Agent-Based Load Balancing on Homogeneous Minigrids: Macroscopic Modeling and Characterization

Jiming Liu, *Senior Member*, *IEEE*, Xiaolong Jin, and Yuanshi Wang

Abstract—In this paper, we present a macroscopic characterization of agent-based load balancing in homogeneous minigrid environments. The agent-based load balancing is regarded as agent distribution from a macroscopic point of view. We study two quantities on minigrids: the number and size of teams where agents (tasks) queue. In macroscopic modeling, the load balancing mechanism is characterized using differential equations. We show that the load balancing we concern always converges to a steady state. Furthermore, we show that load balancing with different initial distributions converges to the same steady state gradually. Also, we prove that the steady state becomes an even distribution if and only if agents have complete knowledge about agent teams on minigrids. Utility gains and efficiency are introduced to measure the quality of load balancing. Through numerical simulations, we discuss the utility gains and efficiency of load balancing in different cases and give a series of analysis. In order to maximize the utility gain and the efficiency, we theoretically study the optimization of agents' strategies. Finally, in order to validate our proposed agent-based load balancing mechanism, we develop a computing platform, called *Simulation System for Grid Task Distribution* (SSGTD). Through experimentation, we note that our experimental results in general confirm our theoretical proofs and numerical simulation results from the proposed equation system. In addition, we find a very interesting phenomenon, that is, agent-based load balancing mechanism is topology-independent.

Index Terms—Homogeneous minigrids, load balancing, task distribution, agents, macroscopic modeling, steady states, convergence, grid simulation.

1 Background

In order to meet the increasing demand of large-scale scientific computation in the fields of life sciences, biology, physics, and astronomy, the notion of "computational grid" was proposed in mid 1990s [1], [2], [3], [4]. It has been observed that computers (such as PCs, workstations, and clusters) in the Internet are often idle. Grid computing aims to integrate idle computational power over the Internet and provide powerful computation capability for users all over the world [1], [2], [3], [5].

Since a grid connects numerous geographically distributed computers, and tasks are submitted to grid nodes in a distributed fashion, an important issue is how to evenly distribute submitted tasks to nodes. This is a *load balancing* problem, one of the scheduling problems on the grid. By solving this problem, we can optimally utilize computational resources of the grid. In this paper, we will propose an agent-based load balancing mechanism.

1.1 Scheduling on Grids

The scheduling problem on grids has been widely studied [6], [7], [8]. Many schedulers for grid computing have been developed [9], such as AppLeS [10], [11], Nimrod-G [12], GrADs [13], [14], and Condor-G [15]. The scheduling issues on grids lie in several aspects, among which *resource allocation* [6], i.e., how to allocate computational resources (e.g., CPU-hours, storage, and network bandwidth) to submitted tasks, and *task allocation* [7], [8], i.e., how to allocate tasks to different nodes, attract most attention. In [6], Galstyan et al. proposed an agent-based resource allocation model for grid computing. Their model is based on the reinforcement learning technique, and consequently can be adaptive to a dynamically changing grid environment.

In [7], [8], [16], [17], Casanova et al. have studied the task allocation problem in a grid environment, where the submitted task is arbitrarily divisible. In other words, a task can be divided into arbitrary chunks. Their scheduling mechanism assumed a *master/worker* architecture, i.e., a *master*, acting as a scheduler, is responsible for dividing the submitted tasks and allocating the obtained task chunks to different *workers*. In their work, they paid special attention to task transfer time and network latency. Their algorithm, called *uniform multiround* (UMR), allocates chunks using multiple rounds so as to overlap communication (i.e., transfer time and network latency) and computation and, consequently, decrease the total time for handling the task, i.e., *makespan*.

In Casanova et al.'s work, the main problem lies in the *master/worker* architecture. Because a grid environment usually involves a large number of nodes and tasks, such a centralized architecture is not feasible [6]. In the paper, we will provide an agent-based, decentralized task allocation mechanism. Specifically, the paper will focus on load

- *J. Liu and X. Jin are with the Department of Computer Science, Hong Kong Baptist University, Kowloon Tong, Kowloon, Hong Kong. E-mail: {jiming,jxl}@comp.hkbu.edu.hk.*
- *Y. Wang is with the School of Mathematics and Computational Science, Zhongshan University, Guangzhou, China. E-mail: mcswys@zsu.edu.cn.*

Manuscript received 25 Feb. 2003; revised 14 June 2004; accepted 30 Sept. 2004; published online 20 May 2005.
For information on obtaining reprints of this article, please send e-mail to: tpds@computer.org, and reference IEEECS Log Number TPDS-0018-0203.

balancing among nodes while allocating tasks rather than makespan. Our mechanism is inspired from real and artificial ants' behavior.

1.2 Ant-Based Task Distribution

In natural environments, a group of ants can collect dispersed objects into piles without any manager. Montresor and Meling have developed an inverse artificial ants system, where artificial ants disperse a group of tasks evenly on idle nodes [18], [19]. The detailed behavioral rules of ants are as follows:

1. SearchMax: an ant wanders across the network, looking for overloaded nodes.
2. SearchMin: an ant wanders across the network, looking for underloaded nodes.
3. Transfer: an ant transfers a task from the most overloaded node to the most underloaded one.

In one of their experiments, there are 100 idle nodes on a grid. Initially there are 10,000 tasks on a node. Twenty ants are generated to disperse the tasks. The ants obey the above three rules. After 50 iterations, the tasks are evenly dispersed on the idle nodes, that is, there are 100 tasks on each idle node.

In the experiments of [18], [19], when a user provides tasks on a node, ants are generated to fulfill the tasks. Ants move from nodes to nodes to distribute the tasks. To obtain more information about nodes, ants indirectly communicate with each other. There is a communication layer in each node. Neighbors of a node are defined as a set of nodes known to that node. In the communication layer, there is a collection of neighbors of that node. A visiting ant may add a new neighbor to the collection if the ant has discovered a new node. A visiting ant may remove an old neighbor if the neighbor is discovered to be unreachable. Therefore, the collection of neighbors for a node is highly dynamic during load balancing. Visiting ants not only make the collection of neighbors dynamic, but also use it to know the network topology (neighbors). This kind of indirect communication is called stigmergy [20], which is used by real ants. Based on the detected network topology, ants make decisions such as selecting the best route for accomplishing their tasks. As shown in [18], [19], an ant does not always select the best neighbor as its route, sometimes it selects another route to avoid obstructing as too many ants might select the same route.

Therefore, an ant's strategies in moving between nodes come from communication layers left by other ants. On the other hand, an ant's strategies also come from its own experience directly since it carries its experience when moving [18], [19]. The statistic information from communication layers and its own experience determines an ant's strategies. The strategies change dynamically as the environment is modified by mobile agents. They may be regarded as unchanged during the time period when the change of environment is small. Since ants' strategies in moving between nodes result in the final distributions of load balancing, it is important to establish a relationship between the strategies and the final distributions. In Section 2, we will provide a macroscopic characterization in order to find such a relationship. Furthermore, ants' strategies, which come from statistic information, can be improved to raise the utility gain and the efficiency of load balancing. We will discuss this issue in Section 5.

1.3 Macroscopic Analysis

Microscopic simulations, as shown in [18], [19], can provide interesting experimental results. However, the results are empirical and cannot directly present the relationship between different factors. On the other hand, macroscopic models can directly describe the dynamic behavior of the system and show how the changes of local factors affect the global dynamics. There is a series of such macroscopic models, such as Web site competition [21], coalition formation [22], distributed robot collaboration [23], cyclic feedback [24], infection [25], [26], neural networks [25], biological system [27], turning patterns [28], and chemical reaction-diffusion systems [29].

To give a macroscopic characterization of agent-based load balancing on grids, we should view load balancing from a macroscopic perspective. In the load balancing process of [18], [19], tasks are carried by ants from nodes to nodes. Since what we concern is task distribution on grids, we can assume that we only see tasks' movement between nodes without seeing ants. Therefore, we can regard each task as a mobile ant-like agent. Here, "mobile" means that the agent is active and independent in their decision making. Then, from a macroscopic point of view, the agent-based load balancing becomes a process of agent dispersion. An agent's behaviors in load balancing include: 1) leaving a node where it has queued, 2) wandering on the network, and 3) joining a team in a visited node.

Quantitatively, the effect of agents' behavior on load balancing is reflected by the number and size of teams on grids. Therefore we can use the quantities to describe the load balancing mechanism.

1.4 Problem Statements

As mentioned in Section 1.2, experiments in [18] show that load balancing through artificial ants converges to a steady state which corresponds to an even distribution. However, results are not the same in other papers. In [30], a market mechanism is used to describe an agent-based load balancing. Since agents have incomplete information about nodes, load balancing converges under some conditions, but oscillates and even becomes chaotic under other conditions [30]. Therefore, the main problems we concern in our work are:

1. Does load balancing through artificial ants always converge to a steady state? Furthermore, if load balancing converges to a steady state, does the steady state always correspond to an even distribution although agents only have incomplete information about the grid?
2. How to characterize the quality of load balancing? Is load balancing always worth doing? In other words, when load balancing is not worth doing and when high efficiency can be achieved?
3. Can the agents' strategies employed in load balancing be optimized? That is, the optimization of agents' strategies in load balancing should be addressed in order to improve the efficiency.

In this paper, we will first propose an agent-based load balancing mechanism. Based on our macroscopic characterization and experimental simulations on the mechanism, we study the above problems in depth.

