香港浸會大學
HONG KONG BAPTIST UNIVERSITY

計算機科學系
Department of Computer Science

Report production coordinator: Suan Choi, suan@comp.hkbu.edu.hk, +852 3411 7079
Report Web Site: http://www.comp.hkbu.edu.hk/tech-report

Self-Organized Combinatorial Optimization

Jiming Liu, Yu-Wang Chen, Yong-Zai Lu,
Gen-Ke Yang

COMP-08-001

Release Date: 8 May 2008

Department of Computer Science, Hong Kong Baptist University

Abstract

In this paper, we present a self-organized computing approach to solving hard combinatorial optimization problems, e.g., the traveling salesman problem (TSP). First of all, we provide an analytical characterization of such an approach, by means of formulating combinatorial optimization problems into autonomous multi-entity systems and thereafter examining the microscopic characteristics of optimal solutions with respect to discrete state variables and local fitness functions. Next, we analyze the complexity of searching in the solution space based on the representation of fitness network and the observation of phase transition. In the second part of the paper, following the analytical characterization, we describe a decentralized, self-organized algorithm for solving combinatorial optimization problems. The validation results obtained by testing on a set of benchmark TSP instances have demonstrated the effectiveness and efficiency of the proposed algorithm. The link established between the microscopic characterization of hard computational systems and the design of self-organized computing methods provides a new way of studying and tackling hard combinatorial optimization problems.

Report Cover Designer: So Kin Ming

Self-Organized Combinatorial Optimization

Jiming Liu, Yu-Wang Chen, Yong-Zai Lu, Gen-Ke Yang

Department of Computer Science, Hong Kong Baptist University, Kowloon Tong, Hong Kong

School of Electronic, Information and Electrical Engineering, Shanghai Jiaotong University, Shanghai 200240, China

{jiming,ywchen}@comp.hkbu.edu.hk

Abstract

In this paper, we present a self-organized computing approach to solving hard combinatorial optimization problems, e.g., the traveling salesman problem (TSP). First of all, we provide an analytical characterization of such an approach, by means of formulating combinatorial optimization problems into autonomous multi-entity systems and thereafter examining the microscopic characteristics of optimal solutions with respect to discrete state variables and local fitness functions. Next, we analyze the complexity of searching in the solution space based on the representation of fitness network and the observation of phase transition. In the second part of the paper, following the analytical characterization, we describe a decentralized, self-organized algorithm for solving combinatorial optimization problems. The validation results obtained by testing on a set of benchmark TSP instances have demonstrated the effectiveness and efficiency of the proposed algorithm. The link established between the microscopic characterization of hard computational systems and the design of self-organized computing methods provides a new way of studying and tackling hard combinatorial optimization problems.

Keywords: Combinatorial optimization; Traveling salesman problem; NP-complete; Microscopic characteristics; Fitness network; Phase transition; Self-organized computing

1. Introduction

Combinatorial optimization is pervasive in most fields of science and engineering. Its aim is to optimize an objective function on a finite set of feasible solutions. For example, the traveling salesman problem (TSP) (Gutin and Punnen 2002), one of the classical combinatorial optimization problems, can be described as a problem of search for the shortest tour among a set of cities. Generally speaking, most of combinatorial optimization problems in practice are deemed as computationally intractable, and have been proven to belong to the class of NP-complete problems (Garey and Jonhson 1979), where NP stands for "Non-deterministic Polynomial time". For NP-complete problems, although the optimality of a possible solution can be verified in polynomial time, the computational time for finding the optimal solution grows exponentially with the dimension of input variables in the worst case. Furthermore, if we can find a polynomial time algorithm to solve one NP-complete problem, then all the other NP-complete problems will become solvable in polynomial time (Papadimitriou, 1994).

However, in modern computer science, it has been commonly conjectured that there are no such polynomial time algorithms for any NP-complete problems (Cormen et al. 2001). Alternatively, a variety of nature-inspired optimization techniques have been developed for

finding near-optimal solutions of NP-complete problems within a reasonable computational time. Examples of such algorithms include: simulated annealing (Kirkpatrick, Gelatt Jr., and Vecchi 1983), genetic algorithm (Forrest 1993), ant colony optimization (Bonabeau, Dorigo, and Theraulaz 2000), particle swarm optimization (Kennedy and Eberhart 1995), among others.

It is interesting to note that most of the existing optimization methods employ a centralized control model, and rely on a global objective function for evaluating intermediate and final solutions. In dealing with hard combinatorial optimization problems, however, it would be difficult and time-consuming to collect the global information due to the interaction in, and the dimensionality of, computation. In order to overcome the limitation of existing centralized optimization, several decentralized, self-organized computing methods have been studied in recent years with the help of complex systems and complexity science.

Boettcher and Percus (2000) presented a stochastic search method called extremal optimization (EO). The method is motivated by the Bak-Sneppen evolution model (Bak and Sneppen 1993; Sneppen 1995), in which the least adapted species are repeatedly mutated following some local rules. To further improve the adaptability and performance of the EO algorithm, a variation of EO called τ-EO (Boettcher and Percus 2000) was subsequently presented by introducing a tunable parameter τ. So far, EO and its variants have successfully addressed several physical systems with two-degree-of-freedom entities, i.e., the combinatorial optimization problems with binary state variables, such as graph bi-partitioning problems (Boettcher, 2000), Ising spin glasses (Middleton, 2004), community detection in complex networks (Duch and Arenas 2005), etc. Recently, Liu et al. (2005; 2006) presented a general *autonomy-oriented computing* (AOC) framework for the optimization of self-organized distributed autonomous agents. AOC is also a bottom-up computing paradigm for solving hard computational problems and for characterizing complex systems behavior. Computational approaches based on AOC systems have been applied to distributed constraint satisfaction problem (Liu, Han, and Tang 2002), image feature extraction (Liu, Tang, and Cao 1997), network community mining problem (Yang and Liu 2007), etc. In addition, Han (2003) introduced the concept of local fitness function for evaluating the state of autonomous agents in computational systems.

In order to design a reasonable self-organized computing method, there are generally several important issues that should be considered:

1. How can we construct a mapping from combinatorial optimization problems to multi-entity complex systems?
2. What types of local fitness function should be used to guide the computational systems to evolve towards the global optimum?
3. What are the microscopic characteristics of optimal solutions?
4. How can we characterize the computational complexity of such a method in searching a solution space?

A number of literatures have been focused on one or some of the above questions over past few years (Mézard, Parisi, and Zecchina 2002; Han and Cai 2003; Goles et al. 2004; Achlioptas, Naor, and Peres 2005). In this paper, we attempt to answer each of those questions, and furthermore, provide a solid theoretical foundation for the self-organized computing method under study. First of all, we will model combinatorial optimization problems into multi-entity systems, in which a large number of self-organizing, interacting agents are involved. We then statistically examine the microscopic characteristics of optimal solutions with respect to the notions of discrete

state variable and local fitness function. Moreover, we will analyze the complexity of search in a solution space based on the representation of fitness network and the observation of phase transition. Finally, based on the analysis, we will describe a self-organized computing algorithm for solving hard combinatorial optimization problems.

