_S^1)
6. If S' is a feasible solution, call ProcessSolution(S')
7. Return S'

}

- **GEF-SA**

Evolution(S)
{
1. S_r = random-select(N_S^1)
2. $\Delta = E_{\text{GEF}}(S_r) - E_{\text{GEF}}(S)$
3. If $\Delta \le 0$ // perform down-hill move
4. $S' = S_r$
5. Else $\Delta > 0$
6. Choose a random number r in [0, 1]
7. If $r \le e^{-\Delta/T}$ // perform up-hill move
8. $S' = S_r$
9. Else // do not accept the new state
10. return S
11. If S' is a feasible solution, call ProcessSolution(S')
12. Return S'

}
ProcessSolution(S)
{
#If it is a k-coloring problem:
Output the result S
#If it is a minimum coloring problem:
If S uses m colors and m is the chromatic number of the graph
Output the result S and m
Else if S uses m colors and m is less than the current number of possible colors
Current Number of Possible Colors = m
// reduce the number of possible colors, so N_S^1 is reduced

}

Algorithm	Measurement	
	An agent move of a_i	**A time step**
LEF-Alife&AER	$O(2\|e_i\| + m)$	$O\left(\sum_{i=1}^{n}(2\|e_i\| + m)\right) = O(4\|e\| + nm)$
GEF-GR	$O(2\|e_i\| + nm + n)$	$O(2\|e_i\| + nm + n)$
GEF-SA	$O(2\|e_i\|)$	$O(2\|e_i\|)$

Table 4. Complexity of an agent move and a time step of the LEF-Alife&AER, GEF-GR, and GEF-SA algorithms.

The setting of $P_{\text{least-move}}$ in the LEF method is equal to $1.5n/(1.5n + 1)$, where n is the problem size. P_{greedy} in GEF-GR is also simply set to $1.5n/(1.5n + 1)$. Parameter settings in GEF-SA are the same as in [29]: *initialT* = 10, *freezeLim* = 10, *sizeFactor* = 2, *cutoff* = 0.1, *tempFactor* = 0.95, and *minPercent* = 0.02.

The following experiments examine the performance of finding a solution. First, we give the CPU runtime on a PIII-1G PC WinXP for finding a solution and then measure the runtime as operation counts [38], which is the number of agent moves used in finding a solution.

The cost of an agent move in the LEF includes steps 2 through 8 in Evolution(s), and a time step includes n agent moves. The cost of an agent move in GEF-GR includes steps 1 through 6. The cost of an agent move in GEF-SA includes steps 1 through 11. A time step in both GEF-GR and GEF-SA only includes one agent move. The complexity mainly relates to calculating conflicts and comparing EF values (the complexity of picking up the smallest value from m values is $O(m)$). Table 4 lists the complexities of the LEF, GEF-GR, and GEF-SA methods where $|e|$ is the number of edges in the graph, $|e_i|$ is the number of edges from node V_i, and m is the current number of colors being used. So, in a k-coloring problem m is constantly equal to k, and in a minimum coloring problem m is a dynamic decreasing value but is always less than n.

So from Table 4 we can get this relationship of complexity for an agent move:

GEF-SA < LEF < GEF-GR.

It is easy to think that the more complex an agent's move is, the better its performance will be. If so, the number of agent moves for finding a solution would have the relation:

GEF-SA moves > LEF moves > GEF-GR moves.

Is this true? Let us look at the experimental results in sections 3.3.1 and 3.3.2 where some comparisons are given.

3.3.1 *k*-coloring problem

We compare the CPU runtime (Figure 13) and the number of moves (Figure 14) for finding a solution for a given k colors using different LEFs (equations (3) and (4)) and GEFs (equations (1), (2), and (9)). The following comparisons are included:

a. Consistent cases: LEF equation (3) *versus* GEF equation (1), and LEF equation (4) *versus* GEF equation (9).

b. Inconsistent cases: LEF equation (3) *versus* GEF equation (9), LEF equation (4) *versus* GEF equation (1), LEF equation (3) *versus* GEF equation (2), and LEF equation (4) *versus* GEF equation (2).

c. Different evaluation functions for LEF/GEF: LEF equations (3) *versus* (4), GEF equations (1) *versus* (2) *versus* (9).

d. Different exploration heuristics for the same evaluation function: GEF-GR equation (1) *versus* GEF-SA equation (1) *versus* LEF equation (3).