1.5 Organization of the Paper

The rest of the paper is organized as follows: In Section 2, we describe the homogeneous minigrid environment we will concern and then propose an agent-based load balancing mechanism on grids. In Section 3, the load balancing mechanism is modeled using differential equations and all terms in the equations are explained. The convergence and the perfectness of load balancing are discussed in Section 4. In Section 5, the utility gains and the efficiency of load balancing are discussed. In Section 6, we theoretically show how to optimize agents' strategies. In order to validate our proposed load balancing mechanism, we provide our experimental results on a real computing platform, called *Simulation System for Grid Task Distribution* (SSGTD) in Section 7. Finally, we conclude our paper and discuss our future work in Section 8.

2 Agent-Based Load Balancing

As we have mentioned, a grid usually involves a large number of nodes and tasks. Particularly, nodes are geographically distributed. Tasks are submitted to different nodes in a decentralized fashion. Generally speaking, grid nodes are not dedicated to a grid environment. Therefore, a grid lacks of a fixed topology. In addition, because of the distributed nature of a grid, it is hard to globally collect accurate status information of nodes [6]. Given such a situation, a centralized scheduler is not feasible to handling such a complex scheduling problem [6]. Otherwise, the scheduler will be a bottleneck of the grid. Therefore, we need to provide a *scalable* and *decentralized* scheduling mechanism.

2.1 Minigrid Environments Considered

Before we present our agent-based load balancing mechanism, we first describe the specific type of grid environments with divisible tasks, which we will consider in this paper.

1. In our present work, we will focus on load balancing of divisible tasks on grids. Specifically, tasks are divided into independent chunks with the same size before they are submitted to grids.[1] The divisible load/task theory (DLT) has been studied in depth [31]. In DLT, a load can be arbitrarily partitioned into chunks for a group of processors. There is no precedence relation among the obtained chunks.

   A good example of a divisible load comes from the STAR project, a large international collaboration involving about 400 high energy and nuclear physicists from many countries [32]. The divisible load in the STAR is to analyze over 250 tera-bytes of raw data. In [32], Robertazzi and Yu have discussed how to use the grid computing technology to handle the above mentioned heavy load. In addition, as we have seen, Casanova et al. have discussed the task allocation problem in a grid environment where tasks are also assumed to be divisible and independent [7], [8], [16], [17].
2. In this paper, we study the load balancing problem of divisible tasks in homogeneous minigrid environments. Generally speaking, in a grid environment, nodes are heterogeneous in terms of processing speed, memory size, storage space, etc.. However, in a local minigrid environment, nodes are usually homogeneous. The Distributed ASCI Supercomputer (DAS) is a geographically distributed, cluster-based minigrid designed by the Advanced School for Computing and Imaging (ASCI) [33], [34]. It aims at providing five Dutch universities with powerful computational resources for research programs on parallel and distributed computing, such as image processing, weather forecasting, and quantum chemistry. DAS is a homogeneous minigrid environment containing 200 nodes in total. The Parallel Architecture Research Laboratory (PARL) at Clemson University has also set up a computational minigrid consisting of totally 792 homogeneous Pentium III processors, which belong to five Beowulf Clusters interconnected through dedicated Ether links [35], [36].
3. This paper aims at examining whether or not tasks can be distributed to nodes and finally achieve a balanced steady state. Hence, in our work, we assume that the process of load balancing is relative short, during which there is neither new task submitted nor old task finished.

2.2 Agents

Multiagent systems have been widely used in distributed problem solving. In our distributed scheduling mechanism, we use agents to carry tasks. Immediately after a task is submitted to a node, an agent will be automatically dispatched to the task. The agent will carry the task to search for appropriate agent teams[2] to queue. In principle, an agent accompanies a task until it is finished on a certain node.

The goal of an agent is to search suitable nodes for its task, where the agent will obtain a higher utility. Utility may be defined based on different metrics, such as *waiting time* and *service time*. Because in our work, nodes are homogeneous and tasks have the same size, we consider the service time for all tasks at any node as being the same. Hence, in the paper, we only consider *waiting time*. Obviously, here waiting time is only related to the length of the agent team where the agent is queuing. Given such a utility definition, agents prefer to queue at a small agent team so as to achieve a higher utility.

Utility may also be related to other metrics, such as *task transfer time*[3] and *network latency*. In this paper, we assume that as compared to waiting time and service time, task transfer time and network latency may not be significant.

2.3 Load Balancing Mechanism

The load balancing mechanism we concern is as follows: Initially, a group of tasks is submitted to a minigrid. Then a group of agents, whose number is equal to that of the tasks,

1. In the latter part of the paper, "task" refers to independent chunks with the same size in terms of processing time.

2. Here, we refer to the agents queuing at a certain node as an "agent team."

3. The time for transferring a task from one node to another one.

are dispatched to these tasks. From now on, agents asynchronously wander on the network of nodes to search for appropriate nodes for their tasks. Then, load balancing becomes the dispersion of agents.

Consider a system of multiple mobile agents. The agents are dispatched by one or more nodes on a minigrid. Each agent has to obtain a service at a node on the minigrid and each agent's service lasts the same time. Agents have incomplete information about nodes on minigrids. They have to exploit proper nodes and queue for services. To search for small teams, an agent moves among nodes randomly and makes decisions by itself from its own experience and the communication layers it has visited. Because of their goal for obtaining higher utilities, agents will not join a very large team. That is, there is a maximum size for each team. Let m denote the maximum team size, then agents will not join a team whose size is larger than m. Agents probabilistically decide to join the team at a node, or form a new team if there is no other agent. After joining a team, an agent can also probabilistically decide to leave the team and move to other nodes. Therefore, agents' behaviors can be classified into two types: *leaving* and *queuing*. The strategies for the two types will determine the global dynamic behavior of the system.

It should be pointed out that in the paper, without loss of generality, we assume agents properly behave according to their strategies.

3 Theoretical Formulation

In this section, we will provide a macroscopic characterization of the agent-based load balancing mechanism. Our characterization is inspired by the differential equation based modeling work in [22], [23], [37], [38], [39].

As described in Section 2, the agent-based load balancing we concern is regarded as agent dispersion. That is, a group of agents wander on the network and search for proper teams to queue. From the macroscopic point of view, the teams of agents on minigrids vary dynamically as agents leave and queue. The number and size of agent teams reflects the performance of the agent-based load balancing mechanism. We use two quantities to macroscopically characterize load balancing: 1) the size, s, of an agent team and 2) the number, n_s, of teams whose size is s. It follows from Section 2 that there is a maximum team size. Then, during the period of load balancing, we have $1 \leq s \leq m$.

Initially, agents are created for tasks by nodes. Although a mobile agent will not join a team whose size is larger than m, the maximum team size may be larger than m at the beginning. This case can be explained as follows:

Suppose that initially the maximum team size is larger than m. It follows from the mechanism in Section 2 that the agents, who are in the pth ($p \geq m+1$) position of various teams, will leave the teams. The leaving state is concurrent and asynchronous. Therefore, the leaving process would be completed very rapidly since no decision should be made by the departing agents. The departing agents wander synchronously on networks. Each agent decides by itself whether or not to join the teams it encounters. After all the pth ($p \geq m+1$) agents leave their positions and queue in other nodes, the maximum team size in the system will not be larger than m.

Therefore, we will not concern the case where the maximum team size is larger than m. We will focus on the case where the maximum team size is not larger than m in this paper.

We use variables n_s to represent the load balancing process we concern. At first, we give an example to describe the idea behind our characterization. Consider the change of n_2, the number of teams of size two. On one hand, consider the increase of n_2. A team of size three becomes a team of size two after an agent's leaving, then the increase of n_2 is proportional to n_3, the number of teams of size three. This can be denoted by $+l_3 n_3$. Two teams of size one become a team of size two after an agent's queuing, then the increase of n_2 is also proportional to $n_1 n_1$. This can be denoted by $+j_1 n_1^2$. On the other hand, consider the decrease of n_2. A team of size two becomes two teams of size one after an agent's leaving, then the decrease of n_2 is proportional to n_2. This can be denoted by $-l_2 n_2$. A team of size two becomes a team of size three after an agent's queuing, then the decrease of n_2 is also proportional to $n_1 n_2$. This can be denoted by $-j_2 n_1 n_2$. Here, j_1, j_2, l_2, and l_3 are proportion values. Therefore, the change rate $\frac{dn_2}{dt}$ of n_2 can be expressed as:

$$\frac{dn_2}{dt} = l_3 n_3 + j_1 n_1^2 - l_2 n_2 - j_2 n_1 n_2. \quad (1)$$

Based on the above idea, we give the following general macroscopic characterization to quantitatively describe the load balancing mechanism we concern:

$$\begin{aligned} n_1' &= 2l_2 n_2 - 2j_1 n_1^2 + \sum_{k=3}^{m} l_k n_k - n_1 \sum_{k=2}^{m-1} j_k n_k, \\ n_s' &= -l_s n_s + j_{s-1} n_1 n_{s-1} - j_s n_1 n_s + l_{s+1} n_{s+1}, \ 2 \leq s \leq m-1, \\ n_m' &= -l_m n_m + j_{m-1} n_1 n_{m-1}, \end{aligned} \quad (2)$$

where n_1', n_s', and n_m' describe the quantitative change rate of agent teams of size one, s $(2 \leq s \leq m-1)$, and m, respectively; j_s and l_s are positive coefficients; $n_s \geq 0, 1 \leq s \leq m$,[4] and $\sum_{s=1}^{m} s n_s(0) = S$. Here, S is the total number of agents initially.

