Artif Intell Rev (2012) 38:313–324
DOI 10.1007/s10462-011-9256-4

Toward understanding the optimization of complex systems

Jiming Liu · Yu-Wang Chen

Published online: 22 June 2011

Abstract In response to the increasing demands for solving various optimization problems arisen from complex systems, the paper focuses on the study of a general-purpose distributed/decentralized self-organized computing method, with the help of agent-oriented modeling method. The goal of this paper is to unravel the microscopic characteristics and the hidden complexity of complex computational systems, and furthermore, construct proper mechanisms that will support self-organized computation.

Keywords Complex systems · Optimization · Self-organized computation · Complexity · Agent-oriented modeling

1 Introduction

Large-scale, highly-distributed optimization problems are pervasive in the field of complex systems, such as optimal path routing in public transportation systems, planning and scheduling in manufacturing systems, microarray gene ordering in biological genomics, search optimization in social networks. Traditionally, to a specific optimization problem, a well-defined mathematical model is indispensable for finding the optimal solution of extremizing the functional objective. However, in complex systems there exist an enormous number of interacting and dynamically-evolving entities, modules, or subsystems, and so their

The abstract of this paper was accepted for the poster session at the European Conference on Complex Systems (ECCS) 2009.

J. Liu · Y.-W. Chen (✉)
Department of Computer Science, Hong Kong Baptist University, Kowloon Tong, Hong Kong
e-mail: ywchen@comp.hkbu.edu.hk

J. Liu
e-mail: jiming@comp.hkbu.edu.hk

Y.-W. Chen
Manchester Business School, The University of Manchester, Manchester M15 6PB, UK

optimization requirements are hardly possible to be satisfied by strict mathematical tools or centralized optimization methods. Consequently, in past decades computer scientists have developed a series of autonomic, distributed and nature inspired computing techniques for solving specific optimization problems in complex systems. For example, Autonomy-Oriented Computing (AOC) (Liu et al. 2005) could collectively detect certain image features by numerous autonomous imaging entities; Google's PageRank (Brin and Page 1998) used a distributed search engine, and could return the searching results in a few seconds from billions of websites; Message-passing method (Frey and Dueck 2007) could efficiently cluster the data points by affinity propagation. Generally speaking, all these computing methods implicitly utilize the locally interacting mechanisms and the collective characteristics in tackling the optimization requirements of complex systems.

In response to the increasing demands for solving various optimization problems arisen from complex systems, in this paper we will try to build a fundamental understanding of the optimization of complex systems. The ultimate motivations of this study are two-fold: (1) to unravel the microscopic mechanism and the hidden complexity of computational systems; (2) to explore a general-purpose distributed/decentralized self-organized computing architecture for large-scale optimization problems.

This structure of this paper is organized as follows: In Sect. 2, we propose the agent-oriented modeling method for complex computational systems. We then apply a hierarchical framework to analyze the computational complexity in Sect. 3. In Sect. 4, we present a self-organized computing architecture, with comparative and analytical studies. In Sect. 5, we conclude this paper.

2 Agent-oriented modeling of complex computational systems

For characterizing the optimization requirements, the agent-oriented modeling method has become very popular in complex systems.

2.1 Autonomous agents

Agent-oriented methods have been extensively applied to model various large-scale distributed complex computational systems, such as human systems (Bonabeau 2002), economic systems (Samanidou et al. 2007), software engineering systems (Jennings and Bussmann 2003), etc. In virtue of agent-oriented modeling methods, a computational system A can be modeled as a set of autonomous entities called agent. Consequently, the system state can be formulated by the microscopic state of the agents, and the dynamics of system evolution can be described by the interactions between agents. Here, the notion of agent is an abstract concept, which can be considered as decision variable, constraints, modules, and even if problem-specific sub-subsystems. For example, the computing agents can represent the discrete variables called *spins* in the Ising model (McCoy and Tsun Wu 1973), the variables or clauses in constraints satisfaction systems (Tsang 1993), the nodes or communities in complex networks (Strogatz 2001; Newman 2003). Generally, the agents, like its biological counterpart, can collect local (limited) information and assess its environment. At the same time, each of them can compete or cooperate with neighboring agents for achieving individual or global objectives, and then make reasonable actions on the basis of a set of behavioral rules. More specifically, an autonomous computing agent a can be characterized with the following basic properties:

- a finite number of possible states S_a;
- local fitness function $f(a)$ for evaluating its current state and environmental conditions;
- behavioral rule repository R_a.