3.3.2 Minimum coloring problem

Finding the chromatic number of the graph in a minimum coloring problem is a global goal. The LEF of each agent can only use local information to guide itself and sometimes there is not enough information to make a good decision. We test the CPU runtime (Figure 15) and number of agent moves (Figure 16) for finding a $\chi(G)$-color solution when $\chi(G)$ is unknown. The system starts with a large number of colors m, and decreases it if a feasible m-color solution is found, until m is equal to $\chi(G)$. The following comparisons are included:

a. Consistent case: LEF equation (4) *versus* GEF equation (9).

b. Inconsistent case: LEF equation (4) *versus* GEF equation (2) *versus* GEF equation (12).

c. Different GEFs: GEF equations (2) *versus* (9) *versus* (12).

d. Different exploration heuristics for the same evaluation function: GEF-GR equation (2) *versus* GEF-SA equation (2).

3.3.3 Observations from the experiments

(1) LEFs work for the *k*-coloring and minimum coloring problems. The GEF does not always lead the evolution more efficiently than the LEF although it uses more information to make decisions and has centralized control over the whole system. In the k-coloring problems, the LEF (especially LEF equation (4)) beats the GEFs in CPU runtime in almost all instances. In the minimum coloring problems, although the LEF only knows local information and the purpose is global, it still beats the

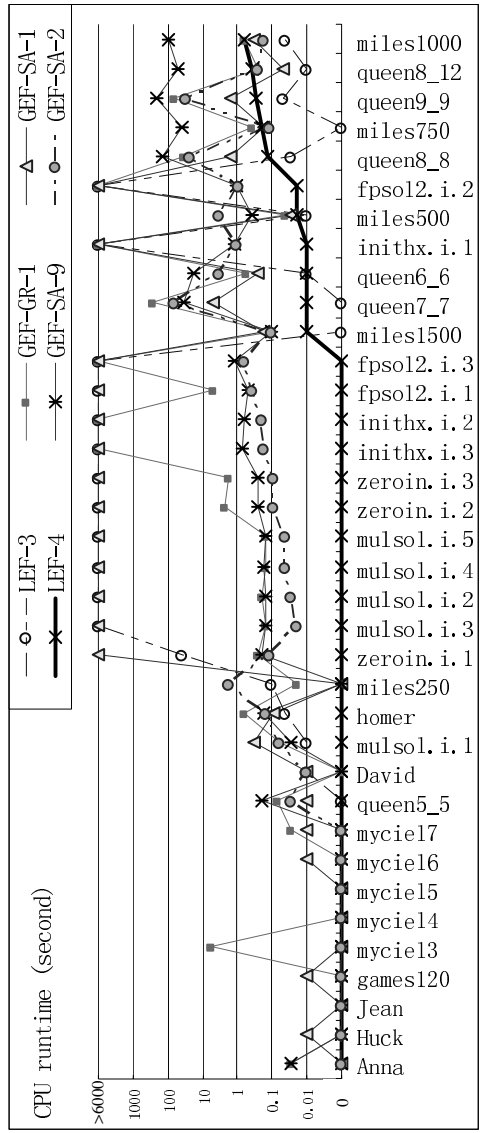


Figure 13. Average CPU runtime (second) of 100 runs for the k-coloring problems. Note that the y-axis is a log-plot, with the lowest value denoting zero (less than 0.001 second) and the highest value denoting that the runtime is more than 6000 seconds.

