Modeling Agent-Based Load Balancing with Time Delays

Yuanshi Wang
Department of Mathematics
Zhongshan University
Guangzhou, China
mcswys@zsu.edu.cn

Jiming Liu, Xiaolong Jin
Department of Computer Science
Hong Kong Baptist University
Kowloon Tong, Hong Kong
{jiming, jxl}@comp.hkbu.edu.hk

Abstract

In grid computing, agent-based load balancing is one of the most important problems. In this paper, we present a macroscopic model to describe the dynamics of agent-based load balancing with time delays. We concern the number and size of teams where tasks queue. The time gap, during which a single agent searches a suitable node and transfers a task to the node, is incorporated into balancing process as delay. Our model is composed of functional differential equations. By numerical simulations, we show that variables (the number and size of teams, etc.) in the model remain nonnegative, which is in agreement with the physical background of the variables. We show that although there is a period of oscillation, the dynamic behavior tends to a steady state, which is in agreement with the recent experiments on Anthill. An interesting phenomenon is shown: the larger the delay, the longer the period of oscillation, and the slower the converging speed of load balancing.

1. Introduction

The next generation of e-business is led by community computing under the concept of having computing resources over the Internet integrated and shared [1, 2, 7]. One of the key technologies in community computing is grid computing. Grid computing is originally motivated from large-scale scientific computation where supercomputers are often needed. Scientists try to integrate idle computers on networks to a "computing grid" to replace supercomputers. After a large-scale scientific computation is decomposed into different independent tasks, the remained problem is to disperse the tasks, that is, to balance the tasks on different idle computers on networks.

Traditionally there is a master who disperses tasks to his slaves. This is not applicable in grid computing as the networks of idle computers lack fixed structures. In a natural environment, a group of ants can collect objects into piles without any master. This phenomenon gives a clue to solve our problem. Resnick [12] simulated the phenomenon by artificial ants and found that in order to collect objects into piles, the ants only need to obey three simple rules:

1. An ant wanders around randomly until it encounters an object;
2. If it was carrying an object, it drops the object and continues to wander randomly;
3. If it was not carrying an object, it picks the object up and continues to wander.

The goal of Resnick's artificial ants is to collect objects. In order to disperse tasks on networks, Montresor and Meling [3, 9] built artificial ants where rules are inverse to those in Resnick's simulations:

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 tasks from the most overloaded node to the most underloaded node.

Under the simple rules, the experiments in [3, 9] showed that the artificial ants can disperse tasks evenly. In one of their experiments, there are 100 idle nodes on grids. 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. The experiments in [3, 9] gave empirical simulations of load balancing. However, this kind of microscopic simulations cannot describe the dynamic behavior directly and cannot reflect how the continuous changes of local factors affect the global dynamic behavior. Macroscopic models, on the other hand, can offer such analysis [4, 5, 6, 8, 10, 11, 14, 16].

In the simulations of [3, 9], each dispersed task is carried by an ant. Ants search overloaded and underloaded nodes, and transfer tasks from the overloaded nodes to the underloaded nodes. There is a time gap that ants spend on SearchMax, SearchMin and Transfer. The time gap is called time delay.

In this paper, we give functional differential equations, as motivated by the model in [8, 15], to describe the agent-based load balancing process with time delays. In [8, 15], the dynamic behavior of their systems tended to a steady state without incorporating time delays. Our question to be addressed here is what will be the effect of time delays in the dynamic behavior of load balancing. By numerical simulations, we show that variables (the number and size of teams, etc.) in our model remain nonnegative, which is in agreement with the physical background of the variables. We show that although there is a period of oscillations, the dynamic behavior tends to a steady state, which is in agreement with the recent experiments on Anthill in [3, 9]. An interesting phenomenon is shown: the larger the delay, the longer the period of oscillations, and the slower the converging speed of load balancing.