The rest of the paper is organized as follows: In Section 2, we describe the mapping between combinatorial optimization problems and multi-entity systems, and then provide the analytical results of microscopic distribution on the optimal TSP solutions. In order to systematically analyze the solution space, we present the concept of fitness network, and further discuss the search complexity from the characteristics of phase transition. In Section 3, we present a self-organized combinatorial optimization method, as well as the validation results on a set of benchmark TSP instances. In Section 4, we conclude this paper.

2. Analytic characterization of combinatorial optimization problems

In this section, we present an analytic characterization of combinatorial optimization from the point of view of a self-organizing system. The characterization will provide us with insights into the design of a decentralized, self-organized computing approach to tackling combinatorial optimization problems. In particular, it points out ways of improving the efficiency of solution searching based on such an approach.

2.1. Modeling combinatorial optimization problems into multi-entity systems

Combinatorial optimization is concerned with finding the optimal combination of a set of discrete constrained variables (Papadimitriou and Steiglitz 1998). As summarized by Blum and Roli (2003), a combinatorial optimization problem, $P = (S, F)$, can be defined in terms of:

- a set of discrete variables, $X = \{x_1, \ldots, x_n\}$, and variable domains, $D_1, \ldots, D_n$;
- multiple constraints among variables;
- an objective function F to be optimized, where $F: D_1 \times \cdots \times D_n \rightarrow R$.

Correspondingly, the solution space can be represented as:

$S = \{s = (x_1, \ldots, x_n) \mid x_i \in D_i, s \text{ satisfies all the constraints}\}$,

in which each element s is a candidate solution. Given a combinatorial optimization problem, the aim is to find the optimal solution, $s^* \in S$, such that the global objective, $F(s^*) \leq F(s)$ $(\forall s \in S)$.

Since combinatorial solutions often depend on a non-trivial combination of *multiple elements* with specific states, a combinatorial optimization problem can be straightforwardly translated into a *multi-entity* computational system, if we formulate it from the viewpoint of a complex system. In a complex computational system, each entity can be viewed as an *autonomous agent*, and the agent can collect local (limited) information from the environment, and acts with other interacting agents for achieve the system objective (Liu, Jin, and Tsui 2005). Specifically, an autonomous computational agent, *a,* can be characterized based on the following properties (Liu, Han, and Tang 2002):

- a finite number of possible states and some behavioral rules;
- local information about the computational system and its conditions;
- certain local objectives.

Thus, a multi-entity computational system can be further modeled based on:

- a group of agents, $A = \{a_1, \ldots, a_n\}$;

- interactions between agents;
- a set of reactive rules, governing the interactions among agents.

Drawing on the above intuitive idea of mapping between the representation of multi-entity computational systems and the definition of combinatorial optimization, various methodologies for complex systems modeling and analysis can be explored and applied to handle combinatorial computing problems. Generally speaking, the modeling and analysis techniques for complex systems can be classified into top-down and bottom-up approaches (Liu, Jin, and Tsui 2005). Top-down approaches follow the idea of centralized control, and mainly study the macroscopic system-level characteristics of a complex system. On the contrary, bottom-up approaches, which were developed more recently, employ the concept of decentralized control, and apply the entity-oriented models to characterize the microscopic emergent properties. In order to address the complexity of practical computational systems, we will, in the paper, focus on the study of a novel decentralized, self-organized computing method, as opposed to earlier work on centralized combinatorial optimization.

2.2. Local fitness function

In a multi-entity computational system, the process of self-organization in individual entities is crucial for searching the global optimal solution of the whole system. At any moment, each entity must assess its local conditions, and thus decide its primitive behavior. In doing so, a local fitness function (here, the term "fitness function" is synonymous with objective function) is essential for evaluating the specific states of local entities. For the sake of explanation and illustration, in what follows we will take the TSP as a walk-through example of combinatorial optimization problems.

The TSP has attracted much interest in the computing communities in the past decades due to the fact that it provides insights into a wide range of theoretical questions in discrete mathematics, theoretical computer science, computational biology, among others, and offers models and solutions to many real-world practical problems, ranging from scheduling, transportation, to genome mapping. The TSP is usually stated as an optimization problem of finding the shortest closed tour that visits each city only once. In other words, given a set of n cities and the distance measure, d_{ij}, for all pairs i and j, the optimization goal is to find a permutation π of these n cities that minimizes the following global fitness function:

$$F(s) = \sum_{i=1}^{n-1} d_{\pi(i),\pi(i+1)} + d_{\pi(n),\pi(1)} \tag{1}$$

For a TSP solution s in the solution space S, it can be exclusively represented by the discrete states of all cities. In this paper, if city i, which can be represented as entity x_i, is connected to its kth ($1 \le k \le n-1$) nearest neighbor, the discrete state variable of city i is defined as:

$$s(x_i) = k, \quad i = 1,2,\cdots,n-1 \tag{2}$$

In an ideal computational system, all cities will be connected to their first nearest neighbors, i.e., $s(x_i) = 1$ for all i ($1 \le i \le n$). However, such ideal microstates are often frustrated by the interacting effects between cities, i.e., competitions on the selection of forward-directed edges. As a result, a city may not always be connected to its first nearest neighbor, and possibly dwell on a specific energy state except the ground state, i.e., $s(x_i) > 1$. Correspondingly, a local fitness function can be formulated to evaluate the microscopic dynamical characteristics of all cities.

Now let us assume d_i to be the length of the forward-directed edge starting from city i in a feasible solution s, the local fitness of city i can be defined as:

$$f_s(x_i) = d_i - \min_{j \neq i} d_{ij}, \quad i = 1,2,\cdots,n \tag{3}$$

Correspondingly, the global fitness function for any possible solution s can be represented as:

$$F(s) = \sum_{i=1}^{n} \min_{j \neq i} d_{ij} + \sum_{i=1}^{n} f_s(x_i) \tag{4}$$

Obviously, the first term in Eq. (4) is a constant, which can serve as a lower bound for a given TSP instance. The second term is the sum of the local fitness for all entities. That is to say, the global fitness of a combinatorial solution can be represented as a function of distributed local fitness, and the optimal solution is equivalent to the microstate with the minimum sum of local fitness for all entities. Intuitively, we can optimize the computational system through updating the states of those entities with worse local fitness. In a complex computational system, the relationship between discrete states and local fitness is usually nonlinear, and the value of local fitness is not necessarily equal even if two entities have the same state value.

Since local fitness is not as explicit as global fitness, as discussed in our earlier work (Chen and Lu 2007), the following questions will inevitably arise:

1. Can the optimization guided by local fitness achieve the global optimum of computational systems?
2. What will be the relationship between local fitness and global fitness?