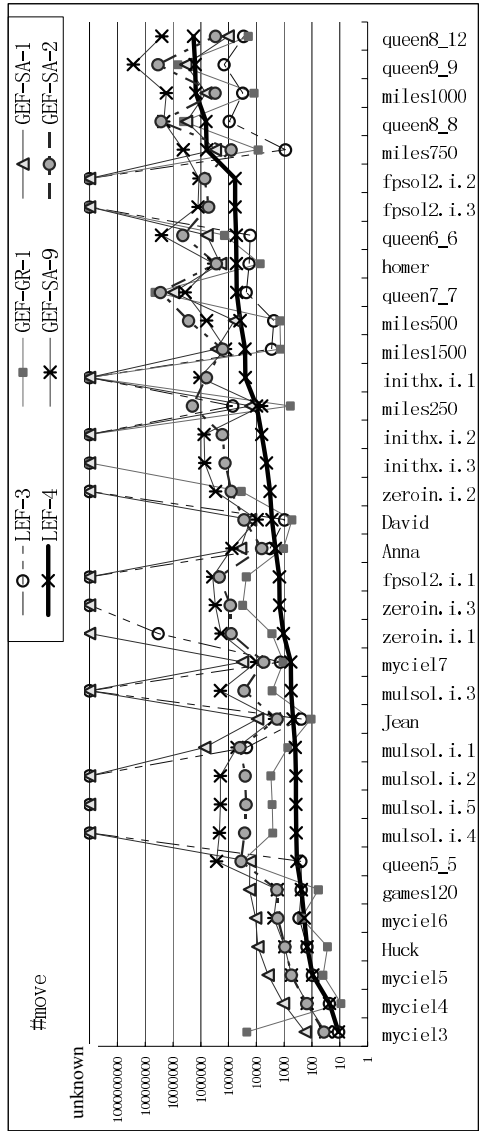


Figure 14. Average number of agent moves of 100 runs for the k-coloring problems. Note that the y-axis is a log-plot, with the highest value "unknown" denoting that we have not tested the number of agent moves because the runtime is more than 6000 seconds.

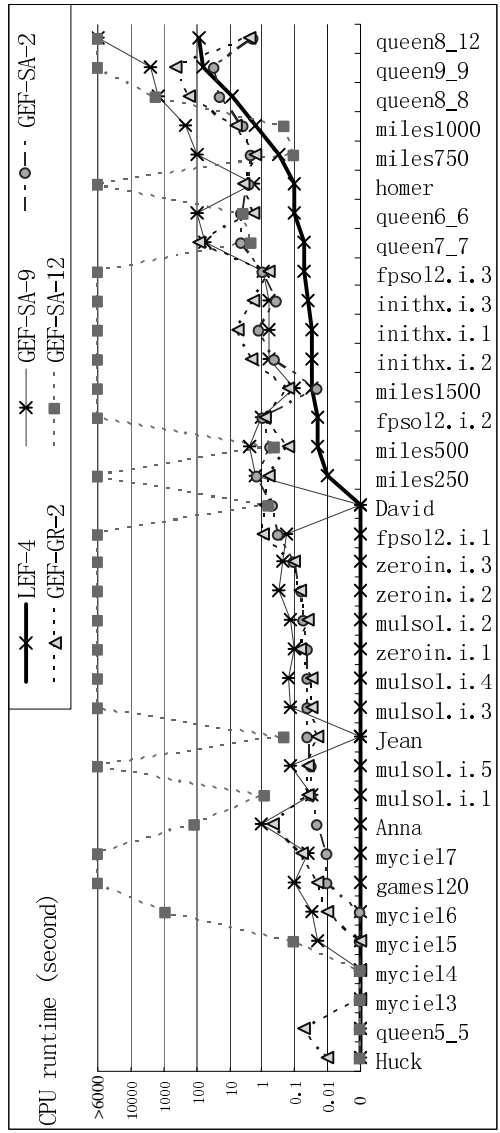


Figure 15. Average CPU runtime (second) of 100 runs for solving the minimum coloring problems. Note that the *y*-axis is a log-plot, with the lowest value denoting zero and the highest value denoting that the runtime is more than 6000 seconds.