The rest of the paper is organized as follows: In Section 2, we describe the load balancing mechanism with time delays and give assumptions of our model. In Section 3, we present the macroscopic model and a detailed explanation of the model. In Section 4, we show the positiveness and convergence of variables we concern. In Section 5, we show the effects of time delays. Conclusions are in Section 6.

2. Balancing with Time Delays

2.1. Load Balancing Mechanism

The agent-based load balancing we concern here is as follows. Initially a pile of tasks is generated on networks. Then a pile of agents, whose number is equal to that of the tasks, is generated. Each task is carried by an agent. We regard a task as an agent. The agents wander on networks and search proper nodes to join and queue. The time delay is that a single agent spends on wandering in order to meet a node. We concern the number of wandering agents, the number and size of teams where tasks queue.

Agents can be generated from every node on networks. Each node on networks supplies the same service and each agent must be served by one node. The service time is assumed constant for every agent. Agents have local information about the team lengths on networks, but they do not have the global knowledge on the state of grids.

Agents wander among nodes to search for small teams. Single agents independently make decisions from their own experiences. They will not join a very large team because of their patience. That is, there is a maximum size m for teams. In the simulations of [3, 9], a time period is needed for ants' SearchMax, SearchMin, Transfer. Since we look each task as an agent, the time period becomes that a single agent spends on wandering. The time period is assumed to be a positive constant τ. An agents can leave or join the team in a service node, or form a new team if there is no other agents there. After joining a team, a single agent can also leave the team and moves to other nodes. Therefore, agents' behavior can be decomposed into two elements: leaving and queuing. Local factors, such as time delays, initial conditions, strategies for leaving and queuing will determine the global dynamic behavior of the system. The following mechanism shows the formation of load balancing with time delays.

---

00 load balancing with time delays;
01 begin
02 Single Agent (SA) position during initial
03 period $[0, \tau]$: SA either queues at a team of
04 size s_0 or wanders on nerworks;
05 if $s_0 > m$,
06 SA leaves the team of size s_0;
07 the size of the team becomes $s_0 - 1$;
08 goto START;
09 endif
10 if SA does not leave the team of size s_0,
11 goto EXIT;
12 endif
13 the size of the team becomes $s_0 - 1$;
14 START: SA wanders randomly on networks
15 for a period τ and meets a team of size s;
16 if $s > m$ or SA leaves the team of size s
17 goto START;
18 endif
19 if SA joins the team of size s,
20 the size of the team becomes $s + 1$;
21 endif
22 if SA leaves the team of size $s + 1$,
23 the size of the team becomes s;
24 goto START;
25 endif
26 EXIT: SA queues in a team and does not leave;
27 end

2.2. Main Problems

The first problem we concern is to present a macroscopic model to describe the load balancing mechanism in Section

2.1.

The second problem is about the basic properties of the model, such as stability and variables' positiveness. Here, the stability is shown in the experiments on Anthill [3, 9], the positiveness is important to the effectiveness of the model.

The third problem is to study load balancing through the model. In this paper, we concern the effects of time delays on dynamics of load balancing.

2.3. Assumptions

The primitive strategies and additional proper assumptions of our model are listed below:

1. The time that a single agent spends on wandering is assumed to be a positive constant τ. Agents follow the same strategies of leaving and queuing and occupy the same service time. That is, all the agents are peer-to-peer.

2. On networks, every node can generate tasks (agents) and supply the same service which is needed by agents. That is, all the nodes are peer-to-peer.

3. During the initial period $[0, \tau]$, agents either queue in various nodes or wander on networks.

4. When wandering, a single agent meets service nodes on networks randomly.

5. It is beneficial for agents to queue at small teams.

6. Each agent will not queue at a node whose team is of maximum size m. Agents independently decide leaving or queuing at an encountered node.