Usually, the autonomous agents in most of computational systems are considered as homogenous, and their state space can be represented with consistent domain of definitions. However, in more general cases, the agents are heterogeneous and each class of them should be defined, respectively. In the process of evolution, each agent needs to build a local objective function for evaluating the pressure of updating its transient state. The local evaluator can collect local information, including the current state of the agent, the states of ***neighbors*** and the incomplete information of ***environment***, and then guide the agent to take specific behaviors from its behavioral rule repository.

After defined the basic elements, a complex computational system can now be described as:

- a group of autonomous agents $A = \{a_1, a_2, \ldots, a_n\}$;
- interactions between agents, which implicitly affect global patterns at the system level;
- a set of reactive rules, governing the interactions among agents.

As a result, the solution of computational systems can be formulated as the ***state vector*** of all agents $s = \{s_1, s_2, \ldots, s_n\}$. Since the optimization solution is usually a nontrivial combination of a large number of locally constrained variables, the distributed nature of agent-oriented model may provide the possibility of solving the computing problems by local behaviors of agents.

2.2 Interactions among agents

In agent-oriented computational systems, agents are not isolated from each other. The behaviors taken by each agent are usually influenced by its neighboring agents, and the global patterns are emerged from those local interactions. Commonly, there exist two types of interactions, direct interactions and indirect interactions. Direct interactions could occur directly between any pair of agents. Indirect interactions could only be achieved through the long-range aggregated effects or nonlinear feedback between agents and environments, and the shared environments will serve as the media for indirect interactions (Liu 2001, 2008).

Usually, different complex systems have different types of direct and indirect interactions. For the sake of explanation, here we take the Ising computing model as an example. Suppose that the state vector $\sigma = \{\sigma_1, \sigma_2, \ldots, \sigma_n\}$ represent the assignment of the Ising spins, in which each state variable takes on the value of 1 or -1, and so the energy function of the Ising computing model can be represented as:

$$F(\sigma) = \sum_{i=1}^{n} h_i \sigma_i + \sum_{i=1}^{n} \sum_{j=1}^{n} J_{ij} \sigma_i \sigma_j \tag{1}$$

The first summation is the biased energy excitated by magnetic fields. The second is the interacting energy, which can be calculated over all edges in complete graph G. J_{ij} is the coupling coefficient. As the discussion of the Ising model, we find that direct interactions can be represented by the ***interaction matrix*** $J = \{J_{ij}\}$. However, indirect interactions are usually difficult to be explicitly expressed in objective function or constraints.

2.3 Local-to-global mapping

Having described the basic elements and bonds of agent-oriented computational systems, namely, agents and interactions, in this section we will develop several general guidelines and principles for designing local self-organized mechanisms that can realize a desired global optimization objective.

2.3.1 Local fitness function

In self-organized computational systems, inputs are concerned with locally interacted agents and a specific microscopic configuration. Outputs refer to the desired global solution or the macroscopic behavioral patterns. When deploying emergent computing mechanisms to manipulate the input agents and generate the desired microscopic configuration, a local fitness function should be developed firstly, and through it each agent can assess its current state and sense the environments, then select behaviors and perform interactions (Liu et al. 2002). However, a natural question arising is which types of local fitness function are capable of guiding the computational systems to evolve toward the global optimum. In order to answer this question, let us firstly take a view on an elementary case in Fig. 1, which is abstracted from the fitness landscape of epistasis (Poelwijk et al. 2007).

In Fig. 1, we try to illustrate the requirements of building local fitness function with three tentative fitness landscapes, in which each path evolves from the initial solution ‘ab’ to the global optimum ‘AB’ through updating each agent once, and the solid paths are regarded as the favored evolutionary paths. For selecting the favored trajectories, Fig. 1a indicates that local fitness function for each agent should reflect the sign of the global fitness changes, seemingly, in the intermediate process updating ‘a’ to ‘A’ yields the opposite fitness effect for updating ‘b’ to ‘B’; Fig. 1b, c show that local fitness function should be capable of measuring the magnitude of the global fitness changes, e.g., updating ‘a’ to ‘A’ has more positive or less negative fitness effect than updating ‘b’ to ‘B’.