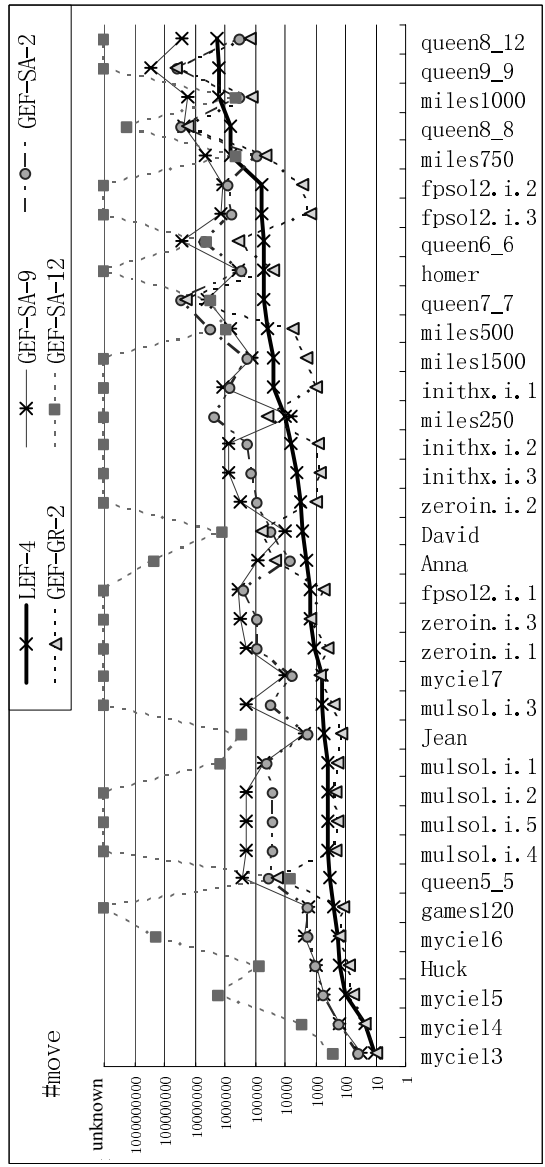


Figure 16. Average number of agent moves of 100 runs for solving the minimum coloring problems. Note that the *y*-axis is a log-plot, with the highest value "unknown" denoting that we have not tested the number for agent moves because the runtime is more than 6000 seconds.

GEFs for most instances except *queen8_12* and *queen9_9*. The LEF is much faster than the GEFs for *homer*, *miles250*, *queen7_7*, *queen8_8*, *fpsol2*, *inithx*, *mulsol*, and *zeroin*. Although GEF equation (9) is consistent with LEF equation (4) and GEF equation (2) works better than GEF equation (9) in most instances, LEF equation (4) still beats GEF equation (2). In addition to coloring problems, the LEF also works for N-queen problems [12–14]. Using the LEF for the N-queen problem can beat a lot of algorithms, but it is not as fast as one specified local search heuristic [18]. One important reason is that the specified local search heuristic embodies preliminary knowledge of the N-queen problem. For more experimental results, such as distributions and deviations of runtime, and performance based on different parameter settings, see [12–14].

(2) Evaluation function affects performance. As we know, the GEF selection will affect performance when solving function optimization problems [39]. This is also true for combinatorial problems. For the k-coloring problem (a decision problem) the performance of LEF equations (3) and (4) is very different. In *fpsol2*, *inithx*, *mulsol*, and *zeroin*, equation (4) beats equation (3) strongly, while equation (3) slightly beats equation (4) in *miles* and *queen*. GEF equation (2) can solve *fpsol2*, *inithx*, *mulsol*, and *zeroin* within 2 seconds, while GEF equation (1) cannot solve it within 6000 seconds. We guess that is because part (a) of equation (2) puts pressure on each node to select a small index number color, which helps to break the symmetry [40] in the coloring problems [41]. In the minimum coloring problem (an optimization problem), the performance difference is very clear. GEF equation (12) cannot solve more than half of those instances, while the others can solve them. GEF equation (2) works better than GEF equation (9) on average.