7. The total number of agents in the system remains constant at time $t \in (\tau, +\infty)$.

3. The Dynamic Model

In this section, we construct a macroscopic model to describe the dynamic behavior of agent-based load balancing with time delays as described in Section 2. The model is based on several quantities: the number of wandering agents, the number and size of teams. The time delay is defined as that a single agent spends on wandering in order to meet a node.

Let y denote the number of wandering agents. Let y_s denote the number of teams whose size is s. Let m denote the maximum team size. Then

$$y \geq 0, y_s \geq 0, 1 \leq s \leq m.$$

Initially agents are generated from nodes on grids, that is, the system consists of S agents which either queue in various nodes or wander on the network, it follows from the seventh assumption that

$$y(t) + \sum_{s=1}^{m} s y_s(t) = S, t \in [0, \tau].$$

During the initial period $[0, \tau]$, the maximum team size may be larger than m. It follows from the seventh assumption in Section 2 that the agents, who are in the lth ($l \geq m+1$) position of various teams, will leave the teams. The leaving state is concurrent 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 lth ($l \geq m+1$) agents leave their positions, the maximum team size in the system will not be larger than m. Therefore we focus on the case that the maximum team size is not larger than m.

The following macroscopic model is composed of functional differential equations with time delays, similar differential equations without time delays have been discussed in [8, 13, 15]:

$$\begin{aligned}
\frac{dy_1(t)}{dt} &= \lambda y(t-\tau) + d_2 y_2(t) - c_1 y(t-\tau) y_1(t), \\
\frac{dy_s(t)}{dt} &= d_{s+1} y_{s+1}(t) + c_{s-1} y(t-\tau) y_{s-1}(t) \\
&\quad - d_s y_s(t) - c_s y(t-\tau) y_s(t), \\
&\quad 2 \leq s \leq m-1, \qquad (1) \\
\frac{dy_m(t)}{dt} &= c_{m-1} y(t-\tau) y_{m-1}(t) - d_m y_m(t), \\
y(t) &+ \sum_{s=1}^{m} s y_s(t) = S,
\end{aligned}$$

where

$$\tau > 0, 0 < \lambda < 1,$$

$$0 < c_s < 1, 0 < d_s < 1,$$

$$y_s \geq 0, y \geq 0,$$

$\frac{dy_s(t)}{dt}$ is the change rate of teams of size $s, 1 \leq s \leq m, t \in (\tau, +\infty)$.

Parameter τ is a positive constant which carries the time delay effect into load balancing process.

Parameter λ is the rate at which a wandering agent meets an idle node and forms a new team of size one, therefore we have $\lambda < 1$.

Parameter c_s is the rate at which a wandering agent meets and joins a team of size s, therefore we have $c_s < 1$.

Parameter d_s is the rate at which a queuing agent in a team of size s leaves, therefore we have $d_s < 1$. Both c_s and d_s are determined by agents' experience after they wander on networks for a period.

The terms in system (1) show the load balancing process with time delays.

In the first equation of (1), the change rate of teams of size one is described. The term $\lambda y(t-\tau)$ denotes that a wandering agent meets an idle node and forms a new team of size one at time t.

The term $d_2 y_2(t)$ denotes that a team of size two at time t becomes a team of size one after an agent's leaving.

The term $-c_1 y(t-\tau)y_1(t)$ denotes that a team of size one becomes a team of size two at time t after a wandering agent's queuing.

In the second equation of (1), the change rate of teams of size s is described, $2 \le s \le m-1$. The term $d_{s+1}y_{s+1}(t)$ denotes that a team of size $s+1$ becomes a team of size s at time t after an agent's leaving.

The term $c_{s-1}y(t-\tau)y_{s-1}(t)$ denotes that a team of size $s-1$ becomes a team of size s at time t after a wandering agent's queuing.

The term $-d_s y_s(t)$ denotes that a team of size s becomes a team of size $s-1$ at time t after an agent's leaving.