It follows from the description in Section 2.1 that there is no net change of agents during the period of load balancing. An interesting verification can be done as follows: According to (2), we have:

$$\sum_{s=1}^{m} s n_s' = 0, \text{ i.e., } \sum_{s=1}^{m} s n_s(t) = \sum_{s=1}^{m} s n_s(0) = S \text{ as } t > 0. \quad (3)$$

To better understand (2), we give some more detailed descriptions as follows:

1. Coefficient j_s indicates the rate of agents that encounter teams of size s and decide to join those teams. j_s is regarded as the strategy of agents' queuing.

4. In [37], we theoretically prove that during the process of load balancing, $n_s(t)$ is guaranteed to be nonnegative.

2. Coefficient l_s indicates the rate of agents that are queuing at the last position of agent teams of size s and decide to leave now. It is regarded as the strategy of agents' leaving. As mentioned in Section 1.2, strategies j_s and l_s come from agents' own experience and visited communication layers. They may keep unchanged during the time period when the change of environment is small. Hence, the changes of j_s and l_s is as follows: During different time periods, they have different constant values. We focus on the case where j_s and l_s are constants.
3. The second subequation in (2) is a general one, which characterizes the quantitative change rate of teams of size s $(2 \leq s \leq m-1)$. The rate increases as an agent joins a team of size $s-1$, or an agent leaves a team of size $s+1$. The rate decreases as an agent leaves a team of size s, or an agent joins a team of size s.
   - "$-l_s n_s$" denotes that there are $l_s n_s$ teams of size s, each of which produces one team of size one and one team of size $s-1$ after its last agent's leaving.
   - "$+j_{s-1} n_1 n_{s-1}$" denotes that $j_{s-1} n_1 n_{s-1}$ teams of size one and $j_{s-1} n_1 n_{s-1}$ teams of size $s-1$ produce $j_{s-1} n_1 n_{s-1}$ teams of size s after agents' queuing.
   - "$-j_s n_1 n_s$" denotes that $j_s n_1 n_s$ teams of size one and $j_s n_1 n_s$ teams of size s produce $j_s n_1 n_s$ teams of size $s+1$ after agents' queuing.
   - "$+l_{s+1} n_{s+1}$" denotes that $l_{s+1} n_{s+1}$ teams of size $s+1$ produce $l_{s+1} n_{s+1}$ teams of size one and $l_{s+1} n_{s+1}$ teams of size s after their last agents' leaving.
4. The first and third subequations in (2) are similar to the second one. For the sake of space limitation, we will not discuss them in detail.

4 Global Convergence of Load Balancing

In the experiments of [18] as shown in Section 1.2, 10,000 tasks are evenly dispersed on 100 idle nodes by ants after 50 iterations. That is, the state of load balancing converges to a steady state which corresponds to perfect balancing. Here, perfect balancing is defined as even distribution. It follows from this experimental result that two interesting questions are raised:

1. Does load balancing always converge to a steady state? Mathematically, the question can be expressed as: whether or not the steady state of (2) is globally stable.
2. Does the final distribution always correspond to perfect balancing? Mathematically, the question can be expressed as: whether or not the steady state of (2) always corresponds to the mathematical form of perfect balancing.

In this section, we try to answer these questions through mathematical proofs, numerical simulations, and analysis of strategies. We show that load balancing with different initial distributions will converge to the same distribution finally, which is in agreement with the experiments in [18]. However, we prove that the final distribution does not always correspond to perfect balancing. An interesting result is given that if agents have complete information about nodes on minigrids, then the steady state corresponds to perfect balancing; otherwise, the steady state does not correspond to perfect balancing. That is, perfect balancing emerges if and only if agents have complete information about nodes on minigrids. This is in agreement with the common knowledge, but is not in agreement with the experiments in [18].

4.1 Uniqueness and Stability

In this section, we show that load balancing with different initial distributions converges to the same final distribution (steady state). At first, we prove that the steady state of (2) is unique. Then, the steady state is shown to be globally stable by numerical simulations. Finally, we theoretically prove that the steady state is globally stable if the maximum team size is two or three.

Let $n^* = (n_1^*, n_2^*, \ldots, n_m^*)$ be an equilibrium (steady state) of (2). By definition of equilibrium, the expressions in the right hand side of (2) are zero at n^*.

By the third subequation of (2), we have:

$$-l_m n_m^* + j_{m-1} n_1^* n_{m-1}^* = 0, \text{ i.e., } n_m^* = \frac{j_{m-1}}{l_m} n_1^* n_{m-1}^*. \tag{4}$$

It follows the subequation for teams of size $(m-1)$ in (2), we have:

$$-l_{m-1} n_{m-1}^* + j_{m-2} n_1^* n_{m-2}^* + l_m n_m^* - j_{m-1} n_1^* n_{m-1}^* = 0, \tag{5}$$

i.e.,

$$n_{m-1}^* = \frac{j_{m-2}}{l_{m-1}} n_1^* n_{m-2}^*, \tag{6}$$

by replacing n_m^* with $\frac{j_{m-1}}{l_m} n_1^* n_{m-1}^*$. Similarly, we can derive:

$$n_s^* = \frac{j_{s-1}}{l_s} n_1^* n_{s-1}^*, \ 2 \leq s \leq m. \tag{7}$$

Inductively, we have:

$$n_s^* = \frac{j_1 j_2 \cdots j_{s-1}}{l_2 l_3 \ldots l_s} n_1^{*s}, \ 2 \leq s \leq m. \tag{8}$$

It can be rewritten as follows: $n_s^* = g_s n_1^{*s}, \ 1 \leq s \leq m$, where $g_1 = 1$, and $g_s = \frac{j_1 j_2 \cdots j_{s-1}}{l_2 l_3 \ldots l_s}$, $2 \leq s \leq m$.

It follows from $\sum_{s=1}^{m} s n_s^* = S$ that:

$$F(n_1^*) = \sum_{s=1}^{m} s g_s n_1^{*s} - S = 0. \tag{9}$$

Since $F(0) < 0$, $F(S) > 0$, and $F'(n_1^*) > 0$ as $n_1^* > 0$, according to the continuity of function F, $F(n_1^*) = 0$ has a unique solution in $(0, S)$.

Theorem 1. *Equation (2) has a unique equilibrium (steady state).*

According to Theorem 1, if load balancing with different initial distributions converges to steady states, the steady states are the same one and can be expressed as above.

By numerical simulations in Case Studies 1 and 2, we show that load balancing with different initial distributions converges to the unique steady state. That is, the unique steady state of (2) is shown to be globally stable.

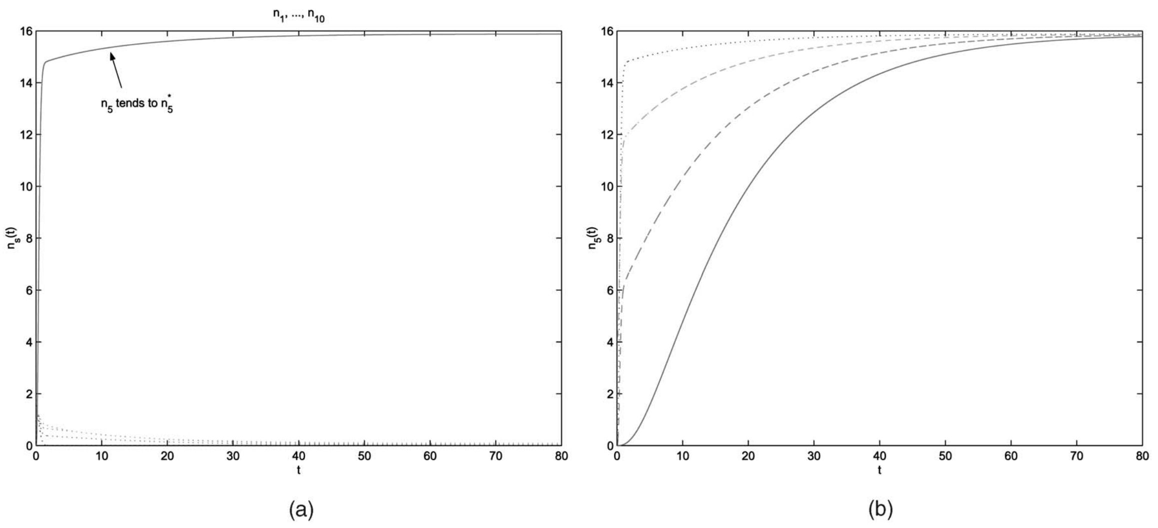


Fig. 1. (a) Case Study 1 with initial load distribution $n(0) = (0, 0, 0, 0, 0, 0, 0, 0, 0, 8)$. Load balancing converges to steady state $n^* = (0, 0, 0, 0, 16, 0, 0, 0, 0, 0)$. (b) Case Study 2 with different initial load distributions. The component $n_5(t)$ of $n(t)$ with different initial values converges to the same value $n_5^* = 16$, which numerically shows the global stability of steady state.