(3) Performance of an LEF and a GEF are more alike if they are consistent. In the k-coloring problem, LEF equation (4) is consistent with GEF equation (9) while LEF equation (3) is consistent with GEF equation (1), so performance between LEF equation (3) and GEF equation (1) is more alike than that of LEF equation (4) and GEF equation (9). LEF equation (4) and GEF equation (9) can solve *fpsol2*, *inithx*, *mulsol*, and *zeroin* quickly while LEF equation (3) and GEF equation (1) cannot. LEF equation (3) and GEF equation (1) beat LEF equation (4) and GEF equation (9) in *miles1000* and *queen8_12*. In the minimum coloring problem, for example, *queen8_12* is difficult for LEF equation (4) and GEF equation (9), but easy for GEF equation (2). These results tell us that if the Nash equilibria of the LEF are identical to the local optima of the GEF, they will behave more alike.

(4) Exploration heuristics affect performance. There are so many efforts on exploration heuristics that we know they will affect performance.

This is also true for the consistent LEF and GEF. Even though they have the same evaluation function (map), they still work differently in some instances. In the k-coloring problem, LEF equation (3) beats GEF equation (1) in *miles* and *queens* while GEF-GR equation (1) beats LEF equation (3) and GEF-SA equation (1) in *mulsol* and *zeroin*. LEF equation (4) and GEF-SA equation (9) are consistent but the LEF works faster than GEF-SA. In the minimum coloring problems, LEF equation (4) and GEF equation (9) are consistent but LEF equation (4) beats GEF equation (9) strongly. Both using GEF equation (2), GEF-SA and GEF-GR perform differently, especially in *queen7_7*, *queen8_8*, and *queen9_9*.

(5) Larger complexity does not always mean better performance of an agent movement. We have the ranking for complexities of an agent move in different algorithms: GEF-GR > LEF > GEF-GR. However, GEF-GR does not always use the fewest agent moves to reach the (optimal) solution state in coloring problems, even in the consistent case. In both the k-coloring and minimum coloring problems, the LEF uses fewer moves than GEF-GR in some instances, such as *queen5_5* and *queen7_7*. So this suggests that in some instances using local information does not always mean decreasing the quality of the decision. Even using the same evaluation function, GEF-GR looks at n neighbors and selects one, while GEF-SA only looks at one neighbor, we still see in some problem instances, such as *queen5_5*, *queen7_7*, and *queen9_9*, that GEF-SA uses fewer moves than GEF-GR when solving k-coloring problems.

(6) Performance of the LEF and GEF is dependent on the problem instance. The LEF fits for most instances except *queen8_12* in the minimum coloring problem. GEF equation (1) and LEF equation (3) are not good for *fpsol2*, *inithx*, *mulsol*, and *zeroin*. As in our other paper [14], the algorithmic details will not change the performance for a problem instance very much. We guess the graph structure (constraint network) will have a big impact on the performance of the LEF and GEF. In other words, the LEF might be good for some networks while the GEF might be good for other networks. In the language of multiagent systems, some interaction structures are more suitable for self-organization to reach a global purpose, while some interaction structures favor centralized control. Finding the "good" networks for the LEF and GEF is a very important future topic.

4. Conclusions

This paper proposes a new look at evolving solutions for combinatorial problems, using NP-complete cases of the k-coloring problem (a decision problem) and the minimum coloring problem (an optimization problem). The new outlook suggests several insights and lines of future work and applications.

Finding solutions for a coloring problem by using generate-test-single-solution (GT-SS) algorithms is like the evolution of a multiagent system. The evaluation function of the GT-SS algorithms is the internal ranking of all configurations. There are two ways to construct evaluation functions: the global evaluation function (GEF) uses global information to evaluate a whole system state, and the local evaluation function (LEF) uses local information to evaluate a single agent state. We analyze the relationship between LEF and GEF in terms of consistency and inconsistency. Table 3 summarizes the differences between GEF and consistent/inconsistent LEF.