The term $-c_s y(t-\tau)y_s(t)$ denotes that a team of size s becomes a team of size $s+1$ at time t after a wandering agent's queuing.

In the third equation of (1), the change rate of teams of size m is described. Here, single agents would not join a team of size m, and there is no teams of size $m+1$. The term $c_{m-1}y(t-\tau)y_{m-1}(t)$ denotes that a team of size $m-1$ becomes a team of size m at time t after a wandering agent's queuing.

The term $-d_m y_m(t)$ denotes that a team of size m becomes a team of size $m-1$ at time t after an agent's leaving.

The equation

$$y(t) + \sum_{s=1}^{m} s y_s(t) = S$$

denotes that there is no neat change in the number of agents (tasks) at time t, $t \in (\tau, +\infty)$.

4. Positiveness and Stability

Since variables we concern are the number of wandering agents and the number of teams, then the variables

$$y(t), y_s(t), 1 \le s \le m$$

should be nonnegative. On the other hand, it follows from the experiments of [3, 9] that the state of load balancing

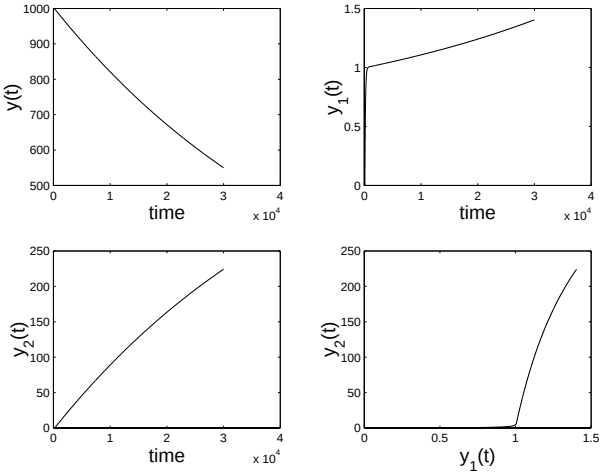

Figure 1. Let $S = 1000, \lambda = c = d = 0.001, \tau = 1, y(t) = 1000, y_1(t) = y_2(t) = 0$ as $t > 0$. **Load balancing tends to a steady state and variables remain nonnegative.**

tends to a steady state after tens of iterations, that is, the dynamic behavior converges to a steady state.

In this section, we let $m = 2$. By numerical integrations, we show that the system (1) is in agreement with the above two properties:

1. Variables $y(t), y_1(t), y_2(t)$ remains nonnegative as $t > 0$;
2. Variables $y(t), y_1(t), y_2(t)$ tend to a steady state as $t \to +\infty$.

The first aspect shows that the number of agents would not be negative in system (1), which is in agreement with the physical background of the variables. The second aspect shows that the dynamic behavior of the system (1) will tend to a steady state, therefore the load balancing will tend to a certain distribution.

Let $m = 2$, then the system (1) becomes:

$$\begin{aligned} \frac{dy_1(t)}{dt} &= \lambda y(t-\tau) + d y_2(t) - c y(t-\tau) y_1(t), \\ \frac{dy_2(t)}{dt} &= c y(t-\tau) y_1(t) - d y_2(t), \\ y(t) &+ \sum_{s=1}^{2} s y_s(t) = S. \end{aligned} \tag{2}$$

In order to make our numerical integrations obviously, variables $y(t), y_1(t), y_2(t)$ are assumed to remain constant in the first time period $[0, \tau]$. To show the positiveness of $y(t), y_1(t), y_2(t)$, we consider the following four cases,