Case Study 1 (Steady State). *Let $m = 10, S = 80, j_s = l_{m-s} = 0.1,\ 0 \le s \le 4,\ j_j = l_{m-j} = 0.001,\ 6 \le j \le 9,\ j_5 = l_5 = 0.00001$. Strategies j_s and l_s are chosen in order to achieve perfect balancing. The initial load distribution is $n(0) = (0, 0, 0, 0, 0, 0, 0, 0, 0, 8)$, i.e., there are only eight teams of size 10 initially.*

In the result of Case Study 1, Fig. 1a, we show that the number of teams of each size in load balancing tends to a constant gradually. That is, each component $n_s(t)$ of a solution $n(t)$ converges to a constant as $t \to +\infty$. In other words, load balancing converges to a steady state.

Case Study 2 (Unique Steady State). *Let $m = 10,\ S = 80,\ j_s = l_{m-s} = 0.1,\ 0 \le s \le 4,\ j_k = l_{m-k} = 0.001,\ 6 \le k \le 9,\ j_5 = l_5 = 0.00001$. Without loss of generality, we consider the fifth component $n_5(t)$. Several initial values $n(0)$ are selected randomly with $\sum_{s=1}^{10} s n_s(0) = 80$.*

It follows from the result shown in Fig. 1b that the fifth components of solutions $n(t)$ with different initial values $n(0)$ converge to the same values n_5^*. Other components of solutions $n(t)$ with different initial values $n(0)$ also converge to the same values but those figures are not shown in this paper. In other words, solutions $n(t)$ to (2) with different initial conditions tend to the same steady state as $t \to +\infty$.

Therefore, the unique steady state of (2) is shown to be globally stable by numerical simulations.

In the following theorem, we theoretically prove that the unique steady state is globally stable if $m \le 3$, that is, load balancing converges to the unique steady state in case where the maximum team size is two or three.

Theorem 2. *In case $m = 2, 3,$ the equilibrium of (2) is globally stable.*

Proof. Suppose $m = 2$, we have

$$n_1' = 2l_2 n_2 - 2j_1 n_1^2,\ \ n_2' = -l_2 n_2 + j_1 n_1^2, \tag{10}$$

where $n_1 + 2n_2 = S$. Then, $n_2 = \frac{1}{2}(S - n_1)$, the first equation of (10) is:

$$n_1' = G(n_1), \tag{11}$$

where $G(n_1) = l_2 S - l_2 n_1 - 2j_1 n_1^2,\ 0 \le n_1 \le S$.

Since $G(n_1)' < 0$ as $n_1 > 0$, then there is a unique equilibrium of (11) as $n_1 > 0$. It follows from $G(0) > 0, G(S) < 0$ that the equilibrium is globally stable on $[0, S]$. That is, the equilibrium of (10) is globally stable.

A similar proof can be given for $m = 3$. □

4.2 Perfectness

Perfect load balancing in experiments of [18] is very interesting. However, it is supposed that there are certain idle nodes on grids in the experiments. This is not in agreement with the nature of load balancing on grids. In fact, the network of idle nodes lacks fixed structures. Then, the number of idle nodes is random. Here, we discuss the perfectness of load balancing by the expression of equilibrium of (2). We concern the necessary and sufficient conditions under which there exists perfect balancing.

Let n_1^* be the unique solution of $F(n_1^*) = 0$ in $(0, S)$. Then, the equilibrium of (2) can be expressed as follows:

$$n^* = (g_1 n_1^*, g_2 n_1^{*2}, g_3 n_1^{*3}, \ldots, g_m n_1^{*m}), \tag{12}$$

where

$$g_1 = 1, \text{ and } g_s = \frac{j_1 j_2 \cdots j_{s-1}}{l_2 l_3 \ldots l_s}, 2 \le s \le m. \tag{13}$$

Perfect balancing corresponds to the form of equilibrium as follows:

$$n^* = (0, \ldots, 0, n_k^*, 0, \ldots, 0), \tag{14}$$

that is, there is k, $1 \le k \le m$, such that $n_s^* = 0$ as $s \ne k$ and $n_k^* > 0$. Therefore, the equilibrium of (2) does not correspond to perfect balancing generally as $j_s > 0$ and $l_s > 0$. When does the equilibrium of (2) correspond to perfect balancing?

We return to the definition of equilibrium of (2). Then, we have: The equilibrium of (2) has the form of perfect balancing if and only if there is k, $1 \leq k \leq m$ such that:

1. $j_s = 0$ as $k \leq s \leq m-1$,
2. $j_s > 0$ as $1 \leq s \leq k-1$,
3. $l_s = 0$ as $2 \leq s \leq k$, and
4. $l_s > 0$ as $k+1 \leq s \leq m$.

The detailed proof can be found in [37].

It follows from the definition of equilibrium that the equilibrium of (2) varies continuously as parameters j_s and l_s vary. So, the form of perfect balancing can be closely reached if and only if the following conditions are satisfied:

1. l_s is small enough as $2 \leq s \leq k$.
2. j_s is small enough as $k \leq s \leq m-1$.
3. j_s remains large enough (relative to the small enough) as $1 \leq s \leq k-1$.
4. l_s remains large enough (relative to the small enough) as $k+1 \leq s \leq m$.

These conditions can be verified in the numerical simulation in Fig. 1a. From the strategies point of view, these conditions mean that agents have complete information about nodes on minigrids:

1. It follows from j_s $(1 \leq s \leq k-1)$ is large enough and l_s $(2 \leq s \leq k)$ is small enough that: for the teams whose sizes are less than k, agents would like to join the teams and would not leave the teams after joining.
2. It follows from l_s $(k+1 \leq s \leq m)$ is large enough and j_s $(k \leq s \leq m-1)$ is small enough that: for the teams whose sizes are larger than k, agents would like to leave the teams and would not join these kinds of teams after leaving.

Therefore, if the equilibrium has the form of perfect balancing, that is, perfect balancing exists in load balancing, then agents have complete information about nodes on minigrids. On the other hand, it is known that if agents have complete information about nodes on minigrids, the load will be evenly balanced on idle nodes, that is, perfect balancing exists in load balancing. Then, we have:

Theorem 3. *Perfect balancing exists in load balancing if and only if agents have complete information about nodes on minigrids.*

Generally speaking, agents do not have perfect knowledge about minigrid nodes. Therefore, perfect balancing does not always emerge in load balancing. The perfect load balancing shown in experiments of [18] exists under the strict conditions we give.

5 Efficiency of Load Balancing

In Section 4, we have shown that load balancing described by (2) converges to a unique steady state. The convergent speed and the final distribution at the steady state present the quality of load balancing. In this section, we will measure the quality of load balancing by analyzing the convergent speed and the distribution at the steady state. The problems we concern are:

1. Whether or not load balancing is worth doing?
2. How about the utility gains and the efficiency during different balancing periods?

It is known that the goal of load balancing is to save time in fulfilling tasks. Since the service time for each agent (task) is assumed the same, we define utility gains of load balancing by waiting time. Let $T > 0$ be the service time for each agent (task). Let the initial distribution of agents (tasks) be $n(0) = (n_1(0), n_2(0), \ldots, n_m(0))$. That is, initially the number of teams of size s on minigrids is $n_s(0)$, $1 \leq s \leq m$. Let S be the total number of agents (tasks) in the system, then $\sum_{s=1}^{m} s n_s(0) = S$.

Suppose there is no load balancing. Consider a team of size m. Then, the first agent in the team need not to wait. The second agent has to wait for time T when the first agent is served. The third agent has to wait for time $2T$ when the first agent and the second agent are served. Inductively, The mth agent has to wait for time $(m-1)T$ when the first $(m-1)$ agents in the team are served. Therefore, the total waiting time of the team is

$$T + 2T + 3T + \ldots + (m-1)T = \sum_{s=1}^{m-1} sT = \frac{1}{2}m(m-1)T \tag{15}$$

if there is no load balancing. Since initially the number of teams of size m is $n_m(0)$, then the total waiting time of teams of size m is $\frac{1}{2}m(m-1)Tn_m(0)$. Inductively, for $1 \leq s \leq m-1$, the total waiting time of teams of size s is $\frac{1}{2}s(s-1)Tn_s(0)$. Therefore, the total waiting time of the system is $\frac{1}{2}\sum_{s=1}^{m} s(s-1)Tn_s(0)$ if there is no load balancing.

Suppose there is load balancing under strategies j_s and l_s, then the distribution tends to steady state $n^* = (n_1^*, n_2^*, \ldots, n_m^*)$. The waiting time for n_1^* teams of size one is zero. The waiting time for n_2^* teams of size two is $n_2^* T$. Inductively, the waiting time for n_m^* teams of size m is $\frac{1}{2}m(m-1)Tn_m^*$. Therefore, the total waiting time of the system is $\frac{1}{2}\sum_{s=1}^{m} s(s-1)Tn_s^*$ after load balancing.

Definition 1. *The total utility gain E_0 of load balancing is defined as the difference between two waiting times:*

$$\begin{aligned} E_0 &= \frac{1}{2}\sum_{s=1}^{m} s(s-1)Tn_s(0) - \frac{1}{2}\sum_{s=1}^{m} s(s-1)Tn_s^* \\ &= \frac{1}{2}T\sum_{s=1}^{m} s(s-1)(n_s(0) - n_s^*), \end{aligned} \tag{16}$$

where $n(0) = (n_1(0), n_2(0), \ldots, n_m(0))$ is the initial distribution of agents, $n^* = (n_1^*, n_2^*, \ldots, n_m^*)$ is the final distribution of agents at the steady state.