A consistency theorem is proposed (Theorem 1) which shows that Nash equilibria (Definition 6) of an LEF are identical to the local optima (Definition 5) of a GEF if the LEF is consistent with the GEF. In spin glass theory, this theorem means that the LEF and GEF have the same ground state structure and the same number of solutions, as well as the same phase transitions [1]. So the LEF can be partly explained by the current theoretical results on traditional GEF [16, 37, 42, 43]. This helps us understand why the LEF can guide agents to evolve to a global goal state, if the global goal can be expressed by a GEF that is consistent with the LEF. When an LEF is inconsistent with a GEF, the LEF's Nash equilibria are not identical to the GEF's local optima. Obviously, the LEF evolution proceeds differently than the GEF evolution. Experimental results support this: performance of LEF and GEF are more alike if they are consistent.

Since the way to construct a GEF is not unique for a problem, we can always construct a GEF by summing up all of the agents' LEFs as $E_{\mathrm{GEF}} = \sum E_{\mathrm{LEF}}$. $\sum E_{\mathrm{LEF}}$ can be consistent or inconsistent with E_{LEF}. This paper gives a preliminary theorem for constructing consistent LEF which have the form of the spin glass Hamiltonian (equation (11)): If the penalty function $g_{ij}(X_i, X_j)$ in an E_{LEF} is a symmetrical penalty, the E_{LEF} is consistent with $\sum E_{\mathrm{LEF}}$. More study on consistent and inconsistent LEFs should give more insights into computation and game theory.

Even though an LEF is consistent with a GEF, they still explore differently. The experimental results show that exploration heuristics affect performance, so even though the LEF is consistent with a GEF, it still behaves differently and works better in some instances.

The LEF and GEF concepts provide both a new view of algorithms for combinatorial problems, and for the collective behavior of complex systems. Using coloring problems clarifies the concepts of "local" and "global." These concepts are important distinguishing characteristics of distributed and centralized systems. In addition to the research reported, there is still a lot of future work. Some remaining questions are: What is the condition for the existence of a consistent GEF for any LEF? What is the condition for the existence of a consistent LEF for any GEF? How do network properties relate to consistency of LEF and GEF in the coloring

problems? How can we construct good LEFs and GEFs for a problem? LEF is about individual and GEF is about the whole system, how about the evaluation functions for compartments? Can we study consistent and inconsistent LEFs in the context of game theory? These will be our future study.

5. Acknowledgments

This paper is supported by the International Program of the Santa Fe Institute and the National Natural Science Foundation of China. The author wants to thank Lei Guo and John Holland for their strong support and discussions. Thanks also go to Mark Newman, Eric Smith, Haitao Fang, Li Ming, Cris Moore, Supriya Krishnamurthy, Dengyong Zhou, Melanie Mitchell, Jose Lobo, John Miller, Sam Bowles, Jim Crutchfield, Chien-Feng Huang and other people for helpful discussions. Thanks to Laura Ware for proofreading.

References

[1] Remi Monasson, Riccardo Zecchina, Scott Kirkpatrick, Bart Selman, Lidror Troyansky, "Determining Computational Complexity from Characteristic 'Phase Transitions,'" *Nature*, **400** (1999) 133–137.

[2] John von Neumann, *Theory of Self-Reproducing Automata* (University of Illinois Press, Urbana-Champaign, IL, 1966).

[3] http://www.red3d.com/cwr/boids/applet. A demonstration of the Boids model of bird flocking is provided.

[4] Martin Gardner, "Mathematical Games: The Fantastic Combinations of John Conway's New Solitaire Game 'Life,'" *Scientific American*, **223** (October 1970) 120–123; http://www.bitstorm.org/gameoflife.

