A Self-Organization Mechanism Based on Cross-Entropy Method for P2P-Like Applications

GANG CHEN and ABDOLHOSSEIN SARRAFZADEH
Unitec New Zealand
and
CHOR PING LOW and LIANG ZHANG
Nanyang Technological University

15

P2P-like applications are quickly gaining popularity in the Internet. Such applications are commonly modeled as graphs with nodes and edges. Usually nodes represent running processes that exchange information with each other through communication channels as represented by the edges. They often need to autonomously determine their suitable working mode or local status for the purpose of improving performance, reducing operation cost, or achieving system-level design goals. In order to achieve this objective, the concept of status configuration is introduced in this article and a mathematical correspondence is further established between status configuration and an optimization index (OI), which serves as a unified abstraction of any system design goals. Guided by this correspondence and inspired by the cross-entropy algorithm, a cross-entropy-driven self-organization mechanism (CESM) is proposed in this article. CESM exhibits the self-organization property since desirable status configurations that lead to high OI values will quickly emerge from purely localized interactions. Both theoretical and experimental analysis have been performed. The results strongly indicate that CESM is a simple yet effective technique which is potentially suitable for many P2P-like applications.

Categories and Subject Descriptors: H.4.0 [**Information Systems Applications**]: General; C.2.4 [**Computer-Communications Networks**]: Distributed Systems—*Distributed applications*; I.2.8 [**Computing Methodologies**]: Artificial Intelligence

General Terms: Algorithms, Performance, Experimentation

Additional Key Words and Phrases: Self-organization, peer-to-peer system, cross-entropy algorithm

Author's addresses: G. Chen (corresponding author), A. Sarrafzadeh, Department of Computing, Unitec New Zealand; email: chengang@ntu.edu.sg; C. P. Low, L. Zhang, School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore.

DOI 10.1145/1867713.1867716 http://doi.acm.org/10.1145/1867713.1867716

ACM Reference Format:
Chen, G., Sarrafzadeh, A., Low, C. P., and Zhang, L. 2010. A self-organization mechanism based on cross-entropy method for P2P-like applications. ACM Trans. Auton. Adapt. Syst. 5, 4, Article 15 (November 2010), 31 pages.
DOI = 10.1145/1867713.1867716 http://doi.acm.org/10.1145/1867713.1867716

1. INTRODUCTION

In recent years, a generic Peer-to-Peer (P2P) paradigm [Androutsellis-Theotokis and Spinellis 2004a; Lua et al. 2005] has been quickly gaining popularity for developing Internet applications in the area of information distribution [Stoica et al. 2003; Androutsellis-Theotokis and Spinellis 2004b], multimedia streaming [Gu and Nahrstedt 2006; Gu et al. 2007], grid computing [Chakravarti et al. 2005], online event monitoring [Liang et al. 2007; Gedik and Liu 2005; Yalagandula and Dahlin 2004], and many more. These applications are collectively referred to in this article as P2P-like applications.

Very often centralized control is not desirable for P2P-like applications due to its lack of scalability, single point of failure problem, and high communication costs [Goldsrmidt and Yemin 1995; Su et al. 2002]. Self-organization technology, on the other hand, has been demonstrated as a promising solution that can usually meet the design goals of building sometimes extremely large and highly adaptive systems [Bonabeau et al. 1999; Cicirello and Smith 2004; Ledlie et al. 2002].

At the abstract level, a P2P-like application or system is commonly modeled as a mathematical *graph* with each *node* in the graph representing a running process and every *edge* representing a network connection between two processes. The very essence of a P2P-like application lies in the fact that every node in the system is both a service provider and a service consumer. Nodes cannot work alone. They rely on each other to fulfill full functionality. At the time of consuming resources from others, a node also shares its resources with others. Thus, it must practice self-control and make autonomous decisions according to its working mode or *local status*. Since nodes are heterogenous by nature, they must determine their suitable local status in order to improve system performance, reduce operation cost, or achieve certain design goals.

For example, in a P2P information monitoring system to be introduced in Section 5.2, a node can choose two alternative local status corresponding, respectively, to *ordinary mode* and *management mode*. An ordinary node offers its resources for application-specific tasks, whereas a management node monitors ordinary nodes and performs management function. In reality, capability of nodes may vary significantly. For instance, some nodes may have access to broadband communication links while others may have extra processing power. Therefore, depending on its capability strength and the local status of its neighbors, a node may choose to take either a management role if it has high-bandwidth connections to many neighbors or an ordinary role if it has relatively more processing power.

In this article, the concept of *status configuration* is introduced to model nodes' local status in a P2P-like application and will be defined in Section 3.

The *goals* of designing such a system are further abstracted as a general-purpose *Optimization Index* (*OI*) [Nocedal and Wright 2009]. After establishing a mathematical correspondence between status configuration and *OI*, aimed at effectively improving *OI*, this article continues to develop a mechanism for self-organizing nodes' local status with concrete contributions as follows:

—Inspired by the cross-entropy algorithm [Rubinstein and Kroese 2004], which is a recently developed optimization method, a *cross-entropy-driven self-organization mechanism* (CESM) is proposed. CESM allows nodes in a P2P-like system to independently optimize their local status. It exhibits the self-organization property since desirable status configurations that lead to high *OI* values will emerge from purely distributed interactions.

—Theoretical analysis of CESM is performed and included in this work. The analysis indicates that, at least for a common type of self-organization problem, CESM allows nodes to discover near-optimal status configurations very quickly. The effectiveness of CESM is further evaluated experimentally on two representative problems, namely the 0-1 knapsack problem [Martello and Toth 1990] and the P2P information monitoring problem. The experiment results seem to agree with our theoretical analysis.

This article focuses on self-organizing nodes' local status in P2P-like applications. As such, only the correlation between *OI* and status configuration is studied. However, it should be noted that a subtle *connection* might exist between *OI* and the edges in a P2P system as well, especially when nodes frequently alter their neighbors. To be more specific, the change of a node's local status might affect its surrounding edges. With the change of edges, the suitable local status of the node might change accordingly. This is clearly demonstrated by various object placement and caching strategies in P2P systems [Mu et al. 2006; Zhang et al. 2004]. The possibility of exploiting this connection to further improve *OI* will not be addressed in this article and will be treated as a potential future research direction.

The remainder of this work is organized as follows. Section 2 summarizes and compares related research works. Section 3 presents a P2P model and defines a self-organization problem. In Section 4, the proposed CESM mechanism will be discussed in detail and theoretically analyzed. Experiment evaluation is reported in Section 5. Finally, Section 6 concludes.

2. RELATED WORK

In the Internet, P2P-like applications enjoy several advantages in comparison with the traditional client-server applications. Specifically, every node in a P2P system shares its resources with others. Thus the total capacity of the system will increase as nodes arrive and demand on the system rises. This is not true for client-server systems with fixed number of servers. The distributed nature of P2P systems also improves robustness since system-wide functionality will not be interrupted by failures of a few nodes. Self-organization techniques have been widely exploited in P2P systems and can be largely classified into

techniques for self-organizing edges and techniques for self-organizing nodes' local status.

In the literature, techniques for edge self-organization have been extensively utilized to improve the efficiency of resource discovery in P2P systems. Many techniques were strongly influenced by the seminal work on small world networks [Kleinberg 2000; Jin et al. 2006; Li et al. 2004; Manku et al. 2003; Zhang et al. 2004]. For example, Zhang et al. introduced an *Enhanced Clustering Cache Replacement scheme* (ECCR) to allow nodes to self-organize themselves into a small world network, in which every node is only a few hops away from each other Zhang et al. [2004]. In the mean time, bio-inspired approaches are quickly gaining popularity. For example, in Forestiero et al. [2009], a bio-inspired P2P algorithm named "Self-Chord" has been utilized to enhance search flexibility in Chord-like P2P systems [Stoica et al. 2003]. By allowing a set of ant-inspired mobile agents to move and reorder the resource keys in the Chord ring, "Self-Chord" enables "class" query for a set of nodes that share common resource characteristics.

In addition to resource discovery, edge self-organization was also explored for balancing resource utilization in P2P systems. For example, in Karger and Ruhl [2004], a protocol was proposed to achieve load balance through balancing the distribution of hash key space and consequently the distribution of data items among the nodes. Bridgewater proposed a *Balanced Overlay Network* (BON) which presented another novel approach for balancing resource usage in P2P systems [Bridgewater et al. 2007]. The number of edges that connect a node to the rest of a P2P system is termed the *in-degree* of the node. In BON, a node's in-degree is kept proportional to the amount of resources shared by it. At any time when extra resources are required, a random walk is performed in the P2P system. The node reached at the end of the walk will be chosen to provide the resources. A similar idea has been studied in Gedik and Liu [2005] as well. A different approach for accessing resources in P2P systems was also considered in Ghanea-Hercock et al. [2006] through manipulating the weight of edges.

Rather than focusing on edge self-organization, this article concentrates on self-organizing nodes' local status. Recently, status self-organization has attracted increasing research interest in the literature. As far as the authors know, most of the research in this direction can be classified into two large categories, namely (1) status self-organization for solving system-wide problems and (2) for improving system-level performance.

One recent example of the first category is a series of technologies for distributed detection of global triggers [Ko et al. 2008]. As a demonstration of a methodology for translating sequence equations of natural phenomenon into sequence protocols, two protocols have been proposed in Ko et al. [2008] to detect when the global average of a variable crosses a threshold or falls outside an interval. The detection is made possible since every node will eventually choose an identical status after the system stabilizes. In this article, however, the authors will consider a different scenario where it is more desirable for nodes to choose varied status.

Another example of the first category is the population protocol for distributed computation of functions or predicates [Angluin et al. 2004]. With properly defined input mapping, output mapping, a group of local status, and a status transition mapping, the self-organization process as driven by the population protocol can stably compute a variety of functions including predicates in a P2P system. The overall computability of this protocol was analyzed in Angluin et al. [2004]. Theoretically, both the trigger detection protocols and the population protocol can find their root in the gossip protocol as a basic means of computing aggregated information [Kempe et al. 2003]. Rather than performing any kind of computations, this article seeks to achieve generic system design objectives in the form of an optimization index which is computable based on the local status of every node in a P2P system. The distributed calculation of this index benefits a lot from the gossip protocol, as described in the next section.