For answering the above questions, in the following we introduce the consistency and equivalence conditions between global fitness and local fitness.

Definition 1. Suppose that solution s' is constructed from s ($\forall s \in S$) by updating the state of entity i using a neighbor rule, local fitness is ***consistent*** with global fitness, *iff* it is true that

$$\operatorname{sgn}\{F(s') - F(s)\} = \operatorname{sgn}\{\sum_{x \in X(s,s',i)} [f_{s'}(x) - f_s(x)]\} \tag{5}$$

where $\operatorname{sgn}\{\cdot\}$ is the symbolic function, and $X(s,s',i)$ denotes the set of entities whose states are changed as a result of the interacting effects of updating the state of entity i. If the neighbor rule is a reversible move class, and then $X(s,s',i) = X(s',s,i)$, the solution space can be represented as an undirected graph, otherwise a directed graph.

Definition 2. Local fitness is ***equivalent*** to global fitness *iff*

(i) local fitness is consistent with global fitness and,

(ii) $\exists \alpha \in R^+, \beta \in R$ such that it is true that for all $s \in S$,

$$F(s) = \alpha \sum_{x \in X} f_s(x) + \beta \tag{6}$$

According to Eq. (5), it is evident that if local fitness is consistent with global fitness, improving one or some entities' local fitness will also optimize the global fitness of the whole computational system, and the process of self-organization will be effective in solving hard computational problems. Equivalence is a special case of consistency. These definitions make it easier for us to understand what types of local fitness function should be defined in designing self-organized computing methods.

2.3. Microscopic analysis of optimal solutions

In a computational system, searching the optimal microstate will no doubt become simpler if we have a definite object in view. Therefore, in this section we will focus on the microscopic analysis of optimal solutions based on the preceding definitions of discrete state variable and local fitness function.

As we know, the optimal solutions for a large number of benchmark TSP problems have been given in Reinelt's TSPLIB (Reinelt 1991; TSPLIB). Therefore, for the sake of illustration and testing, we use the TSP instance, called kroA100 (100 cities) as an example. In order to find a specific TSP tour, we sequentially calculate the discrete state value and local fitness of each city. As a result, the microscopic characteristics of the optimal tour and a random tour can be found and have been comparatively shown in Fig. 1.

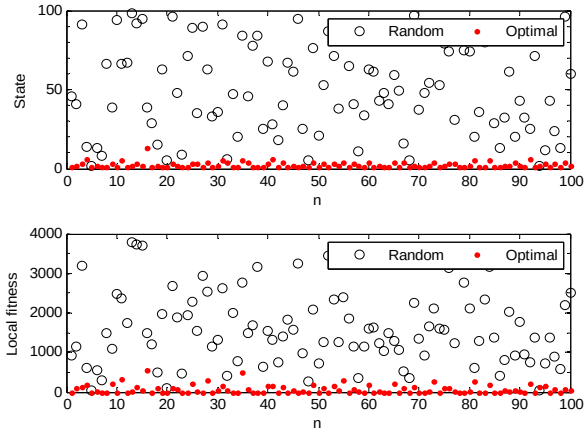

Fig. 1. Microscopic characteristics of the optimal tour and a random tour (kroA100).

As can be noted in Fig. 1, in the optimal solution the values of the state variables for all cities as well as their local fitness are far lower than the upper bound of the variables, e.g., $s(x_i) << n-1$ for all i ($1 \le i \le n$). This implies that those cities with high local fitness in an initial solution should update their states, until the computational system self-organizes itself into a well-organized microscopic state, i.e., the optimal solution.

In order to demonstrate the universality of this microscopic distribution, here we further characterize the statistical properties of the optimal TSP solutions by the kth nearest neighbor distributions (NNDs) (Chen and Zhang 2006). Without loss of generality, in this paper the kth-NND for any possible TSP tour s is defined as follows:

$$p(k) = \frac{r(k)}{n}, \quad k = 1,2,\cdots,n-1 \tag{7}$$

where $r(k)$ is the total number of the kth nearest neighbors for all forward-directed edges in a feasible tour, i.e., $r(k) = |X(k)|$ if $X(k) = \{x_i \mid s(x_i) = k\}$. Obviously, $p(k) \in [0, 1]$, and its sum will be equal to one. Fig. 2 shows the kth-NNDs $p(k)$ of optimal tours on a set of 10 benchmark Euclidean TSP instances, with sizes ranging from 51 to 2392 nodes.

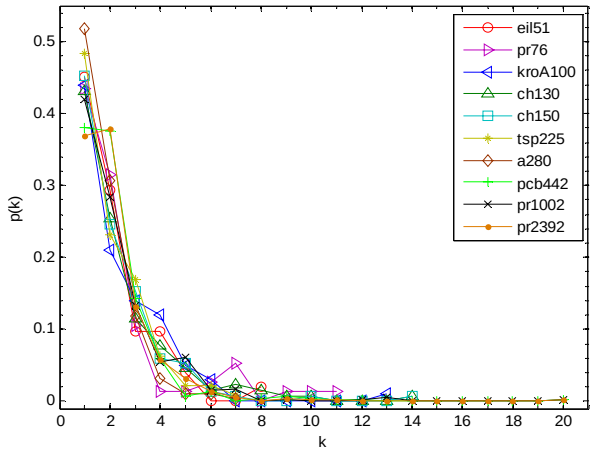

Fig. 2. The kth nearest neighbor distributions $p(k)$ of optimal tours from eil51 to pr2392.

As shown in Fig. 2, each kth-NND $p(k)$ is approximately an exponential decay function of neighboring rank k. Numerous experiments have indicated that the optimal solutions of almost all benchmark TSP instances in TSPLIB conform to this qualitative characterization of microscopic exponential distribution.

It is worth pointing out that the above analytical results of microscopic distributions are very useful both to mathematicians and to computer scientists. From the viewpoint of mathematics, the maximum neighboring rank $K_{\max} = \max\{s(x_i)\}$ for all i ($1 \le i \le n$) in the optimal solution is very useful to reduce the dimension of the state space. Taking pr2392 as an example, if we can approximately estimate it maximum neighboring rank $K_{\max} = 20$, the solution space represented by discrete state variables can be significantly reduced from $(n-1)^n$ to 20^n, and thus a low-dimensional state space can be obtained before being fed into mathematical tools. From the viewpoint of computer science, these analysis results are also of great value on the design of an effective self-organized computing method.

2.4. Neighborhood and fitness network

After unveiling the microscopic characteristics of optimal solutions, we can now further analyze the solution space of combinatorial optimization problems, and try to find a reasonable searching dynamics to reach the optimal solution. In a fundamental sense, the effectiveness of combinatorial optimization methods has a close relationship with the underlying neighborhood definition. The neighborhood mapping, also called move class, is usually used to construct better solutions from the current solution. For any possible solution $s \in S$, if $N(s) \subseteq S$ is defined as a neighborhood function, all those feasible solutions in $N(s)$ are called neighbors of the solution s. With respect to a specific neighborhood $N(s)$, we call s' a strict locally-optimal solution if the objective function $F(s') \le F(s)$, $\forall\ s \in N(s')$. Obviously, the globally-optimal solution s^*, $F(s^*) \le F(s)$, $\forall\ s \in S$, belongs to the set of locally-optimal solutions. If we can find out all the locally-optimal solutions, the global optimum will immediately obtained.

