The Rich Get Richer: Preferential Attachment in the Task Allocation of Cooperative Networked Multiagent Systems With Resource Caching

Yichuan Jiang, *Member, IEEE*, and Zhichuan Huang

***Abstract*—In ne o ked m l iagen tem (NMAS) i h e- o ce caching, e o ce eplica a e cached in fa o of he agen ho acce ed ch e o ce mo ecen l and f eq en l . Ta k e ec ion in NMAS i de c ibed h o gh agen ' ope a ion hen acce ing nece a e o ce di ib ted in he ne o k , and h , agen i h iche e pe ience e ec ing a k ill ha e highe acce o e o ce . To op imi e a k ' e o ce acce ime, e in e iga e o pe of p efe en ial a achmen in he a k alloca ion of NMAS i h e o ce caching: hi o and p e en p efe en ial a achmen , in hich an agen ha highe acce o a e o ce if ha agen ha iche hi o (o p e en) acce ing e pe ience fo ha e o ce. The efo e, agen ha e e (o a e) hea il b dened b a k ma ha e ce ain p efe en ial igh o ne a k in he f e. O e pe imen fo nd ha p efe en ial a achmen in a k alloca ion can effec i el ed ce a k ' e ec - ion ime, pa ic la l hen he ne o k con e i con ide ed and he n mbe of a k i high. In addi ion, e di co e ed o in e e ing phenomena: 1) Comp omi e be een p efe en ial a achmen and load balancing can achie e be e pe fo mance han ingle p efe en ial a achmen hen he e a e oo man a k ai ing, and 2) he in eg a ion of hi o and p e en p efe en ial a achmen can o pe fo m ei he hi o o p e en p efe en ial a achmen alone in a k alloca ion.**

***Index Terms*—Di ib ted em , load balancing, ne o ked m l iagen tem (NMAS), p efe en ial a achmen, e o ce caching, a k alloca ion.**

Manuscript received September 19, 2010; revised May 23, 2011 and September 18, 2011; accepted December 27, 2011. Date of publication March 2, 2012; date of current version August 15, 2012. This work was supported in part by the National Natural Science Foundation of China under Grants 61170164 and 60803060, the Specialized Research Fund for the Doctoral Program of Higher Education of State Education Ministry of China under Grants 200802861077 and 20090092110048, the General Program of Humanities and Social Sciences in University of State Education Ministry of China under Grant 10YJCZH044, the Program for New Century Excellent Talents in University of State Education Ministry of China under Grant NCET-09-0289, and the Special Fund for Fast Sharing of Science Paper in Net Era by Center of Science and Technology Development of State Education Ministry of China under Grant 20110092110053. This paper was recommended by Associate Editor K. T. Seow.

Y. Jiang is with the Key Laboratory of Computer Network and Information Integration of State Education Ministry, School of Computer Science and Engineering, Southeast University, Nanjing 211189, China, and also with the State Key Laboratory for Novel Software Technology, Nanjing University, Nanjing 210093, China (e-mail: jiangyichuan@yahoo.com.cn).

Z. Huang is with the Laboratory for Complex Systems and Social Computing, School of Computer Science and Engineering, Southeast University, Nanjing 211189, China, and also with the State Key Laboratory for Manufacturing Systems Engineering, Xi'an Jiaotong University, Xi'an 710054, China (e-mail: onlyhzc@gmail.com)

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TSMCA.2012.2186439

I. Introduction

TO MEET the increasing demand of large-scale cooperative computational problems, cooperative distributed systems (CDSs) have been investigated [1]–[4]. There are many practical CDSs, such as grids [1], [3], peer-to-peer (P2P) systems [4], and ad hoc networks [2]. In general, many CDSs have the following characteristics.

1) *Nodes are autonomous and cooperative*. Each node behaves autonomously by considering the surrounding situations [5]; the nodes can contribute their idle resources and cooperatively work together to accomplish tasks [6].
2) *Nodes are interconnected through networks, and interactions are local and constrained by network* structures. Network structures can cope with large-scale situations because each node only needs to know its surrounding situations [6], [7]. Moreover, network structures can save nodes' energies because each node only communicates with its neighbors and remote communication can be implemented by forwarding among nodes [2].
3) *Resources are distributed within the networks, and access to resources is crucial to the system performance*. Many CDSs aim to achieve resource sharing, and thus, communication time for accessing resources is crucial [1], [3], [4].
4) *Resource caching can be used to improve the performance of resource access*. Some resources are replicated at certain places in the networks so that the access to those resources is easier [2], [8], [9].

CDSs can be viewed as networked multiagent systems (NMASs) in which agents represent the autonomous nodes and interaction relations represent interconnections among nodes [6]. The concept of NMASs enables users to represent CDSs at an abstract level without needing to worry about the particulars of the target system [6], [7], [10], [11]. An NMAS can be depicted as a graph, $G = \langle A, E \rangle$, consisting of vertices (agents A) and edges (interactions E); some resources are placed within the network and can be accessed by agents for the execution of tasks [12].