Besides problem solving, system-level performance can also be improved through self-organizing nodes' local status. For instance, the HoneyAdapt protocol, proposed in Ko et al. [2008] as a self-adaptive grid computing protocol, enables distributed workstations to independently select the most suitable algorithm (the algorithm selected by a node is viewed as its status) for any given task. Similar with the trigger detection protocols, the choices made by every node will finally converge to an identical algorithm which is considered the most suitable one. Another example is the *connectivity-based distributed node clustering scheme* (CDC) which presented a scalable and efficient solution for discovering connectivity-based clusters in P2P systems [Ramaswamy et al. 2005]. Through local interactions, nodes in a P2P system self-organize themselves into separate clusters. The local status of a node in this research is its cluster membership. Using the clusters thus formed, the cost of system-wide broadcast can be reduced by following a version of clustered broadcast termed *forest broadcast* [Ramaswamy et al. 2005].

The techniques for status self-organization have also been explored for controlling service provision and task allocation in P2P applications. In this regard, the services offered or tasks performed represent the local status of a node. Cuenca-Acuna and Nguyen considered the UDDI service, which is typically replicated across multiple nodes [Cuenca-Acuna and Nguyen 2004]. They proposed a resource management framework with the aim of controlling the set of service providing nodes. Unfortunately, a centralized approach based on Genetic Algorithm (GA) [Goldberg 1989] was utilized to find desirable status configurations which are not outcomes of self-organization. As a result, the overall performance of their system depends heavily on the correct functioning of the central controller. A distributed approach for task allocation was proposed in Cicirello and Smith [2004]. By modeling nodes as wasps, each node will adaptively accept only a specific type of tasks such that the total cost involved for task fulfillment can be significantly reduced. The authors of this article have also proposed a similar mechanism to control service provisioning by following a policy-driven approach [Chen et al. 2008].

Instead of addressing a specific problem, this article considers status self-organization under a generic optimization framework [Papadimitriou and

Steiglitz 1998]. The self-organization mechanism to be presented in this work, namely CESM, is essentially a stochastic and distributed optimization algorithm. It shares many similarities with evolutionary computation algorithms (EO) [Yao and Xu 2006; Zlochin et al. 2004; Bäck and Schwefel 1993; Fogel 1995]. However, traditional EO such as GA does not allow individual components of an optimization problem to be handled independently by different nodes and hence is only applicable in hierarchically organized P2P systems.

It should be noted that the design of CESM draws much of its inspiration from the cross-entropy algorithm, which is a general Monte Carlo approach to combinatorial optimization and importance sampling [Rubinstein and Kroese 2004]. In view of this, it is therefore eligible to view CESM as a *special case* of the Monte Carlo method [Rubinstein and Melamed 1998b], a special case that is designed purposefully for P2P-like applications. A sampling technique is extensively used in CESM and based on the sampling results, nodes are able to independently determine their suitable local status. Consequently, a system-wide optimization problem can be solved in a purely distributed manner.

As far as the authors know, nobody has ever considered using the cross-entropy method before for system-wide optimization through self-organizing nodes' local status. We notice that the entropy concept has been extensively utilized before for system-wide optimization via evolution of the system's modules [Prokopenko et al. 2006]. However, in Prokopenko et al. [2006], entropy was used to measure the performance of a system design and evolutionary algorithms were further exploited to improve the design, whereas we use entropy to guide the local sampling process performed by every node in a P2P system.

Before the end of this section, it is noted that the cross-entropy method has already been exploited in the literature to find communication paths in telecommunication systems through ant-like mobile agents [Wittner et al. 2003; Heegaard et al. 2008]. Specifically, agents use a variation of the cross-entropy method to determine the most suitable communication paths within a telecommunication system. No edges or the local status of any node in the telecommunication system will be affected by agents' activities. In this article, we aim to use the cross-entropy method to adjust the local status of nodes in P2P-like applications. The effectiveness of this method as a self-organization mechanism will also be theoretically and experimentally analyzed.

3. P2P ENVIRONMENT AND A SELF-ORGANIZATION PROBLEM

Consider a P2P application that comprises of N nodes. Sometimes, for simplicity of analysis, it is assumed that the group of nodes in the system will never change and as a result the value of N is fixed throughout time. Such assumptions were exploited in Ko et al. [2008] and Kempe et al. [2003], for example, for analyzing a collection of distributed processes. When CESM is proposed in this article, we also assume a closed group of nodes. This is because CESM relies on nodes' capability of using past experience with the system to determine their suitable local status in the future. Nevertheless, as long as nodes' influence

on the overall functioning of the system is properly localized and a majority of nodes' neighbors tend to stay in the system, CESM may still be applicable even if nodes are free to leave and join the system, as the experiments in Section 5 demonstrate.

Though not essential, communication in a P2P application is assumed to be reliable. Nodes can communicate with each other through reliable communication channels such as TCP channels. They are inherently cooperative and are willing to share information about their local status. It is to be noticed that the problem of cooperation and incentives is a major concern in P2P systems. In order to encourage cooperation, a carefully designed scheme based on game theory might be utilized such that cooperation is to the ultimate benefit of every node in the system [Gupta et al. 2006]. However this article will not elaborate this issue any further.

A node is represented by p_i, where i is the universal node ID (NID). The set of all nodes is further represented as $\mathcal{P}$. At any time, a node must choose one local status from a set of alternative local status denoted as $\mathcal{S} = \{s_1, s_2, \ldots, s_m\}$. There is no restriction on the size of $\mathcal{S}$ except it must be *finite*. The exact definition of $\mathcal{S}$ varies for different applications. As an example, in a service-oriented P2P system, $\mathcal{S}$ might simply contain two elements, corresponding respectively to *service mode* and *ordinary mode*. At any time t, a node can choose either mode and operate consequently as a service provider or a service consumer.

Each node p_i is associated with a group of neighbors (or local contacts). The direct communication channel between p_i and one of its neighbors p_j is indicated as an edge e_{ij}. The set of all edges is further denoted as $\mathcal{E}$. Since this work focuses on self-organizing the local status of nodes, edges are assumed to be relatively stable. A node will not change its neighbor unless its neighbor has left the system. It is also assumed that with the edges in place, the system is *fully connected*. Each node can reach any other node through at least one communication path.

For a P2P-like application, the *status configuration* of the system at any time t, denoted as C_t, can be treated conveniently as a string constructed from the alphabetic table $\mathcal{S}$. The length of the string is N and each of its elements $C_t[i]$ corresponds to the local status of a node p_i in the system. $\mathcal{C}$ further denotes the *finite* set of all possible status configurations. A status configuration essentially represents a global view of the system. Every node, however, merely has a partial view of C_t. In some constrained situations a node might only know the local status of itself and its neighbors. The mechanism CESM to be introduced in the next section satisfies this constraint.

In a P2P application, the degree of satisfaction of any system design goals can often be quantified through an *Optimization Index* (OI), which will be treated in this article as a function that maps every possible status configuration $C \in \mathcal{C}$ to a *real number*. With proper definitions of OI, we can easily ensure that large OI values will lead to greater satisfaction of system design goals. Obviously the exact meaning of OI may vary with applications, as demonstrated by the 0-1 knapsack problem and information monitoring problem in Section 5. Without being restricted to any specific applications, CESM will be designed to work with generic definition of OI.

With all the concepts described earlier, a *status self-organization problem* comes into sight naturally. In a nutshell, the problem can be boiled down to three basic requirements:

(1) nodes in a P2P application are allowed to exchange information with their neighbors only;
(2) no node has a global view of the system and therefore can determine the suitable local status of other nodes on their behalf;
(3) all nodes will jointly identify a certain status configuration C^*, which will increase OI to its global maximum. C^* is hence termed the *optimal status configuration*.

When the definition of OI changes, the problem can exhibit diverse complexities. In certain cases, it can even be NP-hard. A good example is the 0-1 knapsack problem which is a NP-hard problem that will be experimentally evaluated in Section 5.1. In view of this fact, it is therefore very difficult to satisfy the third requirement given before in the most rigorous sense. Instead, any near-optimal status configurations will also be considered acceptable for our self-organization problem.

Conceptually, a solution to any self-organization problem can usually be captured through a generic feedback framework. Specifically, the behavior of a P2P application that is of common interest to all nodes in the system will be constantly monitored by these nodes, which will treat monitored data as feedback from the system and adjust their local status and connections to neighbors accordingly based on the feedback. Through such constant adjustment, which happens locally at every node, the whole system will gradually self-organize itself towards the satisfaction of certain desirable properties.

Within the context of our status self-organization problem, it is straightforward to see that the behavior of a P2P system to be monitored is OI, since the ultimate goal is to increase OI to its maximum. The monitored OI will be subsequently utilized by CESM to help nodes adjust their local status. Apparently, a node cannot estimate OI based on its local status alone. However, calculation of OI might be very expensive as it requires every node to broadcast its local status to all other nodes in the system. Nevertheless, if OI is defined over the system-wide average values of some variables maintained locally by every node in the system, then it can be calculated very efficiently by using the gossip protocol [Kempe et al. 2003].

Subject to the application domain, the variables that a node maintains might bear varied meanings. As system designers, we need to carefully choose those variables such that the system-level design goals can be adequately captured through them. For example, in a service-oriented P2P system, every node needs to maintain one variable v that quantifies the quality of service it received from its service provider. The quality of service can be measured according to many factors, such as the average response time and Mean Time Between Failures (MTBF). Regardless of the factors involved, a node should be fully capable of calculating v independently by itself, since it observes all interactions with its service provider. In case it cannot calculate v all alone, a node may alternatively

Initialization:
Each node p_i in a P2P system calculates independently its local value of v, namely v_i, and initializes a pair $(\hat{v}_i, \hat{w}_i)$, where $\hat{v}_i := v_i$ and the weight $\hat{w}_i := 1$.

Repeat for each round of communication:

(1) Every node p_i randomly chooses one of its neighbors p_j and sends the pair $\left(\frac{\hat{v}_i}{2}, \frac{\hat{w}_i}{2}\right)$ to p_j through edge e_{ij}. It will also send the pair $\left(\frac{\hat{v}_i}{2}, \frac{\hat{w}_i}{2}\right)$ to itself.

(2) After receiving all pairs of the form $\{(\hat{v}_r, \hat{w}_r)\}$ from neighboring nodes (including itself), node p_i will update its pair $(\hat{v}_i, \hat{w}_i)$ according to

$$\hat{v}_i = \sum_r \hat{v}_r, \quad and \quad \hat{w}_i = \sum_r \hat{w}_r$$

(3) $\frac{\hat{v}_i}{\hat{w}_i}$ is calculated as the latest estimation of $\bar{v}$ at node p_i.

Fig. 1. A protocol for calculating $\bar{v}$.

ask its service provider for help, as our P2P information monitoring problem in Section 5.2 demonstrates.