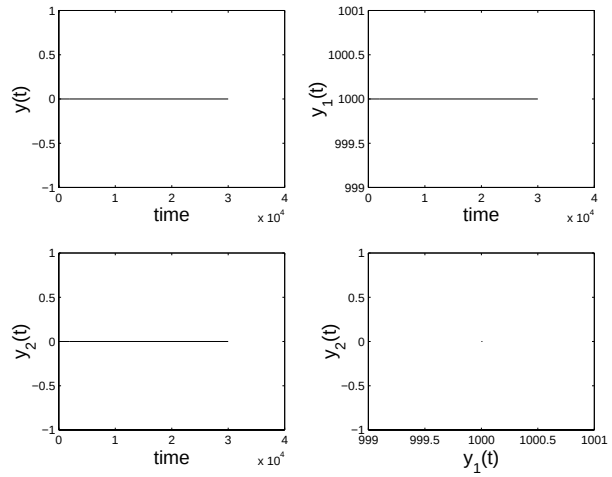

Figure 2. Let $S = 1000, \lambda = c = d = 0.001, \tau = 1, y_1(t) = 1000, y(t) = y_2(t) = 0$ as $t > 0$. **Load balancing is at a perfectly balanced steady state.**

where the first three cases are on boundary. Other cases on boundary can be discussed similarly.

Simulation 1. Let

$$S = 1000, c_1 = d_2 = 0.001, \tau = 1,$$

$$y(t) = 1000, y_1(t) = y_2(t) = 0 \text{ as } t \in [0, \tau].$$

That is, $y_1(t), y_2(t)$ are on boundary as $t \in [0, \tau]$. The simulation in Fig. 1 shows that although $y_1(t) = y_2(t) = 0$ as $t \in [0, \tau]$, they become positive as $t > \tau$, that is, they leave the boundary of $y_1 = y_2 = 0$. By the way, $y(t)$ remains positive as $t > \tau$. Therefore, the simulation shows that solutions of (2) initiated from boundary $y_1(t) = y_2(t) = 0$ as $t \in [0, \tau]$, remain positive as $t > \tau$. It follows from Fig. 1 that variables $y(t), y_1(t), y_2(t)$ tend to steady state as t tends to infinity. Here $y(t)$ decreases smoothly, $y_2(t)$ increases smoothly. There is a time t_0, as $t \in [\tau, t_0]$, $y_1(t)$ increases rapidly, which means that during the time period $[\tau, t_0]$, many wandering agents find idle node and form teams of size one. As $t \in [t_0, +\infty]$, $y_1(t)$ increases not so rapidly, which means that during the time period $[t_0, +\infty]$, there are not so much idle nodes for wandering agents to find and teams of size one increase not so rapidly.

Simulation 2. Let

$$S = 1000, c_1 = d_2 = 0.001, \tau = 1,$$

$$y_1(t) = 1000, y(t) = y_2(t) = 0 \text{ as } t \in [0, \tau].$$

That is, $y(t), y_2(t)$ are on boundary as $t \in [0, \tau]$. The simulation in Fig. 2 shows that although $y(t) = y_2(t) = 0$ as $t \in [0, \tau]$, they are nonnegative as $t > \tau$, that is, they are on the boundary of $y(t) = y_2(t) = 0$. An interesting phenomenon emerges on this case where a perfect balancing exists: if the load distribution is on a perfect balancing, it follows from our model that the load balancing behavior will remain at the perfect balancing state forever. By the way, one can easily prove that perfect balancing states are equilibria of system (1). It follows from Fig. 2 that variables $y(t), y_1(t), y_2(t)$ tend to steady state as t tends to infinity.

Simulation 3. Let

$$S = 1000, c_1 = d_2 = 0.001, \tau = 1,$$

$$y_2(t) = 500, y(t) = y_1(t) = 0 \text{ as } t \in [0, \tau].$$

That is, $y(t), y_1(t)$ are on boundary as $t \in [0, \tau]$. The simulation in Fig. 3 shows that although $y(t) = y_1(t) = 0$ as $t \in [0, \tau]$, they become positive as $t > \tau$, that is, they leave the boundary of $y = y_1 = 0$. By the way, $y_2(t)$ remains positive as $t > \tau$. Therefore, the simulation shows that solutions of (2) initiated from boundary $y(t) = y_1(t) = 0$ as $t \in [0, \tau]$, remain positive as $t > \tau$. It follows from Fig. 3 that variables $y(t), y_1(t), y_2(t)$ tend to steady state as t tends to infinity. Here, there is a time t_0 where $y(t)$ reaches its maximum, which means that since there are too many teams of size two (maximum team size) during initial time period $[0, \tau]$, then many agents leave their teams and become wandering agents. Similar discussions can be given for the decrease of $y_2(t)$ and the increase of $y_1(t)$.