[5] Dirk Helbing, Illes J. Farkas, and Tamas A. Vicsek, "Simulating Dynamical Features of Escape Panic," *Nature*, **407** (2000) 487–490.

[6] P. Bak, C. Tang, and K. Wiesenfeld, "Self-organized Criticality," *Physics Review A*, **38** (1988) 364–374.

[7] R. Axelrod, *The Evolution of Cooperation* (Basic Books, New York, 1984).

[8] David Wolpert and Kagan Tumer, "An Introduction to Collective Intelligence," in *Handbook of Agent Technology*, edited by A. Jeffrey and M. Bradshaw (AAAI Press/MIT Press, Cambridge, 1999).

[9] M. Dorigo and G. Di Caro, "The Ant Colony Optimization Metaheuristic," in *New Ideas in Optimization*, edited by D. Corne, M. Dorigo, and F. Glover (McGraw Hill, UK, 1999).

[10] S. Kirkpatrick, C. D. Gellat, Jr., and M. P. Veechi, "Optimization by Simulated Annealing," *Science*, **220**(4589) (1983) 671–681.

[11] S. Boettcher and A. G. Percus, "Nature's Way of Optimizing," *Artificial Intelligence*, **119** (2000) 275–286.

[12] Han Jing, Jiming Liu, and Cai Qingsheng, "From ALIFE Agents to a Kingdom of *N* Queens," in *Intelligent Agent Technology: Systems, Methodologies, and Tools*, edited by Jiming Liu and Ning Zhong (The World Scientific Publishing Co. Pte, Ltd., 1999). A demonstration of *Alife Model* for solving *N*-queen problems can be downloaded from http://www.santafe.edu/~hanjing/dl/nq.exe.

[13] Jiming Liu, Han Jing, and Yuan Y. Tang, "Multiagent Oriented Constraint Satisfaction," *Artificial Intelligence*, **136**(1) (2002) 101–144.

[14] Han Jing and Cai Qingsheng, "Emergence from Local Evaluation Function," *Journal of Systems Science and Complexity*, **16**(3) (2003) 372–390; Santa Fe Institute working paper 200208036.

[15] Vipin Kumar, "Algorithm for Constraint Satisfaction Problem: A Survey," *AI Magazine*, **13**(1) (1992) 32–44.

[16] J. van Mourik and D. Saad, "Random Graph Coloring: A Statistical Physics Approach," *Physics Review E*, **66**(5) (2002) 056120.

[17] M. E. J. Newman, "The Structure and Function of Networks," *Computer Physics Communications*, **147** (2002) 40–45.

[18] Rok Sosic and Jun Gu, "Efficient Local Search with Conflict Minimization: A Case Study of the *N*-queen Problem," *IEEE Transactions on Knowledge and Data Engineering*, **6**(5) (1994) 661–668.

[19] V. Kumar, "Depth-First Search," in *Encyclopaedia of Artificial Intelligence, Volume 2*, edited by S. C. Shapiro (John Wiley and Sons, Inc., New York, 1987).

[20] S. Minton, M. D. Johnston, A. B. Philips, and P. Laird, "Minimizing Conflicts: A Heuristic Repair Method for Constraint Satisfaction and Scheduling Problems," *Artificial Intelligence*, **52** (1992) 161–205.

[21] Bart Selman, Henry A. Kautz, and Bram Cohen, "Local Search Strategies for Satisfiability Testing," in *Second DIMACS Challenge on Cliques, Coloring, and Satisfiability*, edited by M. Trick and D. S. Johnson, Providence, RI, USA, 1993.

[22] D. McAllester, B. Selman, and H. Kautz, "Evidence for Invariants in Local Search," in *Proceedings of AAAI'97* (AAAI Press/The MIT Press, 1997).

[23] F. Glover, "Tabu Search: Part I," *ORSA Journal on Computing*, **1**(3) (1989) 190–206.

[24] F. Glover, "Tabu Search: Part II," *ORSA Journal on Computing*, **2**(1) (1990) 4–32.