Let v_i denote a variable maintained locally by a particular node p_i. In association with v_i, the system-wide average of v, denoted as $\bar{v}$, can be consequently applied to quantify the performance of the P2P system. In view of this, OI is further defined as

$$\forall C_t \in \mathcal{C}, \quad OI(C_t) = F(\bar{v}). \tag{1}$$

Only one variable v is involved in the definition of OI in (1). Nonetheless, it is straightforward to extend (1) to include as many variables as needed. Meanwhile $F(\cdot)$ is an application-specific function. It can assume varied meanings according to the application domain. For example, if $\bar{v}$ stands for the average response time in a service-oriented P2P system, then we can define $F(\bar{v}) = 1/\bar{v}$ to measure the quality of service of the system. In practice, we don't restrict the type of functions that can be used except that $F(\cdot)$ must be *bounded* and *continuous*.

In principle, for any node p_i and at any time t, the variable v_i that p_i maintains can be modeled as a function that maps C_t to a real number $v_i(C_t)$. Although our definition of v_i involves global knowledge C_t, nevertheless as the preceding example of service-oriented P2P system shows, the calculation of v_i can often be performed locally. The authors also believe that, in practice, OI often takes or can be converted to the form in (1). In case OI cannot be defined as in (1), we need to seek for other efficient methods for calculating it. According to (1), if every node knows $\bar{v}$, OI is known immediately. For every node to quickly estimate $\bar{v}$, a protocol as shown in Figure 1 can be utilized, which is adapted directly from the *push-sum protocol* (a particular type of gossip protocols) in Kempe et al. [2003].

The protocol in Figure 1 is based on repeated information exchange between adjacent nodes. For convenience, the communication is usually organized in rounds. As indicated in Figure 1, during each round, a node is only allowed to send a message to *one* of its neighbors. In practice, however, in order to expedite the propagation of information, a node can usually broadcast a message to all

of its neighbors. For this purpose, an adjustment is to be made to Figure 1. Specifically, if a node p_i has x neighbors, then it should broadcast the pair $(\hat{v}_i/(x+1), \hat{w}_i/(x+1))$ to all its neighbors and itself during each communication round.

Theoretical analysis in Kempe et al. [2003] showed that, whenever the underlying network topology of a P2P system is an *expander*, at most $O(\log(N) + \log(1/\epsilon))$ rounds of communication are sufficient to ensure that the error of estimating $\bar{v}$ by every node in the P2P system has dropped to within ϵ with high probability. The protocol is therefore highly efficient and scalable. It is noted that many existing P2P applications in the Internet have good expansion properties and are therefore expanders [Keyani et al. 2002; Pandurangan et al. 2001].

4. A CROSS-ENTROPY-DRIVEN SELF-ORGANIZATION MECHANISM

In Rubinstein and Melamed [1998a] and Rubinstein and Kroese [2004], the authors developed a generic optimization technique termed the cross-entropy method. At the very basis of this method is the *rare event theory*. In plain words, for large P2P systems, finding a certain status configuration $C \in \mathcal{C}$ which will lead to near-optimal OI can normally be considered as a rare event. To increase the chances of discovering such a status configuration, the probability of sampling any configuration $C \in \mathcal{C}$ should be carefully controlled to favor those with high OI values. This idea gives rise to the *importance sampling* technique, which characterizes the importance of a status configuration C through a *Boltzmann function* defined as

$$H(C, \gamma) = \exp\left\{\frac{OI(C)}{\gamma}\right\}, \tag{2}$$

where $\gamma \in \mathbf{R}^+$ is the *Boltzmann temperature*. As indicated in (2), depending on $OI(C)$, a status configuration C can have varied importance and the level of difference will be amplified upon reducing γ. In particular, it is easy to verify that

$$\lim_{\gamma \to 0} \frac{H(C^*, \gamma)}{H(C, \gamma)} = \infty, \quad C^* \neq C. \tag{3}$$

Ideally, if any status configurations $C \in \mathcal{C}$ can be sampled with a probability below

$$\forall C \in \mathcal{C}, \Theta^*(C) = \frac{H(C, \gamma)}{\sum_{C' \in \mathcal{C}} H(C', \gamma)} = \frac{H(C, \gamma)}{\delta(\gamma)} \tag{4}$$

then by properly adjusting γ, (3) implies that there can be a good opportunity of discovering C^*. Θ^* is termed the *importance sampling density*. $\delta(\gamma)$ in (4) is the *partition function*. Calculating $\delta(\gamma)$ for very small temperature γ is of major significance in many disciplines, including physics, operations research, and computer science [Rubinstein and Kroese 2004].

In order to sample status configurations according to (4), the value of $\delta(\gamma)$ must be known in advance. Unfortunately, the knowledge of $\delta(\gamma)$ directly

implies the knowledge of Θ^* and $H(C^*, \gamma)$. To solve this problem, a group of status configurations $\{C_1, \ldots, C_T\}$ will be sampled first according to an arbitrary sampling density Θ, which will be gradually revised to approximate Θ^*. Using the widely exploited *cross-entropy* measure, the degree of similarity between Θ and Θ^* can be estimated based on sampled status configurations as

$$D(\Theta, \Theta^*) = \sum_{t=1}^{T} \left(\Theta^*(C_t) \times \ln \frac{\Theta^*(C_t)}{\Theta(C_t)} \right). \tag{5}$$

It is known that $D(\Theta, \Theta^*)$ in (5) is absolutely nonnegative. It will reach its minimum 0 if sampling densities Θ and Θ^* are identical at all sampled status configurations. Intuitively, when both $D(\Theta, \Theta^*)$ and γ are small enough, a status configuration $C \in \mathcal{C}$ will be sampled more frequently than other configurations provided that $OI(C)$ is relatively larger. As the sampling is skewed towards high OI area in $\mathcal{C}$, hopefully C^* can be quickly identified. In light of this view, the following optimization problem is expected to be solved.

$$\min_{\Theta} D(\Theta, \Theta^*) = \min_{\Theta} \left\{ \sum_{t=1}^{T} \left(\Theta^*(C_t) \times \ln \frac{\Theta^*(C_t)}{\Theta(C_t)} \right) \right\} = \max_{\Theta} \left\{ \sum_{t=1}^{T} \Theta^*(C_t) \times \ln \Theta(C_t) \right\} \tag{6}$$

Notice that $\delta(\gamma)$ can be treated as a constant for any fixed γ. Hence replacing (4) in (6) produces a simpler problem as

$$\max_{\Theta} \left\{ \sum_{t=1}^{T} H(C_t, \gamma) \times \ln \Theta(C_t) \right\}. \tag{7}$$

In a P2P application, the sampling of any status configuration will be performed in a solely distributed manner. After a node has sampled its local status, it will continue to determine the values of its locally maintained variables and calculate system-wide average of these variables through the protocol in Figure 1. Subsequently, the value of OI in (1) and the Boltzmann function in (2) will also be quickly estimated. With respect to each local status $s_j \in \mathcal{S}$, there is a sampling probability $\theta_i(s_j)$ for node p_i to choose s_j. Since nodes sample their local status independently, $\ln \Theta(C_t)$ in (7) can be rewritten as

$$\ln \Theta(C_t) = \ln \left\{ \prod_{i=1}^{N} \theta_i(C_t[i]) \right\} = \sum_{i=1}^{N} \ln \left(\theta_i(C_t[i]) \right). \tag{8}$$

Using Fermat's theorem [Rudin 1976], the exact value of $\theta_i(s_j)$ that maximizes (7) can be obtained as a solution of the following equation.

$$\frac{\partial \left(\sum_{t=1}^{T} H(C_t, \gamma) \times \ln \Theta(C_t) \right)}{\partial \theta_i(s_j)} = 0 \tag{9}$$

In order to solve (9), it is helpful to divide $\mathcal{S}$ into two groups. One group contains just status s_j while the other group contains the rest. Accordingly, we have

Initialization:
Initialize the sampling density Θ uniformly such that, for every node p_i, its sampling probabilities $\theta_i(s_j), 1 \leq j \leq m$, are of identical values.

Repeat for each round of sampling:

(1) For each node p_i, use its current sampling probabilities $\theta_i(s_j), 1 \leq j \leq m$, to generate a group of T sampled configurations.
(2) After generating every sampled configuration, node p_i will calculate the corresponding value of OI through the protocol in fig. 1. Whenever possible, the local status chosen by p_i, which is related to the highest OI value ever identified, will also be recorded.
(3) Based on the T recently generated samples, each node p_i will update its sampling probabilities $\theta_i(s_j), 1 \leq j \leq m$, according to (12) to jointly produce a new approximation of Θ^*.

Fig. 2. A cross-entropy-driven self-organization mechanism (CESM).

$$\theta_i(C_t[i]) \begin{cases} \theta_i(s_j), & C_t[i] = s_j \\ 1 - \theta_i(s_j), & C_t[i] \neq s_j \end{cases}. \tag{10}$$

Substituting (8) and (10) into (9) produces

$$\sum_{t=1}^{T} \left\{ H(C_t, \gamma) \cdot \frac{1}{\theta_i(s_j)} \cdot \lambda(C_t[i] = s_j) \right\} = \sum_{t=1}^{T} \left\{ H(C_t, \gamma) \cdot \frac{1}{1 - \theta_i(s_j)} \cdot \lambda(C_t[i] \neq s_j) \right\}. \tag{11}$$

Function $\lambda(\cdot)$ in (11) returns 1 when the condition as indicated in its argument is true and returns 0 otherwise. The solution of (11) is

$$\theta_i(s_j) = \frac{\sum_{t=1}^{T} H(C_t, \gamma) \cdot \lambda(C_t[i] = s_j)}{\sum_{i=1}^{T} H(C_t, \gamma)}. \tag{12}$$

Based on the preceding discussion, for all sampling probabilities $\theta_i(s_j)$, if their values are updated according to (12), then $D(\Theta, \Theta^*)$ will be reduced to its minimum and Θ consequently presents a good approximation of Θ^*. In view of this, a cross-entropy-driven self-organization mechanism (CESM) is proposed in Figure 2 in the form of an iterative sampling process. In the remaining part of this article, we will refer to each round of the sampling process in Figure 2 as a *sampling round*. During each sampling round, Θ^* will be approximated by Θ using (12), which will be further utilized to produce T sampled status configurations. Since Θ is reasonably similar to Θ^*, hopefully, status configurations with high OI values (or high OI configurations) can be identified among the T samples.

The inherent simplicity of CESM allows it to be implemented very efficiently in practical P2P applications. Despite of its simplicity, however, at least for a common type of problem, near-optimal status configurations can be quickly identified through CESM, as presented in the subsection that follows.