Simulation 4. Let

$$S = 1000, c_1 = d_2 = 0.001, \tau = 1,$$

$$y_2(t) = 400, y(t) = y_1(t) = 100 \text{ as } t \in [0, \tau].$$

That is, $y(t), y_1(t), y_2(t)$ are all positive as $t \in [0, \tau]$. The simulation in Fig. 4 shows that $y(t), y_1(t), y_2(t)$ are all positive as $t > \tau$. Therefore, the simulation shows that solutions of (2) initiated from $y(t) > 0, y_1(t) > 0, y_2(t) > 0$ as $t \in [0, \tau]$, remain positive as $t > \tau$. It follows from Fig. 4 that variables $y(t), y_1(t), y_2(t)$ tend to steady state as t tends to infinity. Here, there is a time t_0 where $y_1(t)$ reaches its minimum, which means there is a small number of teams of size one.

5. Effects of Time Delays

Since a single agent has to spend time τ to search a node, the length of τ affects the task allocation. In this section, we show the effects of time delays by numerical simulations.

Let $m = 2$, then the system (1) becomes the system (2). We consider the following two cases.

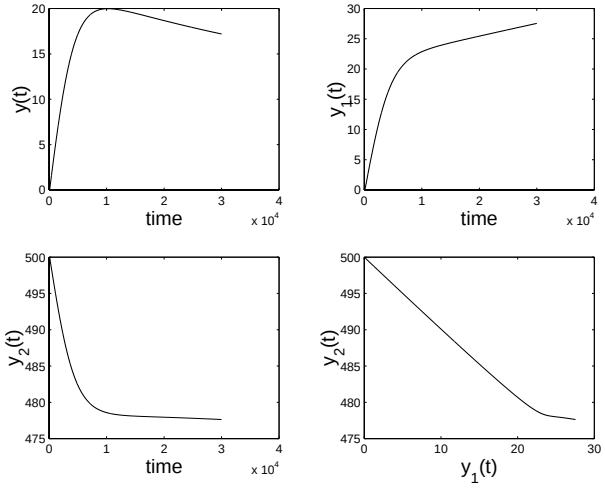

Figure 3. Let $S = 1000, \lambda = c = d = 0.001, \tau = 1, y_2(t) = 500, y(t) = y_1(t) = 0$ as $t > 0$. **Load balancing tends to a steady state and variables remain positive.**

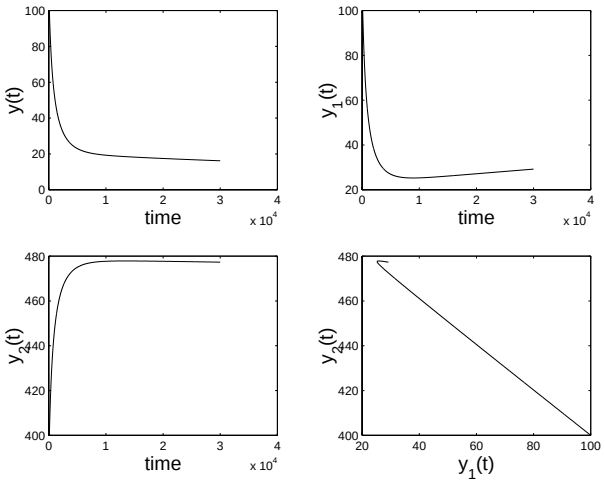

Figure 4. Let $S = 1000, \lambda = c = d = 0.001, \tau = 1, y_2(t) = 400, y(t) = y_1(t) = 100$ as $t > 0$. **Load balancing tends to a steady state and variables remain positive.**