With these two premises, we can define reasonable local fitness functions for different computational systems. For example, in the Ising computing model the local fitness of the spin σ_i can be represented as:

$$f_\sigma(\sigma_i) = \sigma_i(h_i + \sum_{j \in N(i)} J_{ij}\sigma_j) \tag{2}$$

where $N(i)$ is the set of spins directly interacted with the spin σ_i. And then, the global fitness changes can be consistently calculated by the local fitness function when flipping the state of spins. With the definition of the consistent local fitness function, the emergent computing methods can now know how to apply the local behavioral rules in the presence of natural or priority-based selection mechanisms.

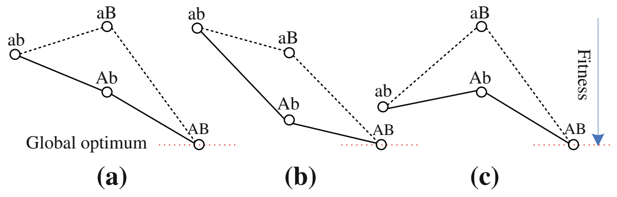

Fig. 1 Illustration of accessible evolutionary paths

2.3.2 Local update rules of emergent computation

In order to self-organize the agent-oriented computational system and achieve desired macroscopic patterns, the agents need to be locally regulated according to their local fitness. In the field of traditional combinatorial optimization, the local update rules are usually regarded as neighboring operation (Ahuja et al. 2002). Theoretically, the updates must make any two macroscopic states reachable in the fitness network of solution space, that is to say, the computational system should be ergodic if we want to find the optimal solution. However, agent-oriented computational systems are usually non-ergodic due to the biased selection mechanisms. In practice almost all nature-inspired optimization methods are committed to providing a non-ergodic searching process for efficiently finding out an optimal or near-optimal solution. In addition, because updating those selected agents will also affect their neighboring agents, the long-range stochasticity and historically aggregated effects can create sufficiently diverse microscopic configurations.

3 Hierarchical analysis of computational complexity

The term "complexity" has different meanings in different disciplines, e.g., statistical physics, computational biology, complex systems, etc. In agent-oriented computational systems, complexity is usually used to characterize the degree of difficulty in predicting the macroscopic properties of the system. In the paper, we try to illuminate the hidden computational complexity at two different levels: microscopic level and macroscopic (system) level.

3.1 The characteristics of microscopic complexity

At the microscopic level, we can say that a system is complex if it consists of a number of interacting components, and the global behavior is difficult to deduce from the local rules of entities. For example, metal has macroscopic properties, such temperature, conductivity, etc., and these macroscopic properties may emerge from the microscopic interactions of atoms. As the interactions increases, the states of each entity become more dependent on the states of those interrelated entities, making it difficult to understand the internal mechanism. In this section, we try to explain the complexity of computational systems by the complexity of agents and interactions. That is to say, the increase in the complexity of computational system may be characterized by a growth in the dimension of systems and the intensity of local interactions.

- The dimension of systems

In computational systems, the dimension of systems can be reflected by the number of autonomous entities. For example, a cellular automata is more complex if it has more automaton, and its computation needs more space and time complexity (Mitchell 1998).

Usually, most conventional theories with the mathematical insights of measuring complexity depend on the dimension of computational systems (Mertens 2002). For example, in *NP*-hard theory all problems are regarded as tractable if their computational complexity in the worst case grows polynomially with the dimension of systems. Contrarily, those problems can only be solved by algorithms with non-polynomial time complexity, such as $O(2^n)$ or $O(n!)$, are called intractable. In last century the complexity theory mainly focused on this worst-case analysis. However, the theory of phase transition recently developed by computer scientists shows that the optimization problem with the same dimension will exhibit a drastic change on the computational complexity (Cheesman et al. 1991).