When load balancing performs, the distribution $n(0)$ is changed into $n(t)$ at time t. Then, we have:

Definition 2. *The utility gain $E(n(t))$ of load balancing at time t is defined as:*

$$\begin{aligned} E(n(t)) &= \frac{1}{2}\sum_{s=1}^{m} s(s-1)Tn_s(0) - \frac{1}{2}\sum_{s=1}^{m} s(s-1)Tn_s(t) \\ &= \frac{1}{2}T\sum_{s=1}^{m} s(s-1)(n_s(0) - n_s(t)), \end{aligned} \tag{17}$$

where $n(0) = (n_1(0), n_2(0), \dots, n_m(0))$ is the initial distribution of agents, $n(t) = (n_1(t), n_2(t), \dots, n_m(t))$ is the distribution of agents at time t.

Definition 3. *The efficiency $\beta(t)$ of load balancing at time t is defined as:*

$$\beta(t) = \frac{d\alpha(t)}{dt}, \tag{18}$$

where $\alpha(t)$ is the fraction of total utility gain that load balancing finishes during time period $[0, t]$:

$$\alpha(t) = \frac{E(n(t))}{E_0}. \tag{19}$$

The definition of efficiency gives the relation between the fraction of utility gains and the consumed time.

In order to discuss the efficiency clearly, we consider the case where both the attachment rate and the detachment rate are uniform, that is, $j_s = j$, $l_s = l$, $1 \le s \le m$. Then, (2) can be rewritten as follows:

$$\begin{aligned} n_1' &= 2ln_2 - 2jn_1^2 + l\sum_{k=3}^{m} n_k - jn_1\sum_{k=2}^{m-1} n_k, \\ n_s' &= -ln_s + jn_1n_{s-1} - jn_1n_s + ln_{s+1},\ 2 \le s \le m-1, \\ n_m' &= -ln_m + jn_1n_{m-1}. \end{aligned} \tag{20}$$

Under certain strategy j and l in (20), the distribution $n(t)$ varies with time t because of load balancing. For different strategy j and l in (20), the corresponding steady state n^* is different. For different strategies j and l and at different time t, both the utility gains and the efficiency of load balancing are different. We show the cases through numerical simulations of (20).

Case Study 3 (Utility Gain and Efficiency). *Let $m=6, T = 1$, that is, the maximum team size is six and the service time is one unit of time. Consider the initial distribution (condition) $n(0) = (0, 0, 0, 0, 0, 100)$, which means there are 100 teams of size 6 initially. Let $l/j = 10^{-6}$, 10^{-4}, 10^{-2}, 10^{0}, 10^1, 10^2, $t = 1, 10, 100$, respectively.*

The simulation result of Case Study 3 is shown in Fig. 2. In Fig. 2a, it is shown that as $l/j = 10^2$, the utility gain at time $t = 10$ is equal to that at time $t = 100$, that is, a little time is enough to reach the maximum utility gain. It is also shown that as $l = 10^{-6}$, the utility gains at time $t = 1, 10, 100$ are all zero, that is, much time of load balancing has little effect on increasing the utility gain.

In Fig. 2b, it is shown that as $l/j = 10$, the gain fractions at time $t = 1, 10, 100$ are $\alpha(0.01) = 8$ percent, $\alpha(0.1) = 30$ percent, $\alpha(200) = 50$ percent, that is, high efficiency emerges during the first balancing time. It is also shown that as $l/j = 10^{-6}$, the gain fractions at time $t = 1, 10, 100$ are close to zero, that is, the efficiency is very low there.

It follows from the numerical simulation in Fig. 2 that some discussions can be given about the efficiency in load balancing:

1. As parameter l/j is large, a little time is sufficient for load balancing to reach its maximum utility gain and efficiency, which is shown in both Figs. 2a and 2b as $l/j \ge 10$. This is in agreement with the special case where the number of nodes is larger than that of agents (tasks), agents' experiences tell them to disperse without queuing. In fact, let $j_s = 0$, $l_s > 0$, $1 \le s \le m$. Then, (2) becomes linear differential equations and can be solved. By solutions of the linear (2), a little time is enough for agents' dispersion. Therefore, as parameter l/j is large, high efficiency emerges in the first time period.
2. As parameter l/j is small, much more time is needed for load balancing to reach its steady state, which is shown both in Fig. 2a as $l < 10^{-4}$. As the time of load balancing increases, the utility gains increase very slowly. On the other hand, the total (final) utility gains are small. Therefore, load balancing is not worth doing in these cases. For example, if there is a little free nodes on the network, load balancing is not worth doing since time is consumed with little gains. Here, there is no obvious time period during which high efficiency emerges.
3. As parameter l/j is assigned a medial value, the balance between utility gain and consumed time should be evaluated. Since

$$lim_{t\to+\infty} n_s(t) = n_s^*,\ 1 \le s \le m, \tag{21}$$

then

$$\frac{dE(n(t))}{dt} \to 0 \text{ as } t \to +\infty. \tag{22}$$

The efficiency $\beta(t)$ becomes very small as time t is large enough. Therefore, if the efficiency $\beta(t)$ is less than a predefined value, load balancing should be stopped; on the other hand, if the gain fraction $\alpha(t)$ is greater than a predefined value, load balancing should be stopped.

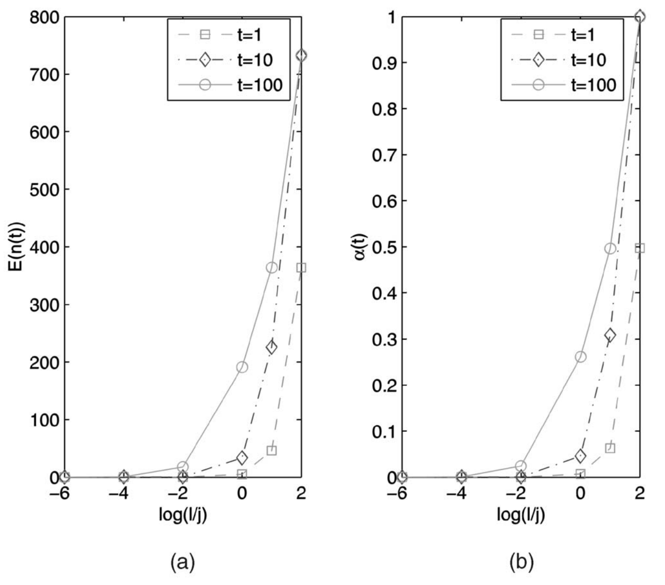


Fig. 2. Case Study 3: For certain strategies $l/j = 10^{-6}, 10^{-4}, 10^{-2}, 10^0, 10^1, 10^2$, utility gains E and gain fraction α vary as time increases, where $t = 1, 10, 100$, respectively.

6 Optimization of Agents' Strategies

While wandering on the network, an ant collects information on nodes it has visited [18]. The information is about the status of other nodes and is left by other ants. According

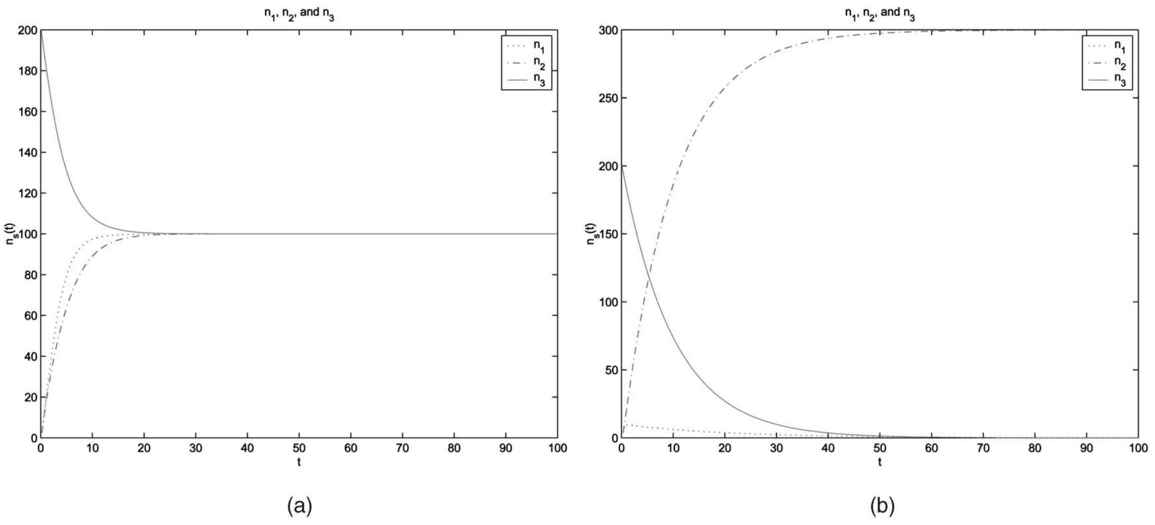


Fig. 3. (a) Case Study 4, where $j_1 = j_2 = 0.001$ and $l_2 = l_3 = 0.1$. The steady state is $n^* = (100, 100, 100)$. The load is not even balanced. (b) Case Study 5, where $j_2 = l_2 = 0$ and $j_1 = l_3 = 0.1$. The steady state is $n^* = (0, 300, 0)$, i.e., the load is absolutely even balanced.

to its own experience and the information from other ants, the ant decides where to go and forms its strategies of leaving and queuing. That is, ants' strategies of leaving and queuing simply come from the statistic information about nodes. Is there any adjustment of strategies to maximize the total utility gains of load balancing?