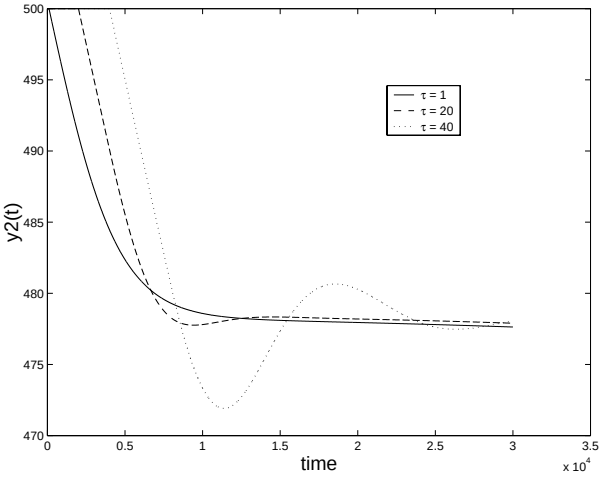

Figure 5. Let $S = 1000, \lambda = c = d = 0.001, \tau = 1, 20, 40, y_2(t) = 500, y(t) = y_1(t) = 0$ as $t > 0$. **the larger the time delay τ, the slower the converging speed of task allocation.**

Simulation 5. Let

$$S = 1000, \lambda = c = d = 0.001,$$

$$y_2(t) = 500, y(t) = y_1(t) = 0 \text{ as } t \in [0, \tau],$$

that is, there are a total of 1000 tasks on networks and there are 500 teams of size 2 in initial period $[0, \tau]$. Simulations in Fig. 5 show that as τ increases, the converging speed of $y_2(t)$ is decreased, more and more oscillations emerge. Similar simulations can be given that the converging speed of variables $y(t), y_1(t)$ vary as time delay τ varies: the larger the time delay τ, the slower the converging speeds.

Simulation 6. Let

$$S = 1000, \lambda = c = d = 0.001,$$

$$y_2(t) = 400, y(t) = y_1(t) = 100 \text{ as } t \in [0, \tau],$$

that is, there are a total of 1000 tasks on networks and there are 400 teams of size 2, 100 teams of size 1 and 100 wandering agents in initial period $[0, \tau]$. Simulations in Fig. 6 show that the larger the time delay τ, the longer the period of oscillation.

In Figs. 5-6, as τ is small, such as $\tau < 1$, the time delay plays an unimportant role in dynamics of load balancing. However, as τ is large, such as $\tau > 20$, the time delay plays an important role in dynamics of load balancing: oscillation emerges; converging speed of load balancing decreases.

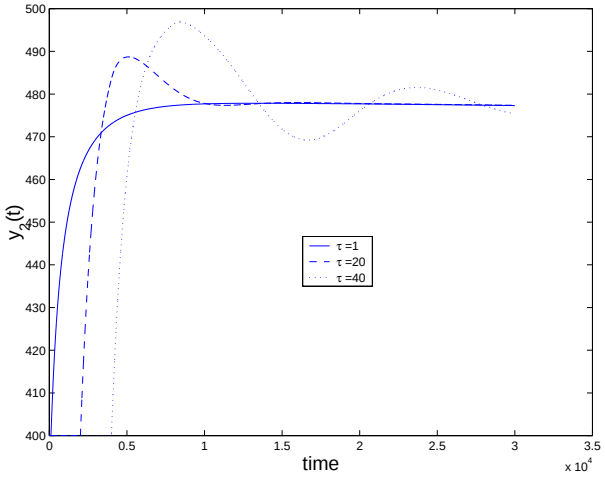

Figure 6. Let $S = 1000, \lambda = c = d = 0.001, \tau = 1, 20, 40, y_2(t) = 400, y(t) = y_1(t) = 100$ as $t > 0$. **the larger the time delay** τ**, the longer the period of oscillation.**