- The complexity of entities

Firstly, the complexity of entities is directly affected by their state space. For example, a Boolean cellular automata is less complex than a multi-valued one, in which each automaton has more possible states (Mitchell 2006). Secondly, the complexity of entities can be measured by the distribution of microscopic properties, such as the scale-free degree distribution in complex network (Barabási and Albert 1999), the exponential kth nearest neighbor distribution in random computing systems (Liu et al. 2008), etc. Here, let us consider the K-satisfiability problem (K-SAT), which is the first optimization problem to be proved *NP*-hard. The goal of K-SAT is to find a solution that simultaneously satisfies a set of M constraints between N decision variables $x \in \{x_1, x_2, \ldots, x_N\}$. Generally, each constraint is represented as a logical *or* clause of K variables or its negations (Mézard et al. 2002). In naturally-generated paradigm, each clause is composed of K randomly-selected variables. Correspondingly, each variable would be connected on average $K \times M/N$ clauses, whereas some variable has more and some has less clauses. The generic structure of K-SAT problems was usually represented as factor graphs (Mézard and Zecchina 2002). As a result, the degree distribution of variable nodes, i.e., the probability distribution of the number of clauses connected to variables, can be represented as:

$$P(c = \xi) = \binom{M}{\xi} \left(\frac{K}{N}\right)^{\xi} \left(1 - \frac{K}{N}\right)^{M-\xi} \tag{3}$$

Obviously, if a variable was connected to only one clause or connected to multiple clauses with the same interactions (positive or negative), its value can be easily decided in the optimal solution, and these backbone variables could remain unchanged during optimization. However, if a variable was connected to multiple clauses with the different interactions, its value is extremely hard to be decided, especially for those variables with relatively high degree in the Poisson distribution. Perturbing the state of these nodes may bring avalanched changes in the microscopic configurations. Similarly, the complexity in a scale-free network may have a direct relationship with a few highly connected hubs in the power-law degree distribution (Albert and Barabási 2007). In addition, we can also deduce that different microscopic distributions result in the phenomenon of phase transition in computational complexity.

- The complexity of interactions

Mathematically, the interactions of computational systems can be represented as the interaction matrix, from which we can also analyze the complexity of interactions. The complexity of interactions is usually having direct relationship with two factors. The first factor is the extensity of interactions, which is less extensive if the interaction matrix is sparser. For example, the Ising model with periodic boundary, in which couplings only exist between the spins σ_i and σ_{i+1}, can be easily solved. The second is the heterogeneity of entities. The computational system is more complex if interactions are more heterogeneous. The extremely simple case is the computational system with totally identical interactions, and it can be easily solved by statistical analysis methods. In statistical physics, the heterogeneous computational systems are usually reduced by mean-field method (Opper and Saad 2001), in which various interactions are replaced by average or effective interaction. In past decades, computer and physical scientists have built various models to explain the complexity of interactions. For example, Kaufman' NK model, in which N represents the number of elements of the system and K reflect the degree of interactions, has been extensively used to study the relationship between the interaction of elements and the complexity of systems (Kauffman 1993). Theoretical analysis indicated that the microscopic parameter K directly reflects the macroscopic complexity of systems (Hordijk 1997; Merz 2004). The small K corresponds to gradual and smooth variations in fitness landscapes. It implies that the neighboring solutions have high autocorrelation, and near-optimal or optimal solutions tend to cluster in the same region. With the increase of K, the interactions make it difficult to decompose the computational system. In the extremely complex condition of $K = N - 1$, all those entities are mutually entangled with nonlinear interactions, and no algorithm will be more efficient than exhaustive search. However, in practice the complex systems are not as complex as the worst case due to the locality of entities.

3.2 Locality of computing agents

The locality and modularity are typical features of complex systems (Variano et al. 2004; Newman 2006). As illustrated by Fig. 2, in agent-oriented computational systems different agents may also have a localized control zone.