In order to analyze the computational complexity of search in a solution space, the concept of fitness landscape has been extensively applied to combinatorial optimization problems (Hordijk 1997; Altenberg 1997; Merz and Freisleben 2002; Reidys and Stadler 2002). Generally speaking,

if the dimension of search space is δ, i.e., the cardinality of neighborhood $|N(s)| = \delta$, by adding an extra dimension that represents the fitness of each solution, the dimension of fitness landscape will be equal to δ+1. Consequently, a hyper-dimensional landscape with peaks and valleys can be used to characterize the search space of optimization problems. Its ruggedness directly reflects the search complexity from an initial solution to the global optimum. If a fitness landscape is highly rugged, it will be difficult to search it for a better solution due to the lower correlation among neighboring solutions (Hordijk 1997; Merz 2004). To study the properties of fitness landscapes, Kauffman (1993) developed a problem-independent *NK*-model with a tunable parameter. In *NK*-model, the symbol *N* represents the number of entities of a computational system, e.g., cities in a TSP instance or genes in a genotype, and *K* reflects the degree of interactions among entities. More specifically, the local fitness contribution of each entity i is decided by the binary states of itself and K other interacting entities $\{i_1,\ldots, i_K\} \subset \{1,\ldots, i-1, i+1,\ldots, N\}$. Thus, the global fitness of a solution $s = x_1, \ldots, x_N$ can be mathematically represented as follows:

$$F(s) = \frac{1}{N}\sum_{i=1}^{N} f_i(x_i; x_{i_1},...,x_{i_K}) \tag{8}$$

The local fitness f_i of entity i depends on its value x_i and the values of K other entities $x_{i_1},...,x_{i_K}$. In simulation, the local fitness function f_i: $\{0, 1\}^{K+1} \rightarrow IR$ assigns a random number from a uniform distribution between 0 and 1 to each of its inputs (Merz 2004). With this *NK*-model, the ruggedness of fitness landscapes can be tuned from smooth to rugged by increasing the value of K from 0 to N–1. When K is small, the global fitness difference between neighboring solutions will be relatively small, and heuristics can easily find better solutions using correlated gradient information. When K is large, a large number of entities between neighboring solutions will possess different states that will greatly influence the global fitness. When $K = N$–1, the landscape will be completely random, and the local fitness of all entities will be changed, even if we update only one entity's state. In this extreme case, no algorithm is substantially more efficient than exhaustive search. However, in practice, almost all computational systems are probably not as complex as the worst case condition of $K = N$–1 in the *NK*-model, e.g., in the TSP the maximum neighboring rank $K_{\max} << N$. Therefore, it leaves us sufficient space to develop reasonable and efficient optimization methods, and hence to tackle those theoretically intractable computational problems.

Although the notion of fitness landscape is useful to characterize the complexity of search space, it is impossible to visualize a solution space when the dimension of neighborhood is higher than 2. Thus, in this paper, we utilize the concept of fitness network to represent the structure of solution spaces. For a given combinatorial optimization problem, the fitness network can be defined by a triple (S, F, N):

- a set of nodes (solutions) s on solution space S;
- a fitness function F: $s \rightarrow R$;
- a neighborhood function $N(s)$, which decides the structure of fitness network.

Thus, the fitness network can be interpreted as graph $G = (V, E)$ with vertex set $V = S$ and edge set $E = \{(s, s') \in S \times S \mid s' \in N(s)\}$. If the move class is a reversible, i.e., if $s' \in N(s)$ then also $s \in N(s')$, the fitness network is an undirected network; otherwise it is a directed network. On this complex network of search space, the out-degree of each nodes s is the cardinality of its

neighborhood $|N(s)|$.

Let us take the illustrative fitness network of Fig. 3 as an example. The hypothetical fitness network corresponds to a solution space with (0, 1) binary sequences of length 4. In this fitness network, we have the Hamming distances $d(s, s') = 1$ for all neighboring solutions s and s', the height in the vertical direction reflects the fitness value, i.e., the height of a node represents the fitness of the solution associated with it. Thus, the objective of optimization is to find the globally optimal solution '0011' in the bottom of the fitness network.

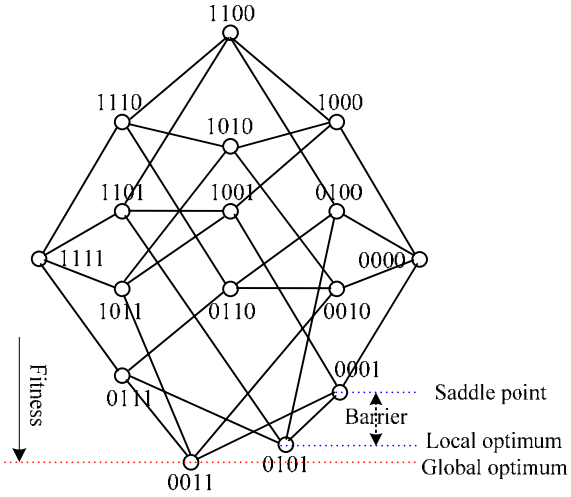

Fig. 3. A hypothetical fitness network on the Hamming graph of binary sequences of length 4.

Based on the representation of fitness network, the optimization dynamics can be described in terms of a searching trajectory navigating through the network in order to find the lowest-fitness node. Therefore, a neighborhood-based algorithm for combinatorial optimization can be essentially characterized by the neighborhood structure $N(s)$ and its searching dynamics in the fitness network.

In order to discuss whether or not an algorithmic solution can obtain the optimal solution from an arbitrary initial solution, let us first consider some properties of the fitness network.

Definition 3. In a fitness network, solution s' is ***reachable*** from s, if there exists at least a searching trajectory, $s = s_0, \ldots, s_k, s_{k+1}, \ldots, s_m = s'$, satisfying that $s_{k+1} \in N(s_k)$, $k = 0, \ldots, m-1$.

If a node is reachable from another node, then there exists a set of consecutive nodes along a sequence of edges between them. Also, if node s_2 is reachable from nodes s_1, and node s_3 is reachable from node s_2, then it follows that node s_3 is reachable from node s_1. The definition of reachability is applicable to both directed and undirected fitness networks.

Definition 4. A fitness network is called ***ergodic***, if all other solutions are reachable from an arbitrary solution s ($\forall s \in S$).