4.1 Effectiveness of CESM

As explained in Section 3, depending on the exact definition of OI in (1), our status self-organization problems can exhibit diverse complexities and some of them might even be NP-hard. CESM, like other stochastic optimization techniques, is theoretically incapable of solving all types of problems in an equally efficient manner. Nevertheless, with respect to one specific type of problem, which will be studied in this subsection, CESM provides an effective and efficient solution. To assist our analysis of CESM, extra notations are introduced in this subsection. Please note that some notations used herein might bear differed meanings from those used elsewhere in the article and should therefore be treated separately. For convenience, let

$$\tilde{H}(i, s_j, \gamma, \Theta) = \sum_{C \in \mathcal{C}} H(C, \gamma) \cdot \Theta(C) \cdot \lambda(C[i] = s_j), 1 \leq i \leq N, s_j \in \mathcal{S} \tag{13}$$

be defined under two conditions: (1) the local status of node p_i is s_j, and (2) the sampling density is Θ. A self-organization problem is said to be *unimodal* in this article provided that there exists a γ such that for any Θ with $\Theta(C^*) > \epsilon$ and any $s_j \neq C^*[i]$

$$\tilde{H}(i, s_j, \gamma, \Theta) \leq (1 - \alpha) \cdot \tilde{H}(i, C^*[i], \gamma, \Theta), 1 \leq i \leq N. \tag{14}$$

where ϵ and α are two small positive constants between 0 and 1. As the authors believe, unimodal problem is a common type of problem in P2P applications. Theoretically, at least two main reasons support this view:

R1. Among all the status configurations in $\mathcal{C}$, if exactly one configuration C^* can increase OI to its global maximum, then the corresponding self-organization problem is unimodal.

R2. In case there are multiple optimal configurations, by properly restricting the sampling process to a certain subset $\mathcal{C}'$ of $\mathcal{C}$, $\mathcal{C}' \subset \mathcal{C}$, which contains just one optimal configuration, the problem becomes unimodal.

R1 as given before can be directly verified from (3) when γ is small enough. In particular, for any node p_i and arbitrarily small constant $\epsilon > 0$, the inequality in (15) can be satisfied as γ approaches 0. Furthermore (15) implies (14) since $\mathcal{C}$ is a finite set. R2 follows straightforwardly from R1, because if we can restrict the sampling process to the subset $\mathcal{C}'$ in R2, there is exactly one configuration $C^* \in \mathcal{C}'$ that increases OI to its maximum among all $C \in \mathcal{C}'$. According to R1, the self-organization problem is therefore unimodal. R2 suggests that a multimodal problem can be converted to a unimodal one if the sampling process can be restricted properly. To actually restrict the sampling process, however, some domain knowledge is necessary. In general, the requirements in either R1 or R2 may be easily met by common P2P applications, which are hence considered unimodal.

$$\forall \Theta(C^*) > \epsilon,\ 0 \leq \lim_{\gamma \to 0} \left\{ \frac{\tilde{H}(i, s_j, \gamma, \Theta)}{\tilde{H}(i, C^*[i], \gamma, \Theta)} \right\} \leq \lim_{\gamma \to 0} \left\{ \frac{\tilde{H}(i, s_j, \gamma, \Theta)}{H(C^*, \gamma) \cdot \epsilon} \right\} = 0 \tag{15}$$

Theorem 1 that follows shows that, for unimodal problems, CESM can quickly self-organize the status configuration C of a P2P application towards its global optimal C^*. The proof of this theorem can be found in Appendix II.

THEOREM 1. *In a P2P system of N nodes, the cross-entropy-driven self-organization mechanism (CESM) as specified in Figure 2 can efficiently solve a unimodal problem with arbitrarily small $\epsilon > 0$. Specifically, with a probability of at least $1 - \delta$, at most $O(\log N + \log \frac{1}{\delta})$ sampled configurations are expected before the optimal status configuration C^* can be identified.*

As indicated in Figure 2, after generating each sampled configuration, distributed calculation of OI will be performed using the protocol in Figure 1. Since $O(\log N)$ rounds of communications are required for evaluating OI (refer to Section 3), an important consequence of Theorem 1 is now immediate.

COROLLARY 1. *In a P2P system with N nodes, each node is expected to perform $O((\log N)^2 + \log N \times \log \frac{1}{\delta})$ rounds of communications with its neighbors in order to identify the optimal status configuration C^* of a unimodal problem through using CESM.*

4.2 A Random Strategy for CESM

Theorem 1 and Corollary 1 in Section 4.1 together show that unimodal problems can be effectively solved by CESM without incurring excessive communication cost. Nevertheless, the search for C^* is strongly influenced by the T samples chosen for every sampling round. In particular, CESM is subject to *immature convergence* if all T samples are located closely around a local optimum in the space of status configurations. To tackle this problem, a random strategy is to be adopted.

One simple strategy which is implemented by the authors is for nodes to occasionally choose a local status uniformly at random. Obviously, not all nodes are equally suitable for this strategy. On one hand, for any node p_i in a P2P system, if no sampling probability $\theta_i(s_j)$ actually *dominates* the local sampling process, it is not necessary for p_i to follow the random strategy which might slow down the convergence. On the other hand, when dominating sampling probabilities exist, the random strategy turns out to be beneficial in order to prevent the sampling process from being trapped by any local optima. As an indication of the degree of dominance, an *entropy* measure defined next is used.

$$Ent(p_i) = -\sum_{j=1}^{m} \left(\theta_i(s_j) \cdot \ln \theta_i(s_j)\right) \tag{16}$$

$Ent(\cdot)$ is maximized when sampling probabilities are uniformly distributed, namely $\theta_i(s_j) = \frac{1}{m}$, $1 \le j \le m$. It will be gradually reduced to its minimum 0 as one of the sampling probabilities $\theta_i(s_j)$ approaches 1 and therefore dominates the local sampling process. Guided by $Ent(\cdot)$, a controlled use of the random strategy can be achieved as in Figure 3. The strategy will be utilized every time upon sampling a new status configuration. Both μ and ρ are treated as tunable parameters of the strategy.

Repeat while generating every new sample:
Each node p_i will calculate locally its entropy measure $Ent(p_i)$ in (16).
If $Ent(p_i) \leq \mu$, **then**

With a probability of ρ, node p_i will select a local status uniformly at random. Otherwise, any local status s_j will be chosen based on its sampling probabilities $\theta_i(s_j)$.

Else

Node p_i will randomly select a local status according to its sampling probabilities $\theta_i(s_j)$.

Fig. 3. A random strategy for CESM.

5. EXPERIMENT ANALYSIS

Experiment study is performed in this section to evaluate the effectiveness of CESM through two problems that are derived from real-life applications of P2P-like applications. The first problem can be considered as a distributed version of the famous 0-1 knapsack problem, which is one of the most studied problems in combinatorial optimization with many practical applications [Martello and Toth 1990]. This problem is utilized here since the optimal solutions to the knapsack problem can be directly identified through the dynamic programming method. The capability of CESM for finding optimal or near-optimal solutions of knapsack problems or other similar problems can therefore be assessed.

Different from the 0-1 knapsack problem, which is theoretical by nature, as a demonstration of the practical usefulness of CESM, the second problem is derived from a P2P information monitoring system. At the time this article was written, the use of the P2P paradigm for system-wide information monitoring was still an interesting but ongoing research issue with limited real-life applications. Due to our difficulty of accessing data from a real-world P2P system that offers an online information monitoring service, we have decided to evaluate the performance of CESM through a simulation system. Nevertheless, we believe the experiment data we obtained comprises strong indications of CESM's potential in practical applications.

For both of the two problems, the performance of CESM is compared with the results of using GA. GA is considered in this article as a competing approach for status organization in P2P applications. The purpose of our comparison is to show the competitive edge of CESM as a general-purpose optimization method for distributed systems. It is worthwhile to note that CESM is a purely distributed mechanism. In contrast, GA as well as many other evolutionary optimization methods demand centralized control. To apply GA in a P2P application, usually the system would need to be organized into a federated or tiered architecture, as Cuenca-Acuna and Nguyen's work demonstrated [Cuenca-Acuna and Nguyen 2004]. Very often the whole system will be divided into a collection of subgroups or regions, for example, by using a distributed clustering algorithm [Ramaswamy et al. 2005]. One node will be elected as *supernode* for every region and is responsible for seeking near-optimal status configurations for its region with the help of GA.

This federated approach is generally valid, since although a P2P system can possibly contain tens of thousands of nodes, in reality each node only has *localized impact* on the operation of the system. As a result, it is not always necessary to define OI on the whole system. Instead the system will be divided into multiple regions and each region will have its own definition of OI. The size of a region is comparatively much smaller than the size of the system. In view of the preceding, it is therefore possible to consider our experiments as comparing the performance of CESM and GA at the level of network regions. The main difference is that, for the GA-based approach a supernode is to be elected to determine the local status of all other nodes in a region, whereas for CESM, nodes autonomously determine their local status without relying on any supernodes.

The implementation of GA follows exactly the descriptions in Goldberg [1989]. Specifically, the widely used one point crossover operator is exploited to drive the search of optimal status configurations, which is facilitated by a single-point mutation operator. If this mutation operator is applied on a candidate configuration in a population, one element of the configuration is selected arbitrarily and the corresponding local status will be chosen uniformly at random. Parameters τ_c and τ_m are used to control respectively the *crossover rate* and the *mutation rate* of GA. Different combinations of the two parameters will be exploited in the experiments to illustrate their impacts on GA. Among all experiments, the size of a population is fixed at 100, and the number of generations is 1000.

As a common setting, the number of nodes N at the beginning of all our experiments is 300. The actual number of nodes may change throughout the experiments if nodes are allowed to leave and later rejoin the system. It may not be suitable to use a number like 300 to measure the size of a real-life P2Psystem. However, as we believe, it is reasonable for measuring the size of a network region. For example, if we are deciding to elect a node as supernode, the number of nodes under its control can rarely exceed 300 due to its limited processing power and networking capacity. This means that the decision of having a supernode will have local impact on less than 300 nodes only. Finally, all experiment results presented in this section are obtained by *averaging* the results of 25 independent simulations.

5.1 Distributed 0-1 Knapsack Problem

In this subsection, the effectiveness of CESM will be examined through a distributed version of the 0-1 knapsack problem. We choose to use knapsack problem for several purposes:

—Knapsack problems are NP-hard problems that are not usually unimodal. Hence Theorem 1 cannot be directly applied. However problems that do not satisfy the unimodal properties abound in the literature. The effectiveness of CESM for nonunimodal problems therefore has to be evaluated through a typical NP-hard problem such as knapsack problems.

—Knapsack problems have many real-life applications in the area of distributed computing and computer networking [Altman et al. 2001]. In

particular, many problems that are of practical value to P2P applications can be easily formulated as knapsack problems. An example will be presented later in this subsection. The problems can therefore be used to show the potential usefulness of CESM in P2P applications and other distributed systems.