In Fig. 2, let us assume ρ_i to be the control radius of the agent a_i, and then we only need consider the set of agents fallen into its spatial control zone, rather than all the other agents, when updating the state of the agent a_i. The set of neighboring agents can be represented as:

$$N(a_i, \rho_i) = \{a : |a - a_i| \le \rho_i\} \tag{4}$$

In heterogeneous computational systems, different agents have different influencing abilities. The larger control radius is, the harder it is to decide its state during optimization. However, the agents in real-world computational systems usually have relatively small control space compared with the dimensionality of systems. To invalidate this summing-up, here we take a TSP computing paradigm, called kroA100 (100 entities) (Reinelt 1991), as an example. The locality of computing agent in the optimal solution can be represented by Fig. 3.

Due to the overlaps of control zones, the computational complexity will fall on the coordination efforts among local control agents, in which each agent will compete and cooperate with neighbors for achieving the balance between selfishness and altruism.

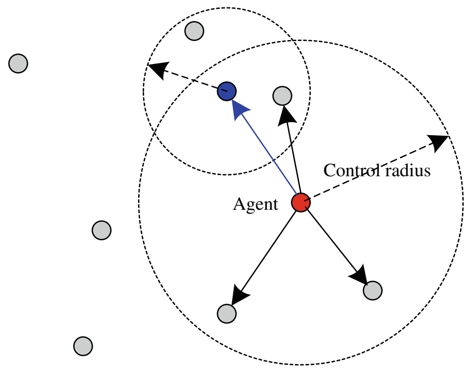

Fig. 2 Illustration of locality of optimizing agents

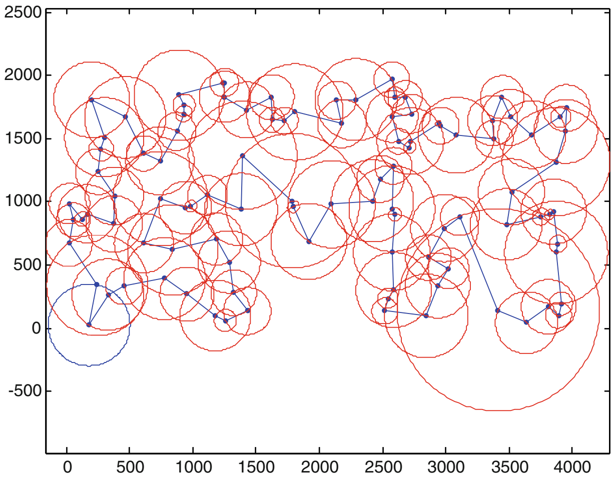

Fig. 3 Locality of computing agents in the TSP computing paradigm kroA100

3.3 Macroscopic fitness network

Traditionally, most literatures applied the notion of fitness landscape to explain the complexity of solution space (Monasson et al. 1999; Achlioptas et al. 2005). However, the fitness landscape is hardly possible to characterize the distribution of solutions in configuration space. Here, we introduce the concept of fitness network. Figure 4 illustrates how to transfer fitness landscape to fitness network by a solution space with (0, 1) binary sequences of length 3. In the fitness network, the Hamming distance between all neighboring solutions is equal to 1, and the height of a node represents the fitness of the solution associated with it.

With the definition of fitness network, we can easily analyze the most reasonable searching trajectory through the architecture of networks. As discussed in our earlier work (Liu et al. 2008), in the fitness work an efficient searching process should consist of the two steps: (1) depth-first search for finding a new-optimal solution efficiently; (2) width-first search for repeatedly escaping from one local optimum to another.

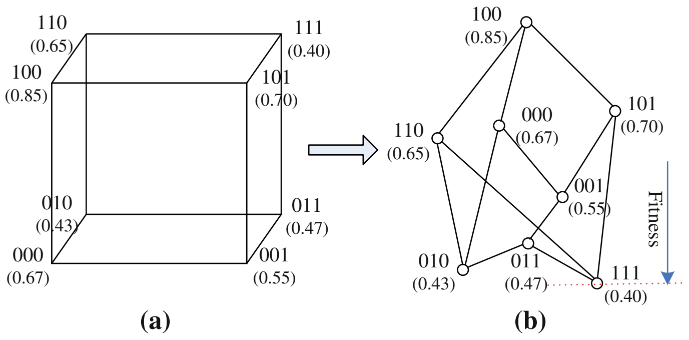

Fig. 4 An example of a fitness network for binary sequence of length 3

4 Solving agent-oriented computational systems

If we want to solve an agent-oriented computational system, it is useful to describe it as self-organized. In the self-organized computational systems, agents are designed to autonomously and dynamically perform their behavior and interactions which will lead to the system function.