Here, the term "ergodic", as adopted from physics, refers to all microstates that are accessible. Theoretically speaking, a probabilistic searching method should guarantee that the optimal

solution is reachable from its initial solution s. Since we usually construct initial solutions by non-deterministic methods, the ergodicity of a fitness network is a necessary condition for finding the optimal solution.

In a fitness network as discussed above, local optima (such as solution '0101' in Fig. 3) are undoubtedly barriers on the way to the optimal solution. Suppose that solution s_1 and s_2 are two local optima, and S^t denotes the solution set in a reachable trajectory from s_1 to s_2, the fitness barrier separating s_1 and s_2, as discussed by Reidys and Stadler (2002), can be defined as follows:

$$F_b[s_1, s_2] = \min\{\max[F(s) \mid s \in S^t] \mid S^t : \text{reachable trajectories from } s_1 \text{ to } s_2\} \tag{9}$$

A critical point $s \in S$ satisfying the mini-max condition (9) is called as a saddle point of the fitness network (e.g., solution '0001' in Fig. 3). And then, the fitness barrier enclosing a local minimum s_1 can be evaluated by the height of the lowest saddle point between s_1 and a more favorable local minimum s_2 (Kern 1993; Reidys and Stadler 2002). Mathematically, it can be represented as follows:

$$F_b(s_1) = \min\{F[s_1, s_2] - F(s_1) \mid s_2 : F(s_2) < F(s_1)\} \tag{10}$$

This depth indirectly reflects the difficulty of escaping from the basins of attraction of the current local optimum s_1. As a rule of thumb, the quality of local optimal solutions will be improved, if we increase the size of neighborhood. It can also be reflected by Eqs. (9) and (10) that an increased set of reachable trajectories may also decrease the depth of a local optimum. However, a larger neighborhood does not necessarily produce a more effective optimization method due to computational complexity, unless one can search the larger neighborhood in a more efficient manner.

2.5. Computational complexity and "phase transition"

As mentioned before, NP-complete combinatorial optimization problems theoretically require exponential time to find a globally optimal solution. However, it is possible for us to solve a typical case in a more efficient manner. In order to characterize the typical-case computational complexity, the concept of phase transition in statistical physics has been successfully applied to the analysis of combinatorial optimization problems (Cheesman, Kanefsky, and Taylor 1991; Monasson 1999; Achlioptas, Naor, and Peres 2005). Phase transition refers to a drastic change in some macroscopic properties, when a control parameter crosses a critical value, e.g., water changing from ice (solid phase), to water (liquid phase), and to steam (gas phase) abruptly, as the temperature increases. Analogous to physical systems, it has been shown that many combinatorial decision problems also possess the phenomena of phase transition. Typical examples include: graph coloring problem (Cheesman, Kanefsky, and Taylor 1991), *K*-satisfiability problem (Martin, Monasson, and Zecchina 2001), traveling salesman problem (Gent and Walsh 1996; Zhang 2004), among others.

As a manifestation of phase transition theory in combinatorial optimization, here we discuss how the fundamentals of phase transition can be used to statistically analyze the solution space of combinatorial optimization problems. As an illustration, we take a random two-dimensional Euclidean TSP instance for example, with placing n = 10 cities on a square of area A, and enumerate the length L of all (n–1)!/2 possible tours. For each solution, we calculate its

dimensionless control parameter $L/\sqrt{n \cdot A}$ (Gent and Walsh 1996). Fig. 4 shows the phase transition on the probability distribution of all dimensionless control parameters. In the figure, the cumulative distribution function directly reflects the probability that the solution space takes on solutions less than or equal to a specific control parameter.

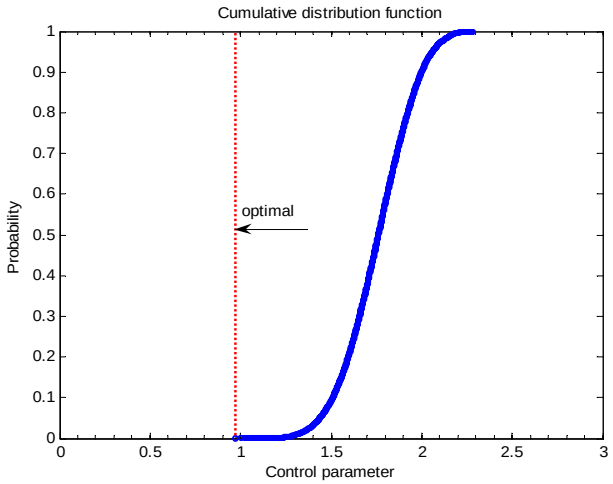

Fig. 4. Phase transition on the cumulative distribution of dimensionless control parameters.

Observing the phenomenon of phase transition as shown in Fig. 4, we can approximately divide the solution space into two regions: (i) one region is under-constrained among entities, in which the density of solutions is high, thus making us relatively easy to find a solution; (ii) another region is over-constrained, in which the probability of the existence of a solution is towards zero, and it is very difficult to obtain a near-optimal solution, especially the optimal solution, which is located on the boundary of phase transition. Since the cumulative distribution is analogous to Gaussian distribution, it means that most of candidate solutions locate in the middle-level of the fitness network.

Based on the above theoretical modeling, simulation, and analysis, we can note that the computational complexity of combinatorial optimization problems always occurs on the boundary of phase transition, and therefore a majority of solution samplings in a searching trajectory should focus on the bottom of the fitness network. A reasonable and effective optimization process for addressing hard combinatorial optimization problems should consist of two types of steps: one for finding a new-optimal solution as efficiently as possible; and another for repeatedly escaping from one local optimum to another with enough fluctuated explorations.

3. Self-organized computing

Following the line of analysis and discussions in the preceding section, in this section we present a self-organized computing method for solving combinatorial optimization problems. This optimization algorithm is inspired by the characteristics of collective emergent behavior in a dynamical system through the self-organization of its interacting entities.

3.1. Self-organized optimization algorithm

In recent years, self-organization has been recognized as one of the most important research themes in studying the evolving process of complex dynamical systems. In self-organizing systems, global behaviors seem to emerge spontaneously from local interactions among entities. A simple self-organization model was proposed by Bak and Sneppen (1993) for studying biological evolution. In the Bak-Sneppen model, the evolution of a multi-entity system is driven by its extremal dynamics. At each iterative step, the entity with the worst fitness is selected to update its state, and simultaneously, the states of their neighbors will also be changed due to the triggered local interactions. After a number of update steps, the local fitness of almost all entities will transcend a threshold value, and this system will reach a highly correlated critical state, in which a little change of one entity may result in chain reactions and re-organize major parts of the complex system (Bak 1996).

As we have analyzed in Sections 2.2 and 2.3, the combinatorial optimization solution can also be optimized by improving extremal local variables. Thus, the self-organized process driven by extremal dynamics can be used as an effective optimization dynamics. In the case of the TSP, the self-organized optimization algorithm under study can be given as follows:

Self-organized Optimization Algorithm

Step-1. Initialization

Randomly construct an initial TSP tour s, which can be naturally represented as a permutation of n cities (path representation). Let s_{best} represent the best solution found so far, and set $s_{best} \leftarrow s$.