Task execution in multiagent systems can be described by the agents' operations when accessing required resources; thus, task allocation is often implemented based on the accessibility of required resources [1], [13]–[16]. In previous work, load balancing was necessary for the allocation of multiple tasks to reduce tasks' waiting time at agents so that tasks could be switched from heavy-burdened agents to light-burdened ones

[1]. If there are too many tasks queuing for an agent, the probability of the agent being assigned new tasks is reduced.

a well-known task sharing protocol in which each agent in a network can be a manager or a contractor at different times or for different tasks [31]. However, managers in this method also need to know the information of negotiated agents. In summary, agent status information needs to be acquired in a timely manner for these related works.

3) ***Task allocation with uncertain information.***

There are some related studies considering task allocation with uncertain and incomplete information. In previous approaches, social or economics techniques are used, such as negotiation, bidding/auction, trust, game theory, etc. For example, Kraus *et al.* [17] present a distributed approach in which a protocol is developed to enable agents to negotiate and form coalitions to execute tasks with uncertain heterogeneous information. Ramchurn *et al.* [18] present trust-based mechanisms of task allocation in the presence of execution uncertainty, which take into account the trust between agents when allocating tasks. Mataric *et al.* [19] use the bidding/auction mechanism to perform task allocation in uncertain environments. Game theory is used in the task allocation of some heterogeneous distributed systems in which the complete information of agent peers is unknown. For example, Grosu and Chronopoulos [21] present a game-theoretic framework for obtaining a user-optimal load balancing scheme in heterogeneous distributed systems. Moreover, in multirobot systems, a dynamic task allocation mechanism is investigated, which allows agents to change their behavior in response to environmental changes or other agents' actions to improve overall system performance [20]. Such a dynamic allocation mechanism needs agents to have sensing and decision-making abilities.

In summary, the aforementioned approaches may bring about heavy costs for agents' computing and communication, which may influence overall system performance, particularly when the number of tasks is high.

4) ***Load balancing in task allocation.***

If too many tasks are allocated to certain agents, the tasks may be delayed and will not receive quick responses. To minimize the amount of time that tasks remain with an agent, tasks may be switched to other agents with lower task loads, which is called *load balancing*. Liu *et al.* [1] present a macroscopic characterization of agent-based load balancing, in which complete information about the number and size of task teams queuing for agents is necessary. Chow and Kwok [32] investigate load balancing for distributed multiagent computing, in which a novel communication-based load balancing algorithm is proposed by associating a credit value with each agent; the credit of an agent depends on its current situation, such as its affinity to a machine, its current workload, its communication behavior, etc. Schaerf *et al.* [12] study adaptive load balancing, in which information on global resource distribution must be known to make global optimal resource selections. Dhakal *et al.* [33] present a regeneration-theory approach to dynamic load balancing in distributed systems in the presence of delays, in which heterogeneity in the processing rates of the nodes is taken into account.

From above, we can see that load balancing is an important idea for the allocation of multiple tasks, where the number of tasks queuing for an agent is the determinative factor of the agent's rights in future task allocation. If there are too many tasks queuing for an agent, the probability of the agent getting new tasks will be reduced [1]. Therefore, the previous load balancing method of task allocation accords with the adage "winner does not take all" [34].

5) ***Our contribution.***

In comparison with the related work, our work makes the following contributions.

a) Previous load balancing methods based on the "winner does not take all" theory can effectively reduce the waiting time of tasks at agents [1]. In contrast to previous work, our "rich get richer" model can effectively reduce the communication time of agents attempting to access resources within the network. In fact, we combine "rich get richer" and "winner does not take all" to implement a compromise between preferential attachment and load balancing, which can achieve better performance than single preferential attachment while there are too many waiting tasks.

b) Previous resource-based task allocation and load balancing methods are often premised on the assumption that the resources needed by the tasks and resources available from the agents are known; therefore, each time that the task allocation is implemented, accurate resource distribution information needs to be acquired timely from the system [6]. In contrast with the previous work, our model is implemented by simply relying on agents' experiences of executing tasks and frees task allocation from knowing the status of available resources. Our model applies well to large dynamic NMASs, in which it is difficult to obtain accurate resource status information timely.

c) Although there are some related studies on the task allocation with uncertain information, they may bring about heavy costs to the agents' computing and communication, which may influence overall system performance, particularly when the number of tasks is high. Our approach avoids bringing about heavy costs for agents' computing and communication because it is implemented by simply relying on agents' experiences of executing tasks.