4.1 The architecture of self-organized computing

In self-organized systems, computing agents sense their states and environments, select behaviors, and perform interactions, until the system A collectively achieves the desired emergent solution.

The basic architecture of self-organized computing can be stated as follows:

1. model the agent-oriented computational system A, and define the local evaluation function and local update rules for agents;
2. ***while*** the stopping criteria (e.g., the maximum number of updates, the maximum CPU time, or desired macroscopic patterns) are satisfied, ***Do*** the self-organized computing by iteratively updating the states of agents:
   (a) assess the current state of each agent by the local evaluation function;
   (b) select some agents according to their local fitness;
   (c) apply behavioral rules in those selected agents;
   (d) accept a new macroscopic state, based on the collectively global aggregated effects.

It is worth noting that the in step 2(b) different selection mechanisms can be selected as the optimization dynamics of self-organized computing. For example, extremal dynamics had been successfully applied in self-organized computational systems (Boettcher and Percus 2000; Duch and Arenas 2005). In the optimization dynamics, the agent with the k-th "worst" local fitness will be updated according to the following probability:

$$p_k = Ck^{-\lambda}(1 \le k \le n) \tag{5}$$

In the large-scale computational systems, the constant C can be approximated by the normalization expression:

$$\lim_{n\to\infty}\sum_{k=1}^{n} p_k = C \cdot \lim_{n\to\infty}\sum_{k=1}^{n} k^{-\lambda} = C\zeta(\lambda) = 1 \tag{6}$$

where $\zeta(\alpha)$ is the Riemann Zeta function. In natural systems, the parameter $\lambda \approx 1 + 1/\ln(n)$ seems to provide the better performance than others (Boettcher and Percus 2003; Chen and Zhang 2006).

4.2 Computational paradigms

In this section, we try to illustrate the self-organized computing method with two simple computational paradigms.

Paradigm 1: As the optimization counterpart of K-SAT, *Max*-SAT is to maximize the number of satisfied clauses, or minimize the number of unsatisfied clauses (Schuurmans and Southey 2001). If we model the logical variables as computing agents, the local fitness function of variable i can be defined as the difference between the number of unsatisfied clauses

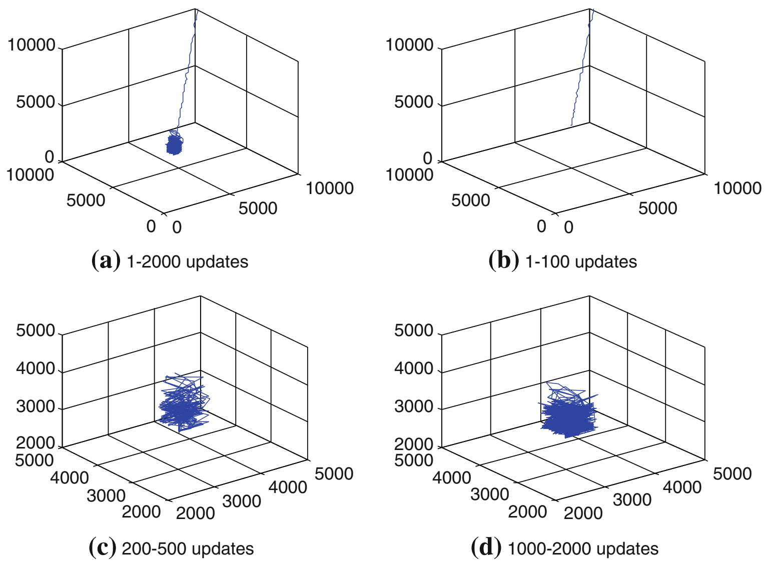

Fig. 5 Searching trajectory of self-organized computation

and the number of satisfied clauses connected to the variable node. We then can select the variables according to the local fitness, and flip the value of the selected variables to try to satisfy more constraints. Repeat the selection and flip operation until a satisfying assignment is found. Simulation results indicate that the computing method can provide superior performance in both effectiveness and efficiency.