Step-2. For the "current" solution s:

(i) Calculate the local fitness f_i according to Eq. (3) for each city i ($1 \le i \le n$);

(ii) Determine the order of all cities $\{\pi(1), \ldots, \pi(k), \ldots, \pi(n)\}$, in which $f_{\pi(1)} \ge \ldots \ge f_{\pi(k)} \ldots \ge f_{\pi(n)}$, i.e., the rank k is going from $k = 1$ for the city with the maximum local fitness to $k = n$ for the city with the minimum local fitness;

(iii) Select a city $\pi(k)$ with the kth "high" local fitness by a scale invariance probability distribution $P_k \propto k^{-\alpha}$ ($1 \le k \le n$), where α is an adjustable parameter;

(iv) Update the state of the selected city $\pi(k)$ by the 2-*opt* move class (Gutin and Punnen 2002), and then a neighborhood $N(s)$ of the current solution s is constructed;

(v) Calculate the global fitness $F(s)$ of all solutions in the neighborhood $N(s)$ and sequence all solutions by the descending order of global fitness. If the best solution s' in the neighborhood is better than the current solution, i.e., $F(s')-F(s) < 0$, accept $s \leftarrow s'$, and set $s_{best} \leftarrow s'$; Otherwise, selecting a neighbor s' having the lth "low" global fitness by another power-law distribution $P_l \propto l^{\beta}$ ($1 \le l \le |N(s)|$) to replace the current solution, where β is another power parameter, and $|N(s)|$ is the cardinality of the neighbor $N(s)$.

Step-3. Repeat at *Step-2* until the termination criterion is satisfied.

Step-4. Return s_{best} and $F(s_{best})$.

Generally speaking, the power parameter α is used to control the self-organized optimization process. For $\alpha = 0$, the updated entities are randomly selected, and for $\alpha \to \infty$, only the extremal entity with the worst local fitness is forced to change at each step. Numerous experiments have shown that a value of $\alpha \approx 1+1/\ln(n)$ seems to provide the best results on self-organized systems (Boettcher and Percus 2003; Chen, Lu, and Chen 2007). Another parameter β is used to adjust the hill-climbing ability on the basin of local optima. It is worth noting that calculating the global fitness of a solution at *Step-2* (v) can be done more efficiently by catching and computing those entities being updated rather than computing all of them from scratch. Usually, a predefined iterative number or *CPU* time can be selected as a termination criterion.

By applying the above proposed self-organized optimization algorithm to the TSP instance of kroA100, where the parameters $\alpha = 1.25$, $\beta = 2.75$, the typical searching dynamics of self-organized optimization algorithm has been plotted in Fig.5.

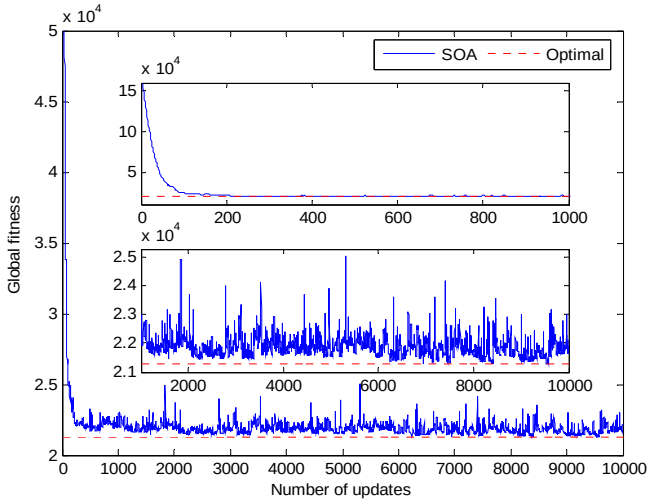

Fig. 5. The searching dynamics produced by the proposed self-organized optimization algorithm.

It can be seen from Fig. 5 that a near-optimal solution can be obtained sufficiently fast (as shown in the inset of Fig. 5) by the initial greedy self-organizing process, and then the enough fluctuated search near the global optimum can help to escape from the basin of the current local optimum and explore new local optima in the fitness network.

The extent of fluctuations, which reflects the ability of hill-climbing near the bottom of fitness network, can be adjusted by the parameter β. As we increase the value of β, the fluctuations become gradually reduced. While $\beta \to \infty$, there are no fluctuations, and the algorithm will directly converge to a local optimum. The experiments also show that the proposed algorithm provides superior performance when $\beta \approx 2.75 \pm 0.25$ for the example problems under simulation.

According to the analytical results in Section 2, the self-organized optimization dynamics combining the greedy self-organizing process and the fluctuated explorations is an effective search process in complex fitness networks. Furthermore, in order to validate the self-organized behavior of the proposed algorithm, we have randomly generated a large-scale Euclidean TSP instance, with $n = 2048$, and then statistically analyzed the optimized results of our optimization algorithm. The kth-NND of an optimized result is presented in Fig.6.

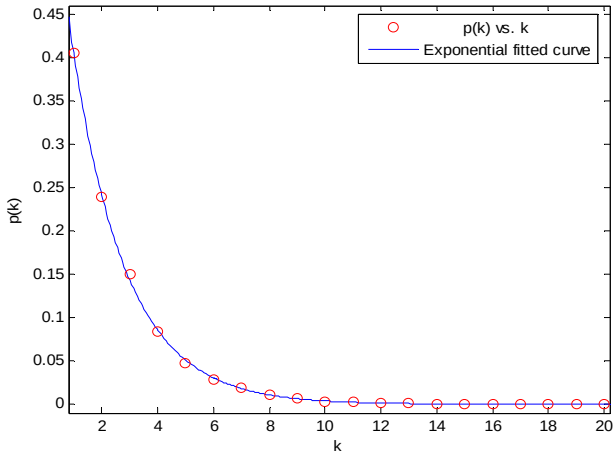

Fig. 6. Optimization result from the self-organized optimization algorithm.

As shown in Fig. 6, the kth-NND for the optimized result obtained by the self-organized optimization algorithm is a nearly perfect exponential distribution. In the exponentially fitted function $p(k) = ae^{-bk}$, the coefficients $a = 0.6791 \pm 0.0142$, $b = 0.5173 \pm 0.0115$, with 99% confidence bounds.

3.2. Comparison with related methods

In past few decades, different nature-inspired or stochastic methods have been proposed for tackling combinatorial optimization problems. In what follows we revisit three classical stochastic search algorithms for purposes of comparison and analysis.