It is known that perfect balancing, where load is evenly balanced, corresponds to the maximum of total utility gains. In this section, we discuss the optimization of strategies in order to maximize total utility gains. It follows from Theorem 1 that the steady state is uniquely determined by agents' strategies of leaving and queuing: different strategies j_s and l_s result in different distributions at steady states, and the steady states do not always correspond to perfect balancing. In this section, we theoretically give the optimization of strategies from the expression of steady states.

First, we show our idea about optimization through an example. Second, the abstract discussion is given. Here, we consider the case $m = 3$, that is, the maximum team size is three.

Case Study 4 (Nonperfect Load Balancing). *Let* $j_1 = j_2 = 0.001$, $l_2 = l_3 = 0.1$. *Initially, there are 200 teams of size three, i.e.,* $n(0) = (0, 0, 200)$. *In other words, there are 600 agents (tasks) to be balanced,* $S = 600$.

The result in Fig. 3a shows that under the given strategies, load balancing converges to the steady state $n^* = (100, 100, 100)$, where there are 100 teams of size one, 100 teams of size two, and 100 teams of size three, respectively. Then, the load is not perfectly balanced.

Now, we revisit the steady state reduced by the given strategies. It follows from the steady state $n^* = (100, 100, 100)$ that there are 300 idle nodes on minigrids. For the total 600 tasks, the corresponding perfect balancing is $(0, 300, 0)$. To reach the perfect balancing, agents' strategies j_1, j_2, l_2, and l_3 should be adjusted from the old ones. In fact, it follows from the perfect balancing $(0, 300, 0)$ that there is no teams of size three and there is no teams of size one at the steady state. Therefore, an agent will not join a team of size two to form a team of size three, i.e., $j_2 = 0$. And, an agent will not leave a team of size two, i.e., $l_2 = 0$. Then, agents' strategies j_1, j_2, l_2, and l_3 can be optimized as follows:

$$j_2 = l_2 = 0, j_1 > 0, \text{ and } l_3 > 0. \tag{23}$$

Case Study 5 (Perfect Load Balancing). *Let* $m=3, j_2=l_2 = 0$, $j_1 = l_3 = 0.1$, $n(0) = (0, 0, 200)$ *to examine the final steady state.*

The result in Fig. 3b shows that under the optimized strategies, load balancing converges to a steady state $n^* = (0, 300, 0)$, i.e., there are 300 teams of size two and there is no team of size one or three. Perfect balancing emerges in this case.

We can generalize the optimization of strategies in the example. In fact, the optimization of strategies j_s and l_s can be obtained from the expression of steady state n^*. That is, the strategies for perfect balancing can be deduced from current strategies j_s and l_s. It follows from n^* that there are $l = \sum_{s=1}^{m} n_s^*$ idle nodes on minigrids. Let $k = \frac{S}{l}$. In order to express our idea more clearly, we assume that k and l are integers. For the case where k and l are not integers, similar discussions can be given.

Since k and l are integers, there are l idle nodes on minigrids and S agents can be evenly balanced among the l idle nodes with team size k. Therefore perfect load balancing should be:

$$y^* = (0, \dots, 0, y_k^*, 0, \dots, 0), \tag{24}$$

where $y_k^* = l$. The perfect load balancing can be reached if strategies j_s and l_s are properly rearranged:

1. An agent will not join a team of size s as $k \le s \le m - 1$, that is, $j_s = 0, k \le s \le m - 1$.
2. An agent will join a team of size s as $1 \le s \le k - 1$, that is, $j_s > 0, 1 \le s \le k - 1$.
3. An agent will not leave a team of size s as $2 \le s \le k$, that is, $l_s = 0, 2 \le s \le k$.
4. An agent will leave a team of size s as $k + 1 \le s \le m$, that is, $l_s > 0, k + 1 \le s \le m$.

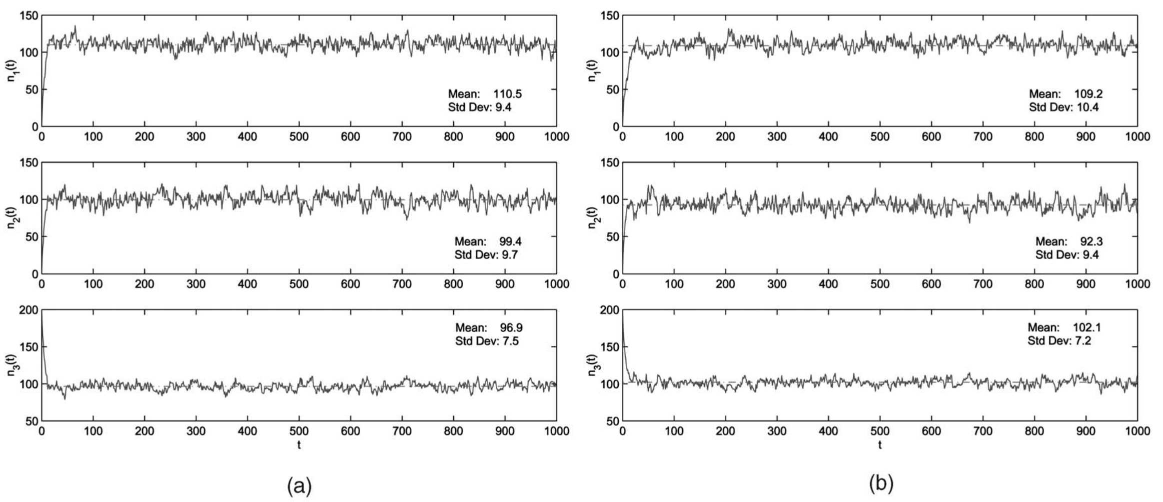


Fig. 4. Let $m=3$, $j_1=j_2=0.001$, $l_2=l_3=0.1$, $n(0)=(0,0,200)$. (a) Fully connected minigrid. The final approximately steady state is $n^*=(111,99,97)$. (b) Scale-free minigrid. The final approximately steady state is $n^*=(109,92,102)$.

Let $N^*=(N_1^*,N_2^*,\ldots,N_m^*)$ be the new steady state under the new strategies. We give the expression of N^* by (2). It follows from $j_s=0, k\le s\le m-1$ that $N_s^*=0, k+1\le s\le m$. It follows from $l_s=0, 2\le s\le k$ that $N_s^*=0, 1\le s\le k-1$. Therefore, the new steady state determined by the new strategies is $N^*=(0,\ldots,0,l,0,\ldots,0)$, which corresponds to perfect load balancing.

Suppose that strategies j_s and l_s cannot be determined accurately, that is, there are only estimated ranges for j_s and l_s. The above discussion is still effective: We can give a range of perfect load balancing y^*, based on which we can give ranges for new strategies j_s and l_s.

7 Experimental Validation

In the previous sections, we have provided a differential equation system for characterizing the agent-based load balancing on minigrids. We have further studied the properties of global convergence and efficiency through theoretical proofs and numerical simulations on the equation system. In order to experimentally validate our system, we have developed a grid computing platform, called *Simulation System for Grid Task Distribution* (SSGTD), using Java language based on the multithread technique, where each agent is simulated with a thread. On this computing platform, we have implemented the proposed agent-based load balancing mechanism. Through experiments, we have observed that the results in general confirm those found from the numerical simulations. Subsequently, we have validated the effectiveness of our equation system.

Due to space limitation, in this section, we will show only the results of the experiments, where the settings are similar to those of Case Studies 4 and 5 in Section 6.

7.1 Topology-Independency

In the following, we will examine the effects of different topologies of minigrids on our agent-based load balancing mechanism.

As we have seen, (2) does not consider the topology of the minigrid. However, in our experiments on the SSGTD platform, in order to simulate a realistic minigrid environment where agents carry tasks to search for appropriate agent teams, we need to address this important issue. A straightforward question is: Can different topologies affect the performance of the multiagent system as well as the final load balancing result? In order to answer this question, we will investigate three different topologies of minigrids:

- Fully connected minigrid, where each node is connected to all other nodes.
- Lattice-like minigrid, where each node is connected to four neighboring nodes, and all nodes form a lattice-like network.
- Scale-free minigrid, where nodes form a scale-free network. It has been proven that the Internet exhibits a scale-free topology [38]. Since a grid is usually based on the Internet, we believe that a grid more or less has a scale-free topology. Therefore, we use a model in [39] to generate scale-free networks.

In the following experiment, we let the maximum team size $m=3$, the initial agent distribution be $n(0)=(0,0,200)$, and $j_1=j_2=0.001$, $l_2=l_3=0.1$. Note that in the previous sections, j_s denotes the *ratio* of wandering agents that join existing agent teams of size s. In this experiment, j_s denotes the *probability* with which a wandering agent joins an existing agent team of size s when it meets this team. Similarly, l_s denotes the *probability* with which an agent at the last position of an existing team of size s leaves the team.

In Fig. 4, we presented our experimental results on a fully connected minigrid and a scale-free minigrid, respectively. Note that due to space limitation, our experimental result on a lattice-like minigrid is omitted, which is similar to those in Fig. 4. We can note from these figures that for all cases, the final states are not strictly steady. All cases achieve *approximately steady states*, where the numbers of agent teams of different size oscillate along a certain value. This is because of the probabilistic feature of agents' strategies. Specifically, at each time step, agents probabilistically decide to join teams that they encounter or leave teams where they are queuing. Therefore, as long as not all probabilities j_s and l_s are zero, at each time step there always exists a few agents joining or leaving existing teams.