[25] B. Selman, H. Levesque, and D. Mitchell, "A New Method for Solving Hard Satisfiability Problems," in *Proceedings of AAAI'92* (MIT Press, San Jose, CA, 1992).

[26] B. Selman, Henry A. Kautz, and Bram Cohen, "Noise Strategies for Improving Local Search," in *Proceedings of AAAI'94* (MIT Press, Seattle, WA, 1994).

[27] J. H. Holland, *Adaptation in Natural and Artificial Systems* (The University of Michigan Press, Ann Arbor, 1975).

[28] J. Kennedy and R. Eberhart, "Particle Swarm Optimization," *IEEE International Conference on Neural Networks IV* (IEEE Service Center, Piscataway, NJ, 1995).

[29] David S. Johnson, Cecilia R. Aragon, Lyle A. McGeoch, and Catherine Schevon, "Optimization by Simulated Annealing: An Experimental Evaluation; Part II, Graph Coloring and Number Partitioning," *Operations Research*, **39**(3) (1991) 378–406.

[30] D. Brelaz, "New Methods to Color Vertices of a Graph," *Communications of the ACM*, **22** (1979) 251–256.

[31] A. Hertz and D. de Werra, "Using Tabu Search Techniques for Graph Coloring," *Computing*, **39**(4) (1987) 345–351.

[32] Jacques Ferber, *Multiagent Systems: An Introduction to Distributed Artificial Intelligence* (Addison-Wesley, New York, 1999).

[33] R. K. Ahuja, O. Ergun, J. B. Orlin, and A. P. Punnen, "A Survey of Very Large-scale Neighborhood Search Techniques," *Discrete Applied Mathematics*, **123** (2002) 75–102.

[34] Jan HM Korst and Emile HL Aarts, "Combinatorial Optimization on a Boltzmann Machine," *Journal of Parallel and Distributed Computing*, **6**(2) (1989) 331–357.

[35] E. Aarts and J. Korst, *Simulated Annealing and Boltzmann Machines: A Stochastic Approach to Combinatorial Optimization and Neural Computing* (John Wiley & Sons, 1989).

[36] Eric van Damme, *Stability and Perfection of Nash Equilibria* (Springer-Verlag, Berlin and New York, 1987).

[37] Marc Mezard, Giorgio Parisi, and Miguel Angel Virasoro, *Spin Glass Theory and Beyond* (World Scientific, Singapore and Teaneck, NJ, 1987).

[38] Ravindra K. Ahuja and James B. Orlin, "Use of Representative Operation Counts in Computational Testing of Algorithms," *INFORMS Journal on Computing*, **8**(3) (1996) 318–330.

[39] Alice Smith and David Coit, "Constraint Handling Techniques: Penalty Functions," in *Handbook of Evolutionary Computation*, edited by T. Baeck, D. Fogel, and Z. Michalewicz (Oxford University Press, 1997).

[40] Ian P. Gent and Barbara M. Smith, "Symmetry Breaking in Constraint Programming," in *Proceedings ECAI'2000*, edited by W. Horn (IOS Press, Amsterdam, 2000).

[41] A. Marino and R. I. Damper, "Breaking the Symmetry of the Graph Colouring Problem with Genetic Algorithms," in *Proceedings Genetic and Evolutionary Computation Conference (GECCO-2000), Late Breaking Papers*, Las Vegas, NV (Morgan Kaufmann Publishers, San Francisco, 2000).

[42] R. Mulet, A. Pagnani, M. Weigt, and R. Zecchina, "Coloring Random Graphs," *Physical Review Letters*, **89** (2002) 268701; cond-mat/0208460.

[43] S. Cocco, R. Monasson, A. Montanari, and G. Semerjian, "Approximate Analysis of Search Algorithms with 'Physical' Methods," cond-mat/0302003.

[44] Wim Hordijk, "A Measure of Landscapes," *Evolutionary Computation*, **4**(4) (1996) 335–360.