– Simulated annealing

Simulated annealing (SA) is a stochastic search technique that simulates the thermo-dynamical process of annealing in metallurgy. At each step, the SA algorithm decides whether to replace the current solution according to the Boltzmann probability $exp(-\Delta E/T)$, where ΔE is the fitness difference between the current solution and a neighboring solution. When the control temperature T is large, a system configuration with worsening energy may be accepted with high probability. As the parameter is sufficiently close to zero by an exponential cooling schedule, the SA algorithm will no longer accept the uphill moves, and then an approximate solution is assumed to be obtained. Since numerous microstates are sampled far from the optimal solution, where the probability of solutions is relatively high, and a feasible solution is easy to be obtained, as discussed in Section 2.5, the SA optimization dynamics is probably not as efficient as the presently studied self-organized optimization dynamics.

– Genetic algorithm

Genetic algorithm (GA) is a population-based searching algorithm based on the evolutionary theory of natural selection. Generally speaking, the algorithm starts from a population of candidate solutions, and each individual is evaluated in terms of the global fitness. At each generation, a set of individuals are probabilistically selected from the current population according to their global fitness, and updated through crossover and mutation operators for creating a new population of

candidate solutions. As compared with our self-organized optimization algorithm, the GA optimization dynamics seems to be a purposeless generation-test process since it mimics evolution at the global system-level solution, and does not go deep into the microscopic entities of computational system.

– *Extremal optimization*

Although extremal optimization and our proposed algorithm are both inspired by the self-organized process of the Bak-Sneppen evolution model, two significant differences should be pointed out:

(i) The definition of local fitness differs. For example, in a TSP tour, the local fitness of city i is defined as $f_i = 3/(p+q)$, if it is connected to its p-th and q-th nearest neighbors, respectively (Boettcher and Percus 2000). Obviously, it is not consistent with the global fitness, as we have discussed, and the global optimum is hardly possible to be obtained.

(ii) The method for selecting a new solution s' to replace the current solution s is different. It is more like a physical process rather than optimization.

3.3. Experimental validation

In order to further validate the effectiveness and efficiency of the proposed self-organized optimization algorithm, we have selected a set of 10 benchmark Euclidean TSP instances, with sizes ranging from 51 to 2392 nodes, as the test benchmarks. They all are available from Reinelt's TSPLIB (Reinelt 1991).

In our computational tests, SA, τ-EO, and the proposed self-organized algorithm (SOA) use the same 2-*opt* move class at each iterative step. SA usually requires to tune several parameters case by case for obtaining a proper decreasing temperature schedule. The parameters of τ-EO and SOA are approximately set as: $\tau = \alpha \approx 1+1/\ln(n)$. GA employs the superior edge recombination crossover operator and displacement mutation operator as concluded in Refs. (Potvin 1996; Larrañaga et al. 1999). All these algorithms are encoded in C++ and implemented on Pentium IV (2.4 GHz).

Table 1 shows the comparative computational results. The mean value $\overline{F}$ and standard deviation σ of error(%)=100×($F(s)$–$F(s^*)$)/$F(s^*)$, where $F(s)$ denotes the solution found by a specific algorithm in each of 10 trials, are used to measure the performance and effectiveness of optimization algorithms. The computation time (t_{CPU}) is used to evaluate the efficiency of optimization algorithms.

Table 1 Comparison of the computational results from SA, GA, τ-EO and SOA.

Problem	$F(s^*)$	SA			GA			τ-EO			SOA		
		$\overline{F}$	σ	t_{CPU}	$\overline{F}$	σ	t_{CPU}	$\overline{F}$	σ	t_{CPU}	$\overline{F}$	σ	t_{CPU}
eil51	426	0.00	0.00	2	0.00	0.00	25	0.00	0.00	2	0.00	0.00	2
pr76	108159	0.21	0.24	3	0.00	0.00	30	0.00	0.00	3	0.00	0.00	3
kroA100	21282	0.69	0.45	5	0.05	0.07	65	0.12	0.09	5	0.02	0.04	5
ch130	6110	0.74	0.54	10	0.29	0.25	102	0.31	0.34	10	0.13	0.15	10
ch150	6528	0.92	0.61	12	0.38	0.40	149	0.51	0.42	12	0.22	0.20	12
tsp225	3916	1.42	0.77	21	0.66	0.35	212	0.65	0.49	21	0.45	0.34	21
a280	2579	1.45	0.71	45	0.64	0.51	362	0.53	0.46	45	0.32	0.31	45
pcb442	50778	2.07	1.05	125	1.53	0.62	1011	1.52	0.68	125	1.15	0.53	125
pr1002	259045	3.16	1.24	300	1.89	0.80	2957	1.95	0.76	300	1.77	0.69	300
pr2392	378032	4.45	2.23	900	3.83	1.51	5326	3.55	1.54	900	3.21	1.25	900

It can be seen from Table 1 that the proposed self-organized optimization method provides superior optimization performances in both effectiveness and efficiency, and outperforms the state-of-the-art simulated annealing, genetic algorithm, and extremal optimization for all 10 test instances.

4. Concluding remarks

The main contributions of this work are mainly twofold:

First, we have developed an analytic characterization for combinatorial optimization problems from the self-organizing system's point of view. Based on the definitions of discrete state variable and local fitness, in this paper we have discussed the empirical observation of the microscopic distribution with respect to an optimal solution. We have also introduced a notion of fitness network in order to characterize the structure of the solution space, and have statistically analyzed the searching complexity of hard optimization problems from the characteristics of phase transition.

Second, we have provided as well as demonstrated the performances of a self-organized optimization method for solving hard computational systems. For different optimization problems, only the definitions of a consist local fitness function and the neighboring rules are required.

Based on our analytic and algorithmic discussions, it is noted that the presented work offers new insights into, as well as may inspire, further studies on the systematic and microscopic analysis of NP-complete problems. The work paves the way for utilizing self-organization in solving combinatorial optimization problems. Our future work will involve in-depth studies on the optimization dynamics of computational systems by means of introducing the fundamentals of self-organizing systems (Liu and Tsui 2006).

Acknowledgements

The work reported in this paper has been supported in part by a HKBU Faculty Research Grant (FRG) (FRG/06-07/II-66) and in part by a Hong Kong Research Grant Council (RGC)

Central Allocation Group Research Grant (HKBU 1/05C).

References

Achlioptas, D., A. Naor, and Y. Peres. 2005. Rigorous Location of Phase Transitions in Hard Optimization Problems. Nature, 435: 759–764.

Altenberg, L. 1997. NK Fitness Landscapes. In Bäck, T., D.B. Fogel, and Z. Michalewicz, (Eds.) Handbook of Evolutionary Computation, Oxford University Press, New York.

Bak, P. 1996. How Nature Works: The Science of Self-Organized Criticality. Copernicus Press, New York.

Bak, P., and K. Sneppen. 1993. Punctuated Equilibrium and Criticality in a Simple Model of Evolution. Physical Review Letters, 71(24): 4083–4086.

Blum, C., and A. Roli. 2003. Metaheuristics in Combinatorial Optimization: Overview and Conceptual Comparison. ACM Computing Surveys, 35(3): 268–308.