—The global optimum of knapsack problems can be sought after a priori using the dynamic programming method. With a predetermined optimum, CESM's capability of finding optimal or near-optimal solutions of NP-hard problems can be assessed.

It is to be noted that many efficient algorithms have been developed in the past to solve knapsack problems [Martello and Toth 1990]. Most of the algorithms, such as the dynamic programming method used in our experiments, demand centralized control. CESM, however, is not designed specifically for solving knapsack problems. Our focus is to find out whether difficult problems such as knapsack problems can be solved in a purely distributed manner by CESM with acceptable efficiency.

For 0-1 knapsack problems, two alternative local status $\mathcal{S} = \{s_1, s_2\}$ are available for every node to choose. Additionally, a node p_i maintains locally two variables π_i and w_i. π_i is termed the *profit* of p_i whereas w_i is its *weight*. The two variables can be represented respectively as two functions of the local status of p_i, that is, $C[i]$. We have

$$\pi_i(C[i]) = \begin{cases} r_1, & C[i] = s_1 \\ 0, & C[i] = s_2 \end{cases} \quad \textit{and} \quad w_i(C[i]) = \begin{cases} r_2, & C[i] = s_1 \\ 0, & C[i] = s_2 \end{cases} \tag{17}$$

where r_1 and r_2 in (17) are two constants, which will be chosen uniformly at random from $[0, 10]$ in our experiments. In regard to each variable, a global average of it can be calculated using the protocol in Figure 1 and will be denoted respectively as $\bar{\pi}$ and $\bar{w}$. Based on the two average values, the OI to be maximized is further defined as a function of $C \in \mathcal{C}$ in what follows

$$OI(C) = \begin{cases} \bar{\pi}, & \bar{w} \leq W \\ W - \bar{w}, & \bar{w} > W, \end{cases} \tag{18}$$

where the *weight constraint* W is 1/3 in our experiments. According to (18), any status configuration C is *acceptable* if and only if $\bar{w} \leq W$. To solve this 0-1 knapsack problem, therefore, nodes must jointly identify acceptable status configurations first before any configuration C^* that increases $OI(C)$ to its maximum can be discovered.

Although our knapsack problems are theoretical by nature, its analogous to real-world problems in P2P systems and other distributed systems can be easily identified. As one recent example, Rivindra et al. had considered the problem of selecting nodes for performing transcoding tasks in P2P media streaming applications [Ravindra et al. 2009]. Given a number of video blocks of identical size for transcoding, each node in a P2P system must determine its local status as either *ordinary node* or *transcoding node*. In the latter case, the node will have to process as many video blocks as its local memory can hold using its local CPU power. The problem is therefore to *minimize* the total CPU

Table I. Performance of CESM under Varied M for 0-1 Knapsack Problems

M	No. of Comm. Per Sampling Round	Percentage of Optimal OI Achieved			Max OI after 1000 Sampling Rounds (%)
		80%	90%	95%	
$M = 2$	40	58.3 (16.6)	130.6 (21.9)	NA	94.8%
$M = 4$	80	65.9 (13.2)	133.3 (15.4)	236.7 (19.8)	98.0%
$M = 6$	120	74.9 (11.7)	109.0 (14.1)	139.8 (18.5)	100 %
$M = 10$	200	79.3 (11.2)	104.5 (12.8)	136.4 (20.1)	100%
$M = 20$	400	82.9 (10.7)	112.2 (11.1)	141.2 (17.8)	100%

processing time required, subject to the constraint that the total number of video blocks to be processed by all transcoding nodes must be no less than the given number of video blocks. As can be easily seen, this problem is essentially a 0-1 knapsack problem. It can be directly converted to the form as given in (18). Any algorithms designed for knapsack problems can therefore be applied to tackle this problem.

Several parameters are jointly involved in evaluating the performance of CESM, namely M, γ, T, μ, and ρ. Here M stands for the number of messages to be sent by every node in a P2P system in order for it to estimate the value of OI, according to the protocol in Figure 1. The meaning of other paramaters has been mentioned before. Please refer to Table X in the Appendix for more information.

As we mentioned in Section 3, system-wide estimation of $\bar{v}$ for any local variable v can be achieved highly efficiently through the protocol in Figure 1. Nevertheless, regardless of how large M is, the estimation of $\bar{v}$ can never be 100% accurate. The inherent error may further affect the correct updating of sampling probabilities in (12). Therefore, to find out the impact of M on the performance of CESM, we start our experiments by first varying the value of M while other parameters remain unchanged. Specifically, $\gamma = 0.01$, $T = 20$, $\mu = 0.05$, and $\rho = 0.01$. No nodes are allowed to join or leave the system, so N is fixed at 300. The results obtained are presented in Table I.

The main body of Table I contains the average number of sampling rounds required to obtain near-optimal solutions of the 0-1 knapsack problem, based on 25 independent simulations. The standard deviation of the experiment results is given inside the parentheses. The second column of this table shows the total number of communications to be performed by every node during each sampling round, with respect to varied settings of M. Meanwhile, the maximum OI achieved after 1000 sampling rounds is also indicated in the last column.

As can be seen from Table I, with small value of M, 80% of the optimal OI can be quickly achieved at early stages of the experiments. For example, when $M = 2$, each node in average needs to have $58.3 \times 40 = 2332$ separate communications before 80% of the optimal OI can be achieved. Notice that during each communication round, only a small list of values as indicated in Figure 1 will be sent through the network. So if any node can send 10 messages per second, which as we believe is a conservative assumption, in total we need to wait for only 233.3 seconds (less than 4 minutes).

After 80% of the optimal OI is achieved, the process to find close-to-optimal status configurations will start to slow down when $M = 2$. This is because each

Table II. Performance of CESM under Varied γ for 0-1 Knapsack Problems

Avg. No. of Sampling Rounds	Percentage of Optimal OI Achieved			Max OI after 1000 Sampling Rounds (%)
	80%	90%	95%	
$\gamma = 0.002$	78.9 (15.1)	101.4 (11.6)	108.9 (14.1)	99.4%
$\gamma = 0.005$	62.5 (8.2)	73.8 (9.3)	83.6 (10.5)	99.5%
$\gamma = 0.01$	53.5 (6.7)	64.9 (8.1)	75.4 (13.5)	100%
$\gamma = 0.02$	56.5 (7.7)	67.3 (8.2)	82.3 (13.7)	100%
$\gamma = 0.05$	55.3 (13.3)	85.1 (18.4)	113.3 (23.8)	99.2%

Table III. Performance of CESM Under Varied ρ for 0-1 Knapsack Problems

Avg. No. of Sampling Rounds	Percentage of Optimal OI Achieved			Max OI after 1000 Sampling Rounds (%)
	80%	90%	95%	
$\rho = 0.005$	82.5 (12.8)	109.2 (21.3)	223.3 (23.2)	97.2%
$\rho = 0.01$	63.0 (8.6)	75.8 (7.7)	86.4 (13.4)	98.1%
$\rho = 0.02$	53.5 (6.7)	64.9 (8.1)	75.4 (13.5)	100%
$\rho = 0.05$	70.6 (12.5)	106.7 (23.1)	822.5 (58.3)	95.3%

node can only obtain a fairly localized average of the two variables π and w defined in (17) if M is too small. Accordingly, a node's estimation of OI in (18) is also locally biased. In view of this, we therefore need to increase the value of M in order to have more accurate estimation of $\bar{\pi}$ and $\bar{w}$. In particular, as indicated in Table I, when $M = 6$, optimal OI can be eventually achieved after 1000 sampling rounds. Meanwhile, the number of sampling rounds required to obtain 90% and 95% of the optimal OI can also be reduced (in comparison to the case of $M = 2$ and $M = 4$) without generating too many communications. For this reason, we consider 6 as a suitable value for M. It will be used throughout the rest of our experiments.

After identifying the suitable value of M, the effectiveness of CESM for 0-1 knapsack problems was further evaluated under varied γ with $T = 100$, $\mu = 0.05$, and $\rho = 0.02$. The results are presented in Table II. As indicated in the table, it is not desirable for γ to be too small such as $\gamma = 0.002$. Since knapsack problems are not unimodal, using too small γ is prone to *immature convergence*. Meanwhile, too large γ is also undesirable since it will slow down the convergence towards C^*. The most suitable values for γ in Table II are 0.01 and 0.02. Other than γ, the effect of ρ was also examined with the results presented in Table III.

The results in Table III are obtained when $\gamma = 0.01$, $T = 100$, and $\mu = 0.05$. The most suitable value of ρ is 0.02, since with this setting, 75.4 sampling rounds are required in average to reach 95% of the optimal OI, comparatively less than other settings of ρ. Meanwhile only with this setting will C^* be consistently identified before the experiments end. In addition to ρ, the impact of T was studied as well and the results are summarized in Table IV. As shown in the table, by increasing T, the average number of sampling rounds required for finding near-optimal configurations can be significantly reduced. One obvious reason is that, to solve NP-hard problems such as our knapsack

Table IV. Performance of CESM under Varied T for 0-1 Knapsack Problems

Avg. No. of Sampling Rounds	Percentage of Optimal OI Achieved			Max OI after 1000 Sampling Rounds (%)
	80%	90%	95%	
$T = 10$	121.2 (19.8)	159.6 (25.2)	332.8 (71.5)	95.5%
$T = 30$	78.2 (17.7)	103.3 (15.2)	129.8 (21.1)	98.7 %
$T = 50$	76.6 (10.2)	90.9 (13.8)	124.7 (19.5)	99.5 %
$T = 80$	61.5 (6.8)	70.7 (8.7)	82.4 (11.4)	100%
$T = 200$	41.8 (5.0)	49.7 (5.6)	58.8 (7.6)	100%

Table V. Performance of GA under Varied Settings of τ_c and τ_m for 0-1 Knapsack Problems

		0-1 Knapsack Problem (% of optimal OI)			
τ_c	τ_m	Generation 250	Generation 500	Generation 750	Generation 1000
0.5	0.01	71.5%	81.0%	83.1%	84.7%
	0.05	78.2%	90.1%	92.1%	93.5%
	0.1	83.9%	95.0%	96.7%	97.2%
0.7	0.01	72.9%	80.0%	82.6%	84.2%
	0.05	82.1%	90.4%	93.0%	94.1%
	0.1	88.3%	95.9%	96.6%	97.1%
0.9	0.01	72.9%	79.8%	82.3%	83.0%
	0.05	83.5%	91.9%	93.5%	94.3%
	0.1	90.5%	96.4%	97.2%	97.7%

problem, it is very important for nodes to accurately estimate the importance of choosing any local status.