TABLE 1
A Comparison of the Final Approximately Steady States in Figs. 4a and 4b

	n_1^*		n_2^*		n_3^*	
	Mean	Std. Dev.	Mean	Std. Dev.	Mean	Std. Dev.
Fully connected minigrid	110.5	9.4	99.4	9.7	96.9	7.5
Lattice-like minigrid	107.1	9.6	103.1	10.4	95.6	7.9
Scale-free minigrid	109.2	10.4	92.3	9.4	102.1	7.2

Although the final states in Fig. 4 are not strictly steady, they are quite close to the steady state shown in Fig. 3a,[5] i.e., $n^* = (100, 100, 100)$. That is to say, our experimental results are by and large in agreement with that obtained from our numerical simulation of (2). This validates that our proposed agent-based load balancing characterization system (i.e., (2)) is effective.

An interesting phenomenon is that as we can see from Table 1 and Fig. 4, for different topologies of the minigrids, the processes of load balancing are similar to each other. In particular, the final approximately steady states are quite close to each other. That means the proposed agent-based load balancing mechanism is insensitive to the topology of the minigrid. It further indicates that no consideration of the minigrid topology in (2) is feasible.

Remark 1. According to the above observations, we can conclude that:

1. The topology of the minigrid does not affect the agent-based load balancing mechanism on minigrids, including the process of load balancing and the final approximately steady state. In other words, the proposed mechanism is topology-independent.
2. Since the proposed load balancing mechanism is insensitive to the topology, the abrupt coming or going of nodes (if there is no agent queuing) will not affect the performance of the load balancing process. In this sense, the proposed load balancing mechanism is fault-tolerant.

7.2 Approximately Perfect Load Balancing

In this section, we will examine whether or not the agent-based load balancing mechanism can lead to the perfect load balancing (as discussed in Section 4.2) on the SSGTD platform.

In the following experiment, we change the probabilities in the previous experiment to $j_2 = l_2 = 0,\ j_1 = l_3 = 0.1$. Because of the topology-independency of the proposed mechanism, in this experiment we only use a scale-free network of gird nodes. The result is shown in Fig. 5.

From Fig. 5, we can note that the final approximately steady state is $n^* = (34, 283, 0)$, i.e., it is not strictly even balanced as the one shown in Fig. 3b, but rather approximately even balanced. This is also because of the probabilistic feature of agents' strategies.

5. The ranges of x-axes in Fig. 3a and Figs. 4a and 4b are different. This is because of their different time scale.

8 Conclusions and Future Work

In this paper, agent-based load balancing on minigrids is regarded as agent dispersion. We have presented a macroscopic characterization of load balancing based on two quantities on minigrids, i.e., the number and size of agent teams. We have shown that load balancing always converges to a steady state, which is in agreement with experiments through artificial ants. As contrary to even distributions in experiments through artificial ants, we have shown an interesting phenomenon, i.e., the steady state does not always correspond to an even distribution. We have theoretically proven that an even distribution emerges if and only if agents have complete knowledge about minigrid nodes. To measure the quality of load balancing, we have defined a utility gain function and have given a series of analysis about the efficiency of load balancing. We have theoretically found that agents' strategies can be optimized to maximize the utility gain of load balancing.

Finally, we have developed a real platform, called *Simulation System for Grid Task Distribution* (SSGTD), to validate our proposed agent-based load balancing mechanism. Through experimentation, we have found that the results from our experiments on the SSGTD platform in general confirm those from our theoretical analysis and numerical simulations in (2). In addition, we have observed another interesting phenomenon, i.e., the agent-based load balancing mechanism cannot be affected by different topologies of minigrids.

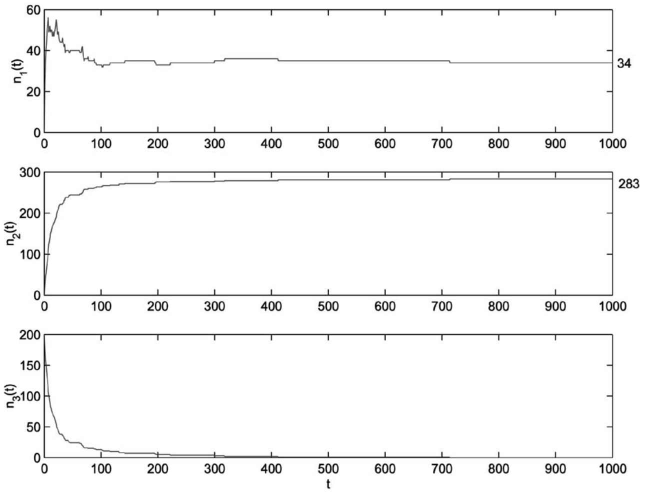


Fig. 5. Let $m = 3$, $j_2 = l_2 = 0$, $j_1 = l_3 = 0.1$, $n(0) = (0, 0, 200)$. The final approximately steady state is $n^* = (34, 283, 0)$. The load is approximately even balanced.

Regarding the future work, the following are three important directions:

1. **Grid environments:** In this paper, we have specifically focused on a type of homogeneous minigrid environments. In order to further exploit the functionality of our agent-based load balancing mechanism, in our future work, we will extend the mechanism to more general environments, where nodes may be heterogeneous in terms of their processing speed, memory size, and storage space, and nodes may dynamically come and go. If nodes are heterogeneous, even for the same task, the service time on different nodes may be different. In order to address the load balancing problem in such a grid environment, we need to extend our proposed mechanism to take into account more factors, such as processing speed of nodes. In this paper, we have not considered task transfer time and network latency. In the future work, we will study them both theoretically and experimentally.
2. **Task interdependency and granularity:** This paper has studied a task distribution environment where tasks are arbitrarily divisible. How to extend our proposed mechanism such that it can work well in the case of more general task environments is an interesting research issue. Some general task environments may have the following characteristics:

   - Tasks may be interdependent rather than independent.
   - Tasks cannot be divisible. Or, tasks are divisible. However, tasks can only be partitioned into chunks of different granularities.
   - Tasks need coallocation of different computational resources, such as, CPU time and memory.

   Providing an agent-based task distribution mechanism for the above environments will be a natural extension of this work.
3. **Agent behavioral variation:** In this paper, we have assumed that agents properly behave according to their strategies. In real world environments, agents perhaps perform without strictly complying their predefined behaviors. For example, because of some unpredictable factors, agents may fail to handle tasks, or agents congest at certain nodes, etc.. In our future work, we need to address the effects of agent behavioral variation on the load balancing mechanism.

Acknowledgments

The authors would like to express their gratitude to the editors and reviewers for their valuable suggestions. They wish to thank Mr. Hoi Fung Lam for the implementation of the SSGTD platform. We want to acknowledge the support of the following research grants: 1) Hong Kong Research Grant Council (RGC) Central Allocation Grant (HKBU 2/03/C) and Earmarked Research Grants (HKBU 2121/03E)(HKBU 2040/02E), 2) Hong Kong Baptist University Faculty Research Grants (FRG), 3) National Grand Fundamental Research 973 Program of China (2003CB317001), and 4) Beijing Municipal Key Laboratory of Multimedia and Intelligent Software Technology (KP0705200379).

References

[1] I. Foster, "Internet Computing And The Emerging Grid," *Nature Web Matters,* http://www.nature.com/nature/webmatters/Grid/grid.html, 2000.

[2] *Grid Computing: Making the Global Infrastructure a Reality,* F. Berman, G. Fox, and T. Hey, eds. John Wiley and Sons, 2003.

[3] *The Grid: Blueprint for a New Computing Infrastructure,* I. Foster and C. Kesselman, eds. Morgan Kaufman, 1999.

[4] I. Foster, J. Geisler, B. Nickless, W. Smith, and S. Tuecke, "Software Infrastructure for the I-WAY High-Performance Distributed Computing Experiment," *Grid Computing: Making the Global Infrastructure a Reality,* F. Berman, G. Fox, and T. Hey, eds., chapter 4, pp. 101-116, John Wiley and Sons, 2003.

[5] I. Foster, "The Grid: A New Infrastructure for 21st Century Science," *Grid Computing: Making the Global Infrastructure a Reality,* F. Berman, G. Fox, and T. Hey, eds., chapter 2, pp. 51-64, John Wiley and Sons, 2003.

[6] A. Galstyan, K. Czajkowski, and K. Lerman, "Resource Allocation in the Grid Using Reinforcement Learning," http://www.isi.edu/lerman/papers/papers.html, 2003.

[7] H. Casanova, A. Legrand, D. Zagorodnov, and F. Berman, "Heuristics for Scheduling Parameter Sweep Applications in Grid Environments," *Proc. Ninth Heterogeneous Computing Workshop (HCW 2000),* pp. 349-363, May 2000.

[8] H. Casanova, J. Hayes, and Y. Yang, "Algorithms and Software to Schedule and Deploy Independent Tasks in Grid Environments," *Proc. Workshop Distributed Computing, Metacomputing, and Resource Globalization,* Dec. 2002.

[9] S. Ho, "Survey of a Grid Meta-Scheduler," http://web.bii.a-star.edu.sg/sebastianh/Scheduler_Survey.pdf, 2002.

[10] F. Berman, R. Wolski, H. Casanova, W. Cirne, H. Dail, M. Faerman, S. Figueira, J. Hayes, G. Obertelli, J. Schopf, G. Shao, S. Smallen, N. Spring, A. Su, and D. Zagorodnov, "Adaptive Computing on the Grid Using AppLeS," *IEEE Trans. Parallel and Distributed Systems,* vol. 14, no. 4, pp. 369-382, Apr. 2003.