Boettcher, S. 2000. Extremal Optimization: Heuristics via Co-Evolutionary Avalanches. Computing in Science & Engineering, 2(6): 75–82.

Boettcher, S., and A.G. Percus. 2003. Optimization with Extremal Dynamics. Complexity, 8: 57–62.

Boettcher, S., and A.G. Percus. 2000. Nature's Way of Optimizing. Artificial Intelligence, 119: 275–286.

Bonabeau, E., M. Dorigo, and G. Theraulaz. 2000. Inspiration for Optimization from Social Insect Behaviour. Nature, 406: 39–42.

Cheesman, P., B. Kanefsky, and W.M. Taylor. 1991. Where the Really Hard Problems Are. Proc. IJCAI'91, pp. 163–169.

Chen, Y.W., and Y.Z. Lu. 2007. Gene Optimization: Computational Intelligence from the Natures and Micro-mechanisms of Hard Computational Systems. LSMS 2007, Lecture Notes in Computer Science, 4688: 182–190.

Chen, Y.W., Y.Z. Lu, and P. Chen. 2007. Optimization with Extremal Dynamics for the Traveling Salesman Problem. Physica A, 385: 115–123.

Chen, Y.W., and P. Zhang. 2006. Optimized Annealing of Traveling Salesman Problem from the nth-Nearest-Neighbor Distribution. Physica A, 371: 627–632.

Cormen, T.H., C.E. Leiserson, R.L. Rivest, and C. Stein. 2001. Introduction to Algorithms. MIT Press.

Duch, J., and A. Arenas. 2005. Community Detection in Complex Networks using Extremal Optimization. Physical Review E, 72: 27104.

Forrest, S. 1993. Genetic Algorithms: Principles of Natural Selection Applied to Computation. Science, 261(5123): 872–878.

Garey, M.R. and D.S. Jonhson. 1979. Computers and Intractability: A Guide to the Theory of NP-Completeness. W. H. Freeman, New York.

Gent, I.P., and T. Walsh. 1996. The TSP Phase Transition. Artificial Intelligence, 88: 349–358.

Goles, E., M. Latapy, C. Magnien, M. Morvan, H.D. Phan. 2004. Sandpile Models and Lattices: A Comprehensive Survey. Theoretical Computer Science, 322: 383–407.

Gutin, G., and A.P. Punnen. 2002. The Traveling Salesman Problem and Its Variations. Kluwer Academic Publishers.

Han, J. 2003. Local Evaluation Functions and Global Evaluation Functions for Computational Evolution. DOI: SFI-WP 03-09-048.

Han, J., and Q.S. Cai. 2003. Emergence from Local Evaluation Function. Journal of Systems Science and Complexity, 16(3): 372–390.

Hordijk, W. 1997. A Measure of Landscapes. Evolutionary Computation, 4(4): 335–360.

Kauffman, S.A. 1993. The Origins of Order: Self-Organization and Selection in Evolution. Oxford University Press.

Kennedy, J., and R. Eberhart. 1995. Particle Swarm Optimization. IEEE International Conference on Neural Networks, 4: 1942–1948.

Kern, W. 1993. On the Depth of Combinatorial Optimization Problems, Discrete Applied Mathematics, 43(2): 115–129.

Kirkpatrick, S., C.D. Gelatt Jr., and M.P. Vecchi. 1983. Optimization by Simulated Annealing. Science, 220(4598): 671–680.

Larraňaga, P., C.M.H. Kuijpers, R.H. Murga, I. Inza, and S. Dizdarevic. 1999. Genetic Algorithms for the Travelling Salesman Problem: A Review of Representations and Operators. Artificial Intelligence Review, 13: 129–170.

Liu, J., J. Han, and Y.Y. Tang. 2002. Multi-Agent Oriented Constraint Satisfaction. Artificial Intelligence, 136: 101–144.

Liu, J., X. Jin, and K.C. Tsui. 2005. Autonomy Oriented Computing: From Problem Solving to Complex Systems Modeling. Kluwer Academic Publishers.

Liu, J., Y.Y. Tang, and Y. C. Cao. 1997. An Evolutionary Autonomous Agents Approach to Image Feature Extraction. IEEE Trans. on Evolutionary Computation, 1(2): 141–158.

Liu, J., and K.C. Tsui. 2006. Toward Nature-Inspired Computing. Communications of the ACM, 49(10): 59 – 64.

Martin, O.C., R. Monasson, and R. Zecchina. 2001. Statistical Mechanics Methods and Phase Transitions in Optimization Problems. Theoretical Computer Science, 265: 3–67.

Merz, P. 2004. Advanced Fitness Landscape Analysis and the Performance of Memetic Algorithms. Evolutionary Computation, 12(3): 303–325.

Merz, P., and B. Freisleben. 2000. Fitness Landscape Analysis and Memetic Algorithms for the Quadratic Assignment Problem. IEEE Trans. on Evolutionary Computation, 4(4): 337–352.

Mézard, M., G. Parisi, and R. Zecchina. 2002. Analytic and Algorithmic Solution of Random Satisfiability Problems. Science, 297: 812–815.

Middleton, A.A. 2004. Improved Extremal Optimization for the Ising Spin Glass. Physical Review E, 69: 055701(R).

Monasson, R., R. Zecchina, S. Kirkpatrick, B. Selman, and L. Troyanskyk. 1999. Determining Computational Complexity from Characteristic 'Phase Transitions'. Nature, 400: 133–137.

Papadimitriou, C.H. 1994. Computational Complexity. 1st edition, Addison Wesley.

Papadimitriou, C.H., and K. Steiglitz. 1998. Combinatorial Optimization: Algorithms and Complexity. Courier Dover Publications.

Potvin, J.Y. 1996. Genetic Algorithms for the Traveling Salesman Problem, Annals of Operations

Research, 63: 339–370.

Reidys, C.M., and P.F. Stadler. 2002. Combinatorial Landscapes. SIAM Review, 44(1): 3–54.

Reinelt, G. 1991. TSPLIB-a Traveling Salesman Problem Library. ORSA Journal on Computing 3(4): 376–384.

Sneppen, K. 1993. Extremal Dynamics and Punctuated Co-Evolution. Physica A, 221: 168–179.

TSPLIB, http://www.iwr.uni-heidelberg.de/groups/comopt/software/TSPLIB95/.

Weinberger, E.D. (1996). NP Completeness of Kauffman's N-k Model, A Tuneable Rugged Fitness Landscape. Technical Report 96-02-003, Santa Fe Institute.

Yang, B., and J. Liu. 2007. An Autonomy Oriented Computing (AOC) Approach to Distributed Network Community Mining. Proc. 1st International Conference on Self-Adaptive and Self-Organizing Systems, pp. 151–160.

Zhang, W. 2004. Phase Transitions and Backbones of the Asymmetric Traveling Salesman Problem. Journal of Artificial Intelligence Research, 21: 471–497.