In order to show the effectiveness of CESM, a comparison study is performed by experimentally evaluating the performance of GA on knapsack problems under varied settings of τ_c and τ_m. The experiment results can be found in Table V. Comparing with Table II, Table V indicates that CESM is more effective on knapsack problems than GA, as fewer samples are required by CESM to discover near-optimal status configurations. Moreover, for two separate settings of γ in Table II, C^* can be eventually identified (refer to the last column of Table II), whereas GA has failed to find C^* for all experiments.

For all the previous experiments in this subsection, nodes are not allowed to join or leave the P2P system. In reality, however, P2P systems often demonstrate high rates of churn. As a matter of fact, before any near-optimal status configurations can be discovered, the set of nodes in the system might have already changed several times. Consequently, for any node in the system, its local status, which was deemed suitable before, may not be suitable anymore if its nearby neighbors changed. Nevertheless, as our previous experiments showed, at least 80% of the global optimum can be quickly achieved under reasonable settings of a few parameters, such as $M = 6$, $T = 30$, $\gamma = 0.01$, $\mu = 0.05$, and $\rho = 0.01$. So if a large proportion of nodes (set to 80% of all nodes in the experiments) tend to stay in the system, every node can still manage to find its suitable local status before a majority of its neighbors changed.

Table VI. Performance of CESM under Varied Settings of P_l and P_j for 0-1 Knapsack Problems

$\mathbf{P_j}$ (probability to join)	$\mathbf{P_l}$ (probability to leave)		
	0.01	0.05	0.1
0.05	90.8%	90.1%	88.1%
0.2	91.6%	90.2%	89.2%
0.5	93.1%	91.0%	89.9%

Experiments have been conducted under varied levels of node dynamicity in order to show its impact on the performance of CESM. Specifically, if a node is currently in the system, with a probability of P_l, it will leave the system in the next sampling round. Once it left, its neighbors lost the connection to it. Meanwhile, if a node being simulated has left the system, with a probability of P_j, it will join back to the system in the next sampling round. Upon joining back, it will resume its connections to its neighbors if they are still in the system.

Nodes' behaviors in real systems can be very complicated. For example, a node might be malfunctioning and constantly sending wrong information to its neighbors. We will not consider such situations in the experiments. At any time, the number of nodes in the system cannot fall below 80% of the original number of nodes in the system, which is 300. Based on the aforesaid settings, we have obtained a collection of experiment results as summarized in Table VI.

The main body of Table VI shows the average percentage of the optimal OI achieved during the experiments from 200 sampling rounds to 1000 sampling rounds. As the table indicates, around 90% of the optimal OI can be achieved in a dynamic P2P system where nodes are free to leave and later rejoin the system. Specifically, with higher value of P_j and lower value of P_l, the system tends to be more stable and accordingly the average percentage achieved is also increased. In general, CESM is not very sensitive to P_j and P_l, as the maximum difference of the results in Table VI is only at 5%. This is because we do not allow more than 20% of nodes to leave the system. It is to be noticed that CESM, with its current form, may not be suitable for handling a P2P system with an excessive number of joining/leaving nodes. New mechanisms might be necessary to reduce the impact of system dynamicity and will be treated here as future work.

5.2 A P2P Information Monitoring Problem

In order to strengthen control and to support resource-sensitive applications, the local activities of nodes (i.e., average CPU load, memory usage, etc.) in a P2P application may often need to be closely monitored [Liang et al. 2007; Yalagandula and Dahlin 2004]. Following a *federated system architecture* as presented by Liang et al. [2007], a P2P system is viewed as a collection of two types of nodes: (1) *ordinary overlay nodes* that execute different application tasks, and (2) *management nodes* that monitor ordinary overlay nodes and perform system management tasks, such as job scheduling and resource allocation. As

noted by Liang et al., providing efficient information management is generally a challenging task that requires optimizing some system-wide performance index, the definition of which is given as follows.

Similar with knapsack problems, a node in an information monitoring system can choose two alternative statuses s_1 and s_2. Let s_1 correspond to ordinary overlay nodes and s_2 represent management nodes. Each node connects to the rest of a P2P system through a group of edges and every edge e_{ij} is characterized by two attributes: the connection bandwidth $b(e_{ij})$ measured in Mbps and the Round-Trip Time (RTT) $d(e_{ij})$ measured in milliseconds. For any two nodes p_i and p_j in the system, the number of edges along the shortest path between them is used to indicate their distance. It is assumed in this subsection that every overlay node is *affiliated* to its nearest management node. This assumption can be realized easily and will not be elaborated here. Suppose that p_i is a management node, the group of overlay nodes affiliated to it is denoted as $Aff(p_i)$.

For every node $p_j \in Aff(p_i)$, the shortest path between p_i and p_j is represented as a sequence of edges from p_i to p_j, namely

$$Path(p_i, p_j) = \left\{e_{ik_1}, e_{k_1k_2}, \ldots, e_{k_nj}\right\}.$$

The first edge along this path, $Path(p_i, p_j)^1 = e_{ik_1}$, refers to the edge used by p_i to reach p_j. The RTT of this path can be determined as

$$d(p_i, p_j) = \sum_{e \in Path(p_i, p_j)} d(e). \tag{19}$$

Let

$$\mathcal{E}_i := \{e | \exists p_j \in Aff(p_i) \,\&\, e = Path(p_i, p_j)^1\}$$

be the set of edges used by p_i to reach any of its affiliates. In addition, $\forall e \in \mathcal{E}_i$, let

$$\mathcal{P}_i(e) = \{p_j | p_j \in Aff(p_i) \,\&\, e = Path(p_i, p_j)^1\}$$

be the set of affiliated nodes p_j that are reachable from a specific edge e of p_i.

To ensure fairness, assume that node p_i will allocate its networking bandwidth as evenly as possible among its affiliated nodes, then the bandwidth allocated for every node $p_j \in Aff(p_i)$ to access p_i, that is, $b(p_i, p_j)$, can be obtained as a solution of a simple constrained optimization problem.

$$\forall e \in \mathcal{E}_i,\ \min \left(\sum_{p_j \in \mathcal{P}_i(e), p_k \in \mathcal{P}_i(e)} \left(b(p_i, p_j) - b(p_i, p_k)\right)^2 \right)$$

$$\textit{such that} \quad b(p_i, p_j) \le \min_{e' \in Path(p_i, p_j)} b(e') \ \textit{and} \ \sum_{p_j \in \mathcal{P}_i(e)} b(p_i, p_j) = b(e). \tag{20}$$

This problem is solvable if

$$\forall e \in \mathcal{E}_i, \quad \sum_{p_j \in \mathcal{P}_i(e)} \left(\min_{e' \in Path(p_i, p_j)} b(e') \right) \geq b(e). \tag{21}$$

The solution of this problem can be determined independently by every management node in the P2P system and will be made known subsequently to all affiliated overlay nodes. In association with (19) and (20), each affiliated node $p_j \in Aff(p_i)$ maintains locally a variable q_j as defined next.

$$q_j = b(p_i, p_j) + \frac{1000}{d(p_i, p_j)} \tag{22}$$

Here, q_j measures the quality of the communication path between p_j and its management node p_i. According to (22), it is desirable for the communication path to have more bandwidth. Meanwhile the communication delay is expected to be as low as possible. Based on (22), the system-wide average of q_j, namely $\bar{q}$, can be efficiently estimated by every node by using the protocol in Figure 1. Based on $\bar{q}$, OI is further defined as

$$\forall C \in \mathcal{C}, \ OI(C) = \bar{q} - 10 \times \frac{\sum_{i=1}^{N} \lambda(C[i] = s_2)}{N}. \tag{23}$$

The second term in (23) refers to the *density* of management nodes in the P2P application and will be calculated similarly as $\bar{q}$. The intuition behind (23) is that each management node brings some cost to the system since its resources will be consumed by monitoring and management tasks instead of application-specific tasks. Hence the increment of $\bar{q}$ should not be achieved at the price of employing so many management nodes.

Driven by (23), experiments have been conducted in a P2P system generated by BRITE's Waxman model [Medina et al. 2001]. The average bandwidth of edges in the system is 10 Mbps. For every edge, its bandwidth is only allowed to deviate from 10 Mbps slightly such that (21) can be satisfied. The RTT of edges in the system follows the normal distribution. The average RTT is 300ms and the standard deviation is 145. The minimum RTT in the P2P system is 10ms and the maximum RTT is 700ms.

Since C^* is not known in advance, the maximum OI obtained by both CESM and GA in the experiments will not be presented in the percentage form. Table VII summarizes the average performance of CESM under varied settings of γ, ρ, and T, when $M = 6$ and $\mu = 0.05$, based on 25 independent simulations. The standard deviation of the experiment results is given inside the parentheses.

Similar observations as those in Section 5.1 have been presented in Table VII. In short, setting γ to a small value will increase the chances of discovering high OI status configurations. Meanwhile, the best choice of ρ is still 0.02. It is also unnecessary for T to be too large such as 200. Instead, any value from the range $[30, 50]$ seems more suitable, since similar performance as when $T \geq 80$ can be achieved without generating too many samples per sampling round.

Similar as the knapsack problem, comparison experiments using GA have been performed with the results shown in Table VIII. As evidenced in this table, CESM is as effective as GA on P2P information monitoring problems. In

Table VII. Performance of CESM for P2P Information Monitoring Problems

Avg. OI Achieved	No. of Sampling Rounds			
	250	500	750	1000
$\gamma = 0.002$	4.841 (0.014)	4.847 (0.014)	4.853 (0.010)	4.854 (0.009)
$\gamma = 0.005$	4.834 (0.015)	4.841 (0.0138)	4.846 (0.0137)	4.849 (0.0135)
$\gamma = 0.01$	4.824 (0.017)	4.839 (0.012)	4.844 (0.010)	4.847 (0.010)
$\gamma = 0.02$	4.802 (0.018)	4.821 (0.017)	4.830 (0.016)	4.834 (0.014)
$\gamma = 0.05$	4.724 (0.025)	4.766 (0.016)	4.784 (0.012)	4.789 (0.012)
$\rho = 0.005$	4.812 (0.016)	4.826 (0.017)	4.830 (0.016)	4.834 (0.016)
$\rho = 0.01$	4.818 (0.016)	4.833 (0.015)	4.837 (0.014)	4.839 (0.014)
$\rho = 0.05$	4.813 (0.014)	4.826 (0.012)	4.833 (0.010)	4.838 (0.008)
$T = 30$	4.790 (0.020)	4.816 (0.011)	4.827 (0.010)	4.831 (0.009)
$T = 50$	4.822 (0.019)	4.837 (0.013)	4.842 (0.010)	4.845 (0.010)
$T = 80$	4.825 (0.020)	4.838 (0.015)	4.845 (0.013)	4.846 (0.013)
$T = 200$	4.833 (0.018)	4.842 (0.014)	4.844 (0.013)	4.848 (0.011))

Table VIII. Performance of GA for P2P Information Monitoring Problems

		Highest OI Obtained			
τ_c	τ_m	Generation 250	Generation 500	Generation 750	Generation 1000
0.5	0.01	4.432	4.721	4.785	4.803
	0.05	4.643	4.754	4.798	4.812
	0.1	4.768	4.813	4.821	4.829
0.7	0.01	4.582	4.786	4.818	4.821
	0.05	4.743	4.825	4.826	4.827
	0.1	4.771	4.834	4.835	4.835
0.9	0.01	4.613	4.801	4.825	4.831
	0.05	4.775	4.831	4.839	4.841
	0.1	4.789	4.836	4.845	4.846

Table IX. Performance of CESM under Varied Settings of P_l and P_j for P2P Information Monitoring Problems

$\mathbf{P_j}$ (Probability to Join)	$\mathbf{P_l}$ (Probability to Leave)		
	0.01	0.05	0.1
0.05	4.822	4.817	4.809
0.2	4.826	4.820	4.813
0.5	4.830	4.825	4.821

particular, for varied settings of CESM, the maximum OI obtained by CESM is consistently higher than those obtained by GA. Moreover, high OI configurations can be discovered more quickly through CESM, especially at early stages of the experiments.