6. Conclusion

In this paper, we have presented a macroscopic model to describe the dynamics of agent-based load balancing with time delays, where the time period among ants' SearchMax, SearchMin, and Tranfer is considered. We give a detailed explanation of our model to establish the close relationship with agent-based load balancing mechanism. To show the stability and variables' positiveness, we give simulations with different initial conditions. The stability we show is in agreement with the recent experiments of [3, 9]; The positiveness that we show here is important to the effectiveness of the model. About time delays, we give an interesting result that as τ is small, the time delay plays a less significant role in the dynamics of load balancing. However as τ is large, the effects of the time delay are obvious: oscillation emerges; converging speed of load balancing decreases.

There are some open problems with respect to the model:

1. The positiveness of variables is not proven strictly;
2. How do steady states vary with initial conditions?
3. How do steady states vary with single agents' strategies of leaving and queuing?
4. To apply the macroscopic model to microscopic simulations, parameters in the model should be determined.

Finally, our model describes load balancing from local factors such as time delays, strategies of leaving and queuing, etc., and shows the global dynamics as affected by the local factors. The model shows the effects of time delays, the convergence of dynamic behavior, the steady states, etc., which are very important to the development of future load balancing systems.

References

[1] 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*, 18:1061–1074, 2002.

[2] W. Agassounon and A. Martinoli. A macroscopic model of an aggregation experiment using embodied agents in groups of time-varying sizes. *In Proceeding of the 2002 IEEE Systems, Man and Cybernetics Conference*, 2002. Tunisia.

[3] O. Babaolu, H. Meling, and A. Montresor. A framework for the development of agent-based peer-to-peer systems. *Proceedings of the 22th International Conference on Distributed Computing Systems*, 2002.

[4] J. Hofbauer and K. Sigmund. *Evolutionary Games and Replicator Dynamics*. Cambridge University, 1998.

[5] T. Hogg and B. A. Huberman. Dynamics of large autonomous computational systems. *Proceedings of the Santa Fe Workshop on Collective Cognition*, 2002.

[6] F. C. Hoppensteadt and C. S. Peskin. *Mathematics in Medicine and the Life Sciences*. Springer-Verlag, 1992.

[7] A. Itai, M. Rodeh, and H. Shachnai. The passport control problem or how to keep a dynamic service system load balanced? *Theoretical Computer Science*, 282:303–318, 2002.

[8] K. Lerman and O. Shehory. Coalition formation for large-scale electronic markets. *Proceedings of the International Conference on Multi-Aent Systems*, 2000.

[9] A. Montresor, H. Meling, and O. Babaoglu. Load-balancing through a swarm of autonomous agents. *Technical Report UBLCS-02-08*, 2002.

[10] W. P. Nelson and A. S. Perelson. Mathematical analysis of delay differential equation model of hiv-1 infection. *Mathematical Biosciences*, 179:73–94, 2002.

[11] 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*, 329, 1987.

[12] M. Resnick. *Turtles, Termites, and traffic Jams: Explorations in Massively Parallel Microworlds*. MIT Press, 1994.

[13] Y. Takeuchi. *Global Dynamical Properites of Lotka-Volterra Systems*. World Scientific, 1996.

[14] A. M. Turing. *The Chemical Basis of Morphogenesis*, volume B327. Phil. Trans. Royal Soc. Lond., 1995.

[15] Y. Wang and J. Liu. Macroscopic model for load balancing on grids. *Proceedings of the International Conference on AAMAS*, July 2003.

[16] Z. Zhang. *Qualitative Theory of Differential Equations*, volume RI 101. AMS, 1992.