III. NMASs With Resource Caching

A. Resource Caching in Multiagent Networks

In previous benchmark work on resource caching, two typical methods are used: One is plain caching, which means that resource replicas are placed at the requesting agents [35], and the other is intermediate caching, which means that resource replicas are placed at intermediate agents (the agents between a requesting agent and the agent that holds the requested

resource) [2], [8], [9]. Generally, intermediate caching is more beneficial for the performance of systems in which the allocated agents are not fixed; moreover, with the intermediate caching method, the number of replicas within the network can be more effectively reduced than with the method that places the replicas at the requesting agents.

Resource caching is not our focus in this paper. Therefore, without loss of generality, we present a simple resource caching mechanism based on abstracting from some related intermediate caching methods [2], [8], [9]. In resource caching, cache locations can be dynamically adapted according to the frequency of resource access. We also describe the eclipse mechanism of resource replicas and make the seldom-accessed resource replicas eclipse step by step, which results in saving of agent storage.

1) Resource Caching Mechanism: While an NMAS is initially set up, the locality of resource r_i is called r_i's *original inhabitation locality*, denoted as OL_{ri}. After the system runs, r_i may be replicated at a place in the network, referred to as r_i's *caching locality*, CL_{ri}.

Obviously, the original inhabitation locality of a resource is always fixed, but a resource's caching locality can be changed. *Let two localities in the network be L_i and L_j. Let the shortest path betweenL_i and L_j be $\{L_i, L_{i+1}, \ldots, L_{j-2}, L_{j-1}, L_j\}$. If a resource changes its caching locality from L_j to L_{j-n} $(1 \leq n \leq j-i)$, we can say that the resource replica migrates from L_j to L_i with n hops; the new locality after migration is represented as $L^n_{L_j \to L_i}$.*

Now, we propose a simple resource caching mechanism as Algorithm 1, where T is the set of tasks.

Algorithm 1. Resource caching in task execution.

1) $\forall\, t \in T$:
   1.1) Allocate the principal agent for t, a_t.
   1.2) $\forall\, r \in R_t$:
      If the current locality of accessed resource r is its original inhabitation locality, OL_r:
         1.2.1) Produce a replica of r, which is represented by cr.
         1.2.2) Migrate cr to $L^n_{OL_r \to L_{a_t}}$.
      else Migrate r **to** $L^n_{CL_r \to L_{a_t}}$.
2) **End**.

From Algorithm 1, when a resource in the original inhabitation locality is accessed, we should produce a replica of the resource and move the replica toward the allocated agent with a series of hops. When the resource replica in the caching locality is accessed, we will only need to move the replica toward the allocated agent with a series of hops.

Definition 1: Compactness degree of an agent set. *Given a set of agents, A, whose compactness degree is the inverse of the mean length of the shortest paths between each pair of agents, shown as*

$$CD_A = 1/\left(\left(\sum_{a_i, a_j \in A} d_{ij}\right) / (|A| \cdot (|A|+1))\right) \quad (1)$$

where d_{ij} denotes the length of the shortest path between a_i and a_j and $|A|$ denotes the number of agents in A. Obviously, the higher CD_A is, the more compact the agents in A are.

Theorem 1: Given two agent sets in the multiagent network $G = \langle A, E\rangle$, $A_1, A_2 \subseteq A, |A_1| = |A_2|$; all agents in$A_1$ and A_2 will access resource r. The probability of producing a replica of r by A_1 is $P_{A1}(r)$, and the probability of producing a replica of r byA_2 is $P_{A2}(r)$. We have the following: $CD_{A1} > CD_{A2} \Rightarrow P_{A1}(r) \leq P_{A2}(r)$.

Proof: We use the inductive method to prove Theorem 1.

1) While $|A_1| = |A_2| = 2$, the agent first accessing r in A_i is a_{i1}, and the second agent accessing r in A_i is a_{i2}. Because the agents in A_1 are more compact than the agents in A_2, the probability that a_{12} accesses the resource replica produced by a_{11} is higher than the probability that a_{22} accesses the resource replica produced by a_{21}. Thus, the probability that a_{12} produces a new resource replica is less than the probability that a_{22} produces a new resource replica. Therefore, we have $P_{A1}(r) \leq P_{A2}(r)$.
2) While $|A_1| = |A_2| = k > 2$, we assume $P_{A1}(r) \leq P_{A2}(r)$.
3) While $|A_1| = |A_2| = k+1$, the agent jth-*ly* accessing r in A_i is $a_{i,j}$. According to Step 2, $P_{A_1 - \{a_{1,k+1}\}}(r) \leq P_{A_2 - \{a_{2,k+1}\}}(r)$. Because $CD_{A1} > CD_{A2}$, the probability that $a_{1,k+1}$ accesses the resource replica produced by any agent in $(A_1 - \{a_{1,k+1}\})$ is higher than the probability that $a_{2,k+1}$ accesses the resource replica produced by any agent in $(A_2 - \{a_{2,k+1}\})$. Thus, the probability that $a_{1,k+1}$ produces a new resource replica is less than the probability that $a_{2,k+1}$ produces a new resource replica. Therefore, we