To end this subsection, experiments have also been performed when nodes are free to leave and later rejoin the P2P system. We follow exactly the same

experiment settings as those used for testing the knapsack problem. The results obtained are summarized in Table IX. As can be verified from the table, system dynamicity will not seriously affect the performance of CESM. Especially, when the system is relatively stable with high value of P_j and low value of P_l, the average OI achieved in Table IX is roughly identical to those reported in Table VII.

6. CONCLUSION

This article focused on addressing a common problem of self-organizing the local status of nodes in P2P-like applications. After introducing the concept of status configuration, a mathematical correspondence was established between status configuration and an optimization index (OI), which serves as a unified abstraction of system design objectives. A protocol for distributed computation of OI was presented as a justification of the practicality of our problem model. Driven by OI and inspired by the cross-entropy optimization method, a cross-entropy-driven self-organization mechanism (CESM) was further proposed.

CESM allows nodes in P2P systems to independently choose their suitable local status based on feedback from their past choices. The mechanism is purely distributed and exhibits the self-organization property. Theoretical analysis of CESM showed that, at least for a common type of problem, the optimal status configuration can be identified very efficiently without generating too much communication cost. Experiment evaluations have also been performed on two representative problems, namely the 0-1 knapsack problem and the P2P information monitoring problem. The experiment results clearly demonstrated the effectiveness of CESM as a general-purpose algorithm suitable for solving the self-organization problems considered in this article.

To conclude this work, it is noticed that spaces exist to further enhance the effectiveness of CESM, especially when a P2P system is highly dynamic with an excessive number of joining/leaving nodes. It is also interesting to evaluate and improve the performance of CESM in cases where nodes frequently change their neighbors while determining their suitable local status. These and other issues will be treated as future works.

APPENDIXES

APPENDIX I

Table X. Notation Index

Notation	Description
p_i	A node in a P2P application, where i is the node ID.
e_{ij}	An edge between two nodes p_i and p_j.
$\mathcal{P}$	The set of all nodes in a P2P application.
$\mathcal{E}$	The set of all edges in a P2P application.
N	The number of nodes in a P2P application.
s_j	A local status that any node can take.
$\mathcal{S}$	A collection of all available local status.
C_t	The status configuration of a P2P application at time t.
$\mathcal{C}$	The set of all possible status configurations.
$C_t[i]$	The local status of node p_i at time t.
OI	The system-wide optimization index to be maximized.
C^*	The optimal status configuration.
v_i	A local variable maintained by node p_i.
$\bar{v}$	The system-wide average value of variable v, which is maintained locally by every node in a P2P application.
M	The number of communication rounds requested by the protocol in Figure 1.
γ	The Boltzmann temperature.
$H(C, \gamma)$	The Boltzmann function defined on C (the status configuration) and γ.
$\delta(\gamma)$	The partition function.
Θ^*	The importance sampling density.
Θ	The sampling density used in practice.
$\theta_i(s_j)$	The sampling probability for node p_i to choose s_j as its local status.
$D(\Theta_1, \Theta_2)$	The degree of similarities between sampling densities Θ_1 and Θ_2 based on cross-entropy measure.
$\lambda(cond)$	A function that returns 1 if the given condition *cond* is true and returns 0 otherwise.
T	The number of samples required for each updating of sampling probabilities in (12).
$Ent(P_i)$	An entropy measure over the sampling probabilities used by node p_i.
μ	A threshold of the entropy measure used in Figure 3.
ρ	The probability for a node to choose its local status uniformly at random, as used in Figure 3.
τ_m	Mutation rate of GA.
τ_c	Crossover rate of GA.

APPENDIX II

In this Appendix, we give the proof of Theorem 1.

PROOF. To prove Theorem 1, it is acceptable for us to focus on just one node p_i since all nodes in a P2P system follow exactly the same procedure to update their sampling probabilities. As the sampling probabilities will be initialized uniformly as in Figure 2, the probability of discovering C^* at the time when the self-organization process just started is

$$\Theta^{(0)}(C^*) = \frac{1}{\|\mathcal{C}\|}.$$

$\|\mathcal{C}\|$ stands for the cardinality of $\mathcal{C}$. As explained shortly, throughout time, this probability is expected to be continuously increased. Since ϵ is arbitrarily small, we can find a small enough γ such that (14) is valid with probability 1. Without loss of generality, assume that $C^*[i] = s_1$. Let $\theta_i(s_1)^{(h)}$ denote the probability of sampling s_1 during the hth sampling round. The corresponding sampling density is represented by $\Theta^{(h)}$. According to (12) the conditional expectation of $\theta_i(s_1)^{(h+1)}$ can be evaluated in practice as

$$\begin{aligned}
& E\left[\theta_i(s_1)^{(h+1)} | \Theta^{(h)}\right] \\
&= E\left[\frac{\sum_{t=1}^{T} H(C_t, \gamma) \cdot \lambda(C_t[i] = s_1)}{\sum_{t=1}^{T} H(C_t, \gamma)} | \Theta^{(h)}\right] \\
&\cong \frac{\bar{H}(i, s_1, \gamma, \Theta^{(h)}) \cdot \theta_i(s_1)^{(h)}}{\sum_{k=1}^{m} \bar{H}(i, s_k, \gamma, \Theta^{(h)}) \cdot \theta_i(s_k)^{(h)}} \\
&\geq \frac{\bar{H}(i, s_1, \gamma, \Theta^{(h)}) \cdot \theta_i(s_1)^{(h)}}{\bar{H}(i, s_1, \gamma, \Theta^{(h)}) \cdot \theta_i(s_1)^{(h)} + \theta_i(s_k)^{(h)} \cdot (1-\alpha) \cdot \left(\sum_{k=2}^{m} \bar{H}(i, s_1, \gamma, \Theta^{(h)}) \cdot \theta_i(s_k)^{(h)}\right)} \\
&= \frac{\theta_i(s_1)^{(h)}}{\theta_i(s_1)^{(h)} + (1-\alpha) \cdot \sum_{k=2}^{m} \theta_i(s_k)^{(h)}} \\
&= \frac{\theta_i(s_1)^{(h)}}{\theta_i(s_1)^{(h)} + (1-\alpha) \cdot (1 - \theta_i(s_1)^{(h)})}.
\end{aligned} \tag{24}$$

The inequality in (24) is established based on (14) for unimodal problems. By taking expectations of $\theta_i(s_j)^{(h)}$ repeatedly with each repetition follows (24), and using the fact that $\Theta^{(1)}$ is the *uniform sampling density*, we obtain

$$E[\theta_i(s_1)^{(h)}] \geq \frac{1}{1 + (m-1) \cdot (1-\alpha)^{h-1}}. \tag{25}$$

(25) implies that

$$\lim_{h \to \infty} E[\theta_i(s_1)^{(h)}] = 1. \tag{26}$$

Therefore, node p_i will eventually choose local status s_1 with probability 1. In other words, the P2P system as a whole will self-organize towards the optimal

configuration C^* almost certainly. Specifically, it is easy to verify that, after

$$h = \log_{1-\alpha}\left(\frac{\varepsilon\cdot\delta}{m-1}\right)+1 = \log_{\frac{1}{1-\alpha}}\left(\frac{\sqrt{m-1}}{\varepsilon}\right)+\log_{\frac{1}{1-\alpha}}\left(\frac{\sqrt{m-1}}{\delta}\right)+1 \tag{27}$$

sampling rounds

$$E[\theta_i(s_1)^{(h)}] \geq \frac{1}{1+\varepsilon\cdot\delta}. \tag{28}$$

Consequently,

$$E[1-\theta_i(s_1)^{(h)}] \leq \frac{\varepsilon\cdot\delta}{1+\varepsilon\cdot\delta}. \tag{29}$$

By Markov's inequality,

$$\begin{aligned}\Pr\left(\left|1-\theta_i(s_1)^{(h)}\right| \leq \varepsilon\right) &\geq 1-\frac{E[1-\theta_i(s_1)^{(h)}]}{\varepsilon}\\ &\geq 1-\frac{\delta}{1+\varepsilon\cdot\delta}\\ &\geq 1-\delta.\end{aligned} \tag{30}$$

Therefore, when $O(\log\left(\frac{1}{\varepsilon}\right)+\log\left(\frac{1}{\delta}\right))$ sampling rounds have been completed, the probability for $\theta_i(s_1)$ to be greater than $1-\varepsilon$ is at least $1-\delta$. At this moment, the probability for the P2P system as a whole to sample the optimal status configuration C^* is lower bounded by

$$(1-\varepsilon)^N.$$

Accordingly, the expected number of samples requested before identifying C^* is at most

$$\sum_{k=1}^{\infty}(k\times(1-\varepsilon)^N\times(1-(1-\varepsilon)^N)^{k-1}) = \frac{1}{(1-\varepsilon)^N}. \tag{31}$$

Provided that

$$\varepsilon = 1-(\log N)^{-1/N}$$

then (31) becomes $\log N$. Notice that $\log\left(\frac{1}{\varepsilon}\right) \leq \log N$ for large enough N, (e.g., $N > 50$). In association with the previous discussion, we conclude that, with a probability of at least $1-\delta$, the expected number of sampled status configurations, which is sufficient for discovering C^*, is

$$O\left(\log N+\log\frac{1}{\delta}\right).$$

□

REFERENCES

ALTMAN, E., JIMENEZ, T., AND KOOLE, G. 2001. On optimal call admission control in resource-sharing system. *IEEE Trans. Comm. 49*, 9, 1659–1668.