[11] F. Berman, R. Wolski, S. Figueira, J. Schopf, and G. Shao, "Application-Level Scheduling on Distributed Heterogeneous Networks," *Proc. 1996 ACM/IEEE Conf. Supercomputing (SC '96),* 1996.

[12] D. Abramson, R. Buyya, and J. Giddy, "A Computational Economy for Grid Computing and Its Implementation in the Nimrod-G Resource Broker," *Future Generation Computer System,* vol. 18, pp. 1061-1074, 2002.

[13] H. Dail, H. Casanova, and F. Berman, "A Decoupled Scheduling Approach for the GrADS Environment," *Proc. 2002 ACM/IEEE Conf. Supercomputing (SC 2002),* Nov. 2002.

[14] S.S. Vadhiyar and J.J. Dongarra, "A Metascheduler for the Grid," *Proc. 11th Symp. High Performance Distributed Computing (HPDC-11),* July 2002.

[15] J. Frey, T. Tannenbaum, I. Foster, M. Livny, and S. Tuecke, "Condor-G: A Computation Management Agent for Multi-Institutional Grids," *Proc. 10th IEEE Symp. High Performance Distributed Computing (HPDC-10),* Aug. 2001.

[16] Y. Yang and H. Casanova, "Umr: A Multi-Round Algorithm for Scheduling Divisible Workloads," *Proc. Int'l Parallel and Distributed Processing Symp. (IPDPS '03),* Apr. 2003.

[17] "Rumr: Robust Scheduling for Divisible Workloads," *Proc. 12th IEEE Symp. High Performance and Distributed Computing (HPDC-12),* June 2003.

[18] A. Montresor, H. Meling, and O. Babaoglu, "Messor: Load-Balancing through a Swarm of Autonomous Agents," Technical Report UBLCS-02-08, Dept. of Computer Science, Univ. of Bologna, Bologna, Italy, May 2002.

[19] O. Babaoglu, H. Meling, and A. Montresor, "Anthill: A Framework for the Development of Agent-Based Peer-to-Peer Systems," *Proc. 22nd Int'l Conf. Distributed Computing Systems (ICDCS 2002),* July 2002.

[20] P. Grasse, "La Reconstruction du Nid et Les Coordinations Interindividuelles Chez Bellicositermes Natalensis et Cubitermes Sp," *Insectes Sociaux,* vol. 6, pp. 41-80, 1959.

[21] L. Lopez and A.F. Sanjuan, "Defining Strategies to Win in the Internet Market," *Physica A,* vol. 301, pp. 512-534, 2001.

[22] K. Lerman and O. Shehory, "Coalition Formation for Large-Scale Electronic Markets," *Proc. Fourth Int'l Conf. Multi-Agent Systems (ICMAS 2000),* pp. 167-174, 2000.

[23] K. Lerman, A. Galstyan, A. Martinoli, and A.J. Ijspeert, "A Macroscopic Analytical Model of Collaboration in Distributed Robotic Systems," *Artificial Life,* vol. 7, no. 4, pp. 375-393, 2001.

[24] J. Hofbauer and K. Sigmund, *Evolutionary Games and Replicator Dynamics.* Cambridge Univ. Press, 1998.

[25] F.J. Hoppensteadt and J.S. Pesk, *Mathematics in Medicine and the Life Sciences.* Springer, 1992.

[26] W.P. Nelson and A.S. Perelson, "Mathematical Analysis of Delay Differential Equation Model of HIV-1 Infection," *Math. Biosciences,* vol. 179, pp. 73-94, 2002.

[27] Y. Wang, "The Necessary and Sufficient Conditions for the Existence of Periodic Orbits in a Lotka-Volterra System," *J. Math. Analysis and Applications,* 2004.

[28] A.M. Turing, "The Chemical Basis of Morphogenesis," *Philosophical Trans. Royal Soc. B,* vol. 327, pp. 37-72, 1952.

[29] Z. Noszticzius, W. Horsthemke, W.D. McCormick, H.L. Swinney, and W.Y. Tam, "Sustained Chemical Waves in an Annular Gel Reactor: A Chemical Pinwheel," *Nature,* vol. 329, pp. 619-620, 1987.

[30] T. Hogg and B.A. Huberman, "Dynamics of Large Autonomous Computational Systems," *Proc. Santa Fe Workshop Collective Cognition,* 2002.

[31] B. Veeravalli, D. Ghose, V. Mani, and T.G. Robertazzi, *Scheduling Divisible Loads in Parallel and Distributed Systems.* IEEE CS Press, 1996.

[32] D. Yu and T.G. Robertazzi, "Divisible Load Scheduling for Grid Computing," *Proc. IASTED Int'l Conf. Parallel and Distributed Computing and Systems (PDCS 2003),* 2003.

[33] H.E. Bal et al., "The Distributed ASCI Supercomputer Project," *ACM Special Interest Group, Operating Systems Rev.,* vol. 34, no. 4, pp. 76-96, Oct. 2000.

[34] "The Distributed ASCI Supercomputer (DAS)," http://www.cs.vu.nl/das/, 2005.

[35] W.M. Jones, L.W. Pang, and D. Stanzione, "Computational Mini-Grid Research at Clemson University," Technical Report PARL-2002-009, Parallel Architecture Research Laboratory, Clemson Univ., Dec. 2002.

[36] W.M. Jones, L.W. Pang, D. Stanzione, and I. Walter, and B. Ligon, "Job Communication Characterization and Its Impact on Meta-Scheduling Co-Allocated Jobs in a Mini-Grid," *Proc. Third Int'l Workshop Performance Modeling, Evaluation, and Optimization of Parallel and Distributed Systems (PMEO-PDS 2004),* Apr. 2004.

[37] X. Jin and J. Liu, "Characterizing Autonomic Task Distribution and Handling in Grids," *Eng. Applications of Artificial Intelligence,* vol. 17, no. 7, pp. 809-823, 2004.

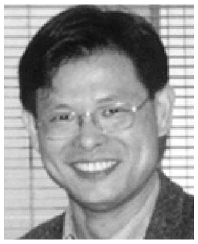
[38] Y. Wang, J. Liu, and X. Jin, "Modeling Agent-Based Load Balancing with Time Delays," *Proc. 2003 IEEE/WIC Int'l Conf. Intelligent Agent Technology (LAT '03),* pp. 189-195, Oct. 2003.

[39] Y. Wang and J. Liu, "Macroscopic Model of Agent Based Load Balancing on Grids," *Proc. Second Int'l Joint Conf. Autonomous Agents and Multi-Agent Systems (AAMAS '03),* July 2003.

[40] J. Liu, X. Jin, and Y. Wang, "Agent-Based Load Balancing on Homogeneous Minigrids: Macroscopic Modeling and Characterization," Technical Report COMP-04-005, Dept. of Computer Science, Hong Kong Baptist Univ., Sept. 2004.

[41] S.-H. Yook, H. Jeong, and A.-L. Barabasi, "Modeling the Internet's Large-Scale Topology," *Proc. Nat'l Academy of Sciences,* vol. 99, no. 21, pp. 13382-13386, 2002.

[42] A.-L. Barabasi, R. Albert, and H. Jeong, "Scale-Free Characteristics of Random Networks: The Topology of the World Wide Web," *Physica A,* vol. 281, pp. 69-77, 2000.



Jiming Liu received the BSc degree in physics from East China Normal University in Shanghai, China, the MA degree from Concordia University, and the MEng and PhD degrees, both in electrical and computer engineering, from McGill University in Montreal, Canada. He is a professor and the head of the Department of Computer Science at Hong Kong Baptist University (HKBU). He leads the Centre for e-Transformation Research (CTR) and AAMAS/AOC Research Lab (Autonomous Agents and Multi-Agent Systems/Autonomy Oriented Computing) at HKBU. His current research interests include: autonomy oriented computing (AOC), self-organization, Web intelligence (WI), Wisdom Web, semantics oriented computing, multi-agent systems (MAS), emergent behavior and amorphous intelligence, contemporary AI and robotics, and spatial reasoning and planning. He has published more than 170 scientific articles in refereed international journals, books, and conferences. In addition, he has published 23 books. Among them, seven are monograph books. Professor Liu is the Editor-in-Chief of *Web Intelligence and Agent Systems* (IOS Press, The Netherlands), *Annual Review of Intelligent Informatics* (World Scientific Publishing), and *The IEEE Computational Intelligence Bulletin* (IEEE Computer Society TCCI). He has been an associate editor and a guest editor for numerous international journals and magazines. He is a senior member of the IEEE.

Xiaolong Jin received the BSc degree from the Department of Applied Mathematics, Beijing University of Aeronautics and Astronautics in July 1998, and the MEng degree from Academy of Mathematics and System Sciences, Chinese Academy of Sciences in July 2001. He is currently pursuing the PhD degree in computer science at Hong Kong Baptist University. His current research interests include: autonomous agents and multiagent systems, autonomy oriented computing, grid computing, peer-to-peer computing, distributed problem solving, satisfiability problems, constraint satisfaction problems, etc.

Yuanshi Wang received the BSc, MSc, and PhD degrees in mathematics from Zhongshan University in 1984, 1989, and 2001, respectively. He is currently an associate professor with the School of Mathematics and Computational Science, Zhongshan University, Guangzhou, China. His main research interests include: evolutionary computation, evolutionary games, replicator dynamics, and their applications.