Paradigm 2: The traveling salesman problem (TSP) (Gutin and Punnen 2002), one of the most typical complex computing problems, is to find the shortest path among a set of spatially-distributed nodes. Let us assume d_{ij} to be the distance between node i and j. Given a feasible solution s, the local fitness function of node i can be defined as:

$$f_s(x_i) = d_i - \min_{j \neq i} d_{ij}, \quad i = 1, 2, \ldots, \quad n \tag{7}$$

where d_i represents the length of the forward-directed edge starting from node i in the current microscopic configuration. Furthermore, the *k-opt* move class (Gutin and Punnen 2002) is applied to update the states of those agents selected by extemal dynamics. Applying the self-organized computing method to an optimization instance ftv170 (Reinelt 1991), a typical searching trajectory (see Fig. 5) can be constructed by the 3-dimensional time-delay coordinates (Li and Alidaee 2002).

It can be seen from Fig. 5 that the initial period of self-organized process is a gradual depth-first search. After reached a near-optimal solution, the searching trajectory exhibits a chaotic behavior for finding the optimal microscopic configuration. With the simulation of Cellular Automata, Langton presented complex living systems may come into being at the edge between order and chaos (Langton 1995). Kauffman suggested that the maximum fitness of complex systems may be obtained at the edges of chaos (Kauffman 1993). As we discussed earlier, this self-organized optimization dynamics can be regarded as an effective search process in complex fitness networks.

4.3 Comparisons and analysis

In past decades, we have witnessed a booming development on nature-inspired computing techniques. However, most of those optimization methods neglected the inevitable relationship between the microscopic characteristics of computational systems and computing mechanisms. For example, evolutionary algorithms seem to be a purposeless generation-test evolutionary process since it only mimics evolution at the level of macroscopic solutions. In ant colony, although there is also no central control, the social insects are not the counterparts of the components of complex computational systems. Artificial life is mainly introduced to simulate man-made systems that exhibit the behavioral characteristics of living organisms. However, self-organized computation studied in this paper can build the intrinsic relationships between problem characteristics and computing method. In addition, the algorithm only use local information and local update rules, thus it has relatively low complexity, and is convenient to be parallelized in programming. In addition, the distributed nature can play an important role in the robustness of computation.

5 Concluding remarks

"*We can only see a short distance ahead, but we can see plenty there that needs to be done*"

Alan M.Turing

In this paper, we try to build a fundamental understanding of the optimization of complex systems, with the help of self-organization theory. Specifically, the paper proposed an agent-oriented computing model and a series theoretical analysis to construct proper mechanisms that will support self-organized computation. Although more theoretical foundations need to be further explored, this framework might shed more light on the internal mechanisms of the optimization of complex systems, and yield both theoretical insights and practical results in the following applications: computational systems modeling and complex problem solving.

References

Achlioptas D, Naor A, Peres Y (2005) Rigorous location of phase transitions in hard optimization problems. Nature 435:759–764

Ahuja RK, Ergun O, Orlin JB, Punnen AP (2002) A survey of very large-scale neighborhood search techniques. Discret Appl Math 123:75–102

Albert R, Barabási AL (2007 August) The architecture of complexity, from network structure to human dynamics. IEEE Control Syst, pp 33–42

Barabási AL, Albert R (1999) Emergence of scaling in random networks. Science 286(5439):509–512

Boettcher S, Percus AG (2000) Nature's way of optimizing. Artif Intell 119:275–286

Boettcher S, Percus AG (2003) Optimization with extremal dynamics. Complexity 8:57–62

Bonabeau E (2002) Agent-based modeling: methods and techniques for simulating human systems. PNAS 99(suppl. 3):7280–7287

Brin S, Page L (1998) The anatomy of a large-scale hypertextual web search engine. Comput Netw ISDN Syst 30:107–117

Cheesman P, Kanefsky B, Taylor WM (1991) Where the really hard problems Are. Proc IJCAI'91, pp 163–169