ANDROUTSELLIS-THEOTOKIS, S. AND SPINELLIS, D. 2004a. A survey of peer-to-peer content distribution technologies. *ACM Comput. Surv. 36*, 4, 335–371.

ANDROUTSELLIS-THEOTOKIS, S. AND SPINELLIS, D. 2004b. A survey of peer-to-peer content distribution technologies. *ACM Comput. Surv. 36*, 4, 335–371.

ANGLUIN, D., ASPNES, J., DIAMADI, Z., FISCHER, M. J., AND PERALTA, R. 2004. Computation in networks of passively mobile finite-state sensors. In *Proceedings of the 23rd Annual ACM Symposium on Principles of Distributed Computing*. 290–299.

BÄCK, T. AND SCHWEFEL, H. P. 1993. An overview of evolutionary algorithms for parameter optimization. *Evolut. Comput. 1*, 1, 1–23.

BONABEAU, E., DORIGO, M., AND THERAULAZ, G. 1999. *Swarm Intelligence: From Natural to Artificial Systems*. Oxford University Press.

BRIDGEWATER, J. S. A., BOYKIN, P. O., AND ROYCHOWDHURY, V. P. 2007. Balanced overlay networks (bon): An overlay technology for decentralized load balancing. *IEEE Trans. Parallel Distrib. Syst. 18*, 8, 1122–1133.

CHAKRAVARTI, A. J., BAUMGARTNER, G., AND LAURIA, M. 2005. The organic grid: Self-Organizing computation on a peer-to-peer network. *IEEE Trans. Syst. Man Cybernet. 35*, 3, 373–384.

CHEN, G., LOW, C. P., AND YANG, Z. H. 2008. Coordinated services provision in peer-to-peer environments. *IEEE Trans. Parallel Distrib. Syst. 19*, 4, 433–446.

CICIRELLO, V. A. AND SMITH, S. F. 2004. Wasp-like agents for distributed factory coordination. *Auton. Agents Multi-Agent Syst. 8*, 237–266.

CUENCA-ACUNA, F. M. AND NGUYEN, T. D. 2004. Self-managing federated services. In *Proceedings of the 23rd IEEE International Symposium on Reliable Distributed Systems*.

FOGEL, D. B. 1995. *Evolutionary Computation: Toward a New Philosophy of Machine Intelligence*. IEEE Press, New York.

FORESTIERO, A., MASTROIANNI, C., AND MEO, M. 2009. Self-Chord: A bio-inspired algorithm for structured p2p systems. In *Proceedings of the 9th IEEE International Symposium on Cluster Computing and the Grid*.

GEDIK, B. AND LIU, L. 2005. A scalable peer-to-peer architecture for distributed information monitoring applications. *IEEE Trans. Comput. 54*, 6, 767–782.

GHANEA-HERCOCK, R. A., WANG, F., AND SUN, Y. 2006. Self-Organizing and adaptive peer-to-peer network. *IEEE Trans. Syst. Man Cybernet. B 36*, 6, 1230–1236.

GOLDBERG, D. E. 1989. *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison-Wesley Professional.

GOLDSRMIDT, G. AND YEMIN, Y. 1995. Decentralizing control and intelligence in network management. In *Proceedings of the 4th International Symposium on Integrated Network Management*.

GU, X. AND NAHRSTEDT, K. 2006. Distributed multimedia service composition with statistical qos assurances. *IEEE Trans. Multimed. 8*, 1, 141–151.

GU, X., WEN, Z., AND YU, P. S. 2007. BridgeNet: An adaptive multi-source stream dissemination overlay network. In *Proceedings of the 26th IEEE International Conference on Computer Communications (InfoCom*. 2586–2590.

GUPTA, R., SEKHRI, V., AND SOMANI, A. K. 2006. Compup2p: An architecture for internet computing using peer-to-peer networks. *IEEE Trans. Parall. Distrib. Syst. 17*, 11, 1306–1320.

HEEGAARD, P. E., HELVIK, B. E., AND WITTNER, O. J. 2008. The cross entropy ant system for network path management. *Telektronikk 104*, 01, 19–40.

JIN, H., NING, X., AND CHEN, H. 2006. Efficient search for peer-to-peer information retrieval using semantic small world. In *Proceedings of Internation Conference on World Wide Web Poster (WWW)*. 1003–1004.

KARGER, D. R. AND RUHL, M. 2004. Simple efficient load balancing algorithms for peer-to-peer systems. In *Proceedings of the 16th Annual ACM Symposium on Parallelism in Algorithms and Architectures*.

KEMPE, D., DOBRA, A., AND GEHRKE, J. 2003. Gossip-based computation of aggregate information. In *Proceedings of the 44th Annual IEEE Symposium on Foundations of Computer Science*.

KEYANI, P., LARSON, B., AND SENTHIL, M. 2002. Peer pressure: Distributed discovery from attacks in peer-to-peer systems. In *Proceedings of the International Workshop on Peer-to-Peer Computing*. 306–320.

KLEINBERG, J. 2000. The small-world phenomenon: An algorithmic perspective. In *Proceedings of the 32nd ACM Symposium on Theory of Computing*.

KO, S. Y., GUPTA, I., AND JO, Y. 2008. A new class of nature-inspired algorithms for self-adaptive peer-to-peer computing. *ACM Trans. Auton. Adapt. Syst. 3*, 3.

LEDLIE, J., TAYLOR, J. M., SERBAN, L., AND SELTZER, M. 2002. Self-Organization in peer-to-peer systems. In *Proceedings of the 10th Workshop on ACM SIGOPS European Workshop*. 125–132.

LI, M., LEE, W., AND SIVASUBRAMANIAM, A. 2004. Semantic small world: An overlay network for peer-to-peer search. In *Proceedings of the 12th IEEE International Conference on Network Protocols (ICNP)*.

LIANG, J., GU, X., AND NAHRSTEDT, K. 2007. Self-configuring information management for large-scale service overlays. In *Proceedings of the 26th IEEE International Conference on Computer Communications (InfoCom)*. 472–480.

LUA, E. K., CROWCROFT, J., PIAS, M., SHARMA, R., AND LIM, S. 2005. A survey and comparison of peer-to-peer overlay network schemes. *IEEE Comm. Surv. Tutor. 7*, 2, 72–93.

MANKU, G. S., BAWA, M., AND RAGHAVAN, P. 2003. Symphony: Distributed hashing in a small world. In *Proceedings of the 4th USENIX Symposium on Internet Technologies and Systems (USITS)*.

MARTELLO, S. AND TOTH, P. 1990. *Knapsack Problems: Algorithms and Computer Implementations*. John Wiley & Sons.

MEDINA, A., LAKHINA, A., MATTA, I., AND BYERS, J. 2001. BRITE: An approach to universal topology genration. In *Proceedings of the IEEE International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunications Systems*.

MU, S., CHI, C. H., LIU, L., AND WANG, H. G. 2006. Object placement and caching strategies on AN. P2P*. In *Advances in Web-Age Information Management*. Springer, 182–192.

NOCEDAL, J. AND WRIGHT, S. 2009. *Numerical Optimization*. Springer.

PANDURANGAN, G., RAGHAVAN, P., AND UPFAL, E. 2001. Building low-diameter P2P networks. In *Proceedings of the 42nd IEEE Symposium on Foundations of Computer Science*. 492–499.

PAPADIMITRIOU, C. H. AND STEIGLITZ, K. 1998. *Combinatorial Optimization: Algorithms and Complexity*. Dover Publications.

PROKOPENKO, M., GERASIMOV, V., AND TANEV, I. 2006. Evolving spatiotemporal coordination in a modular robotic system. In *Proceedings of the 9th International Conference on the Simulation of Adaptive Behavior*, S. Nolfi et al., Eds. 558–569.

RAMASWAMY, L., GEDIK, B., AND LIU, L. 2005. A distributed approach to node clustering in decentralized peer-to-peer networks. *IEEE Trans. Parall. Distrib. Syst. 16*, 9, 814–829.

RAVINDRA, G., KUMAR, S., AND CHINTADA, S. 2009. Distributed media transcoding using a p2p network of set top boxes. In *Proceedings of the 6th IEEE Consumer Communications and Networking Conference*.

RUBINSTEIN, R. Y. AND KROESE, D. P. 2004. *The Cross-Entropy Method, A Unified Approach to Combinatorial Optimization, Monte-Carlo Simulation and Machine Learning*. Springer.

RUBINSTEIN, R. Y. AND MELAMED, B. 1998a. *Efficient Simulation and Modeling*. John Wiley & Sons.

RUBINSTEIN, R. Y. AND MELAMED, B. 1998b. *Modern Simulation and Modeling*. Wiley-Interscience.

RUDIN, W. 1976. *Principles of Mathematical Analysis*, 3rd Ed. McGraw-Hill.

STOICA, I., MORRIS, R., LIBEN-NOWELL, D., KARGER, D. R., KAASHOEK, M. F., DABEK, F., AND BALAKRISHNAN, H. 2003. Chord: A scalable peer-to-peer lookup protocol for internet applications. *IEEE/ACM Trans. Netw. 11*, 1, 17–32.

SU, M. S., THULASIRAMAN, K., AND DAS, A. 2002. A scalable on-line multilevel distributed network fault detectioflonitoring system based on the snmp protocol. In *Proceedings of the IEEE Global Telecommunications Conference*. 1960–1964.

WITTNER, O., HEEGAARD, P. E., AND HELVIK, B. E. 2003. Scalable distributed discovery of resource paths in telecommunication networks using cooperative ant-link agents. In *Proceedings of the Congress on Evolutionary Computation*. 1456–1465.

YALAGANDULA, P. AND DAHLIN, M. 2004. A scalable distributed information management system. In *Proceedings of the Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*. 379–390.

YAO, X. AND XU, Y. 2006. Recent advances in evolutionary computation. *Int. J. Comput. Sci. Techn. 21*, 1, 1–18.

ZHANG, H., GOEL, A., AND GOVINDAN, R. 2004. Using the small-world model to improve Freenet performance. *Comput. Netw. 46*, 555–574.

ZLOCHIN, M., BIRATTARI, M., MEULEAU, N., AND DORIGO, M. 2004. Model-Based search for combinatorial optimization: A critical survey. *Ann. Oper. Res. 131*, 1-4, 373–395.

Received July 2009; revised July 2010; accepted August 2010