Chen Y, Zhang P (2006) Optimized annealing of traveling salesman problem from the nth-nearest-neighbor distribution. Phys A 371:627–632

Duch J, Arenas A (2005) Community detection in complex networks using extremal optimization. Phys Rev E 72:27104

Frey BJ, Dueck D (2007) Clustering by passing messages between data points. Science 315:972–976

Gutin G, Punnen AP (2002) The traveling salesman problem and its variations. Kluwer Academic Publishers, Netherland

Hordijk W (1997) A measure of landscapes. Evol Comput 4(4):335–360

Jennings NR, Bussmann S (2003 June) Agent-based control systems. IEEE Control Syst Mags, pp 61–73

Kauffman SA (1993) The origins of order: self-organization and selection in evolution. Oxford University Press, Oxford

Langton CG (1995) Artificial life: an overview. The MIT press, Cambridge

Li W, Alidaee B (2002) Dynamics of local search heuristics for the traveling salesman problem. IEEE Trans on Syst Man Cybern 32(2):173–184

Liu J (2001) Autonomous agents and multi-agent systems: explorations in learning, self-organization and adaptive computation. World Scientific Publishing Co., Inc., USA

Liu J (2008) Autonomy-oriented computing: the nature and implications of a paradigm for self-organized computing. Keynote talk at the 4th International conference on natural computation (ICNC'08), and the 5th International conference on fuzzy systems and knowledge discovery (FSKD'08), October, pp 18–20

Liu J, Han J, Tang YY (2002) Multi-agent oriented constraint satisfaction. Artif Intell 136:101–144

Liu J, Jin X, Tsui KC (2005) Autonomy oriented computing: from problem solving to complex systems modeling. Kluwer Academic Publishers/Springer, Heidelberg

Liu J, Chen YW, Lu YZ, Yang GK (2008) Self-organized combinatorial optimization. HongKong Baptist University, Computer Science, Technical reports: COMP-08-001. http://www.comp.hkbu.edu.hk/en/research/?content=tech-reports

McCoy BM, Wu Tai Tsun (1973) The Two-dimensional ising model. Harvard University Press, Cambridge

Mertens S (2002 May/June) Computational complexity for physicists. Comput Sci Eng, pp 31–47

Merz P (2004) Advanced fitness landscape analysis and the performance of memetic algorithms. Evol Comput 12(3):303–325

Mézard M, Zecchina R (2002) Random K-satisfiability problem: from an analytic solution to an efficient algorithm. Phys Rev E 66:056126

Mézard M, Parisi G, Zecchina R (2002) Analytic and algorithmic solution of random satisfiability problems. Science 297:812–815

Mitchell M (1998) Computation in cellular automata: a selected review, nonstandard computation. VCH Verlagsgesellschaft, Weinheim 95–140

Mitchell M (2006) Complex systems: network thinking. Artif Intell 170:1194–1212

Monasson R, Zecchina R, Kirkpatrick S, Selman B, Troyanskyk L (1999) Determining computational complexity from characteristic 'phase transitions'. Nature 400:133–137

Newman MEJ (2003) The structure and function of complex networks. SIAM Rev 45(2):167–256

Newman MEJ (2006) Modularity and community structure in networks. PNAS 103(23):8577–8582

Opper M, Saad D (2001) Advanced mean field methods: theory and practice. MIT Press, Cambridge

Poelwijk FJ, Kiviet DJ, Weinreich DM, Tans SJ (2007) Empirical fitness landscapes reveal accessible evolutionary paths. Nature 445(25):383–386

Reinelt G (1991) TSPLIB-a traveling salesman problem library. ORSA J Comput 3(4):376–384

Samanidou E, Zschischang E, Stauffer D, Lux T (2007) Agent-based models of financial markets. Reports Prog Phys 70:409–450

Schuurmans D, Southey F (2001) Local search characteristics of incomplete SAT procedures. Artif Intell 132:121–150

Strogatz SH (2001) Exploring complex networks. Nature 410:268–276

Tsang EPK (1993) Foundations of constraint satisfaction. Academic Press, London

Variano EA, McCoy JH, Lipson H (2004) Networks, dynamics, and modularity. Phys Rev Lett 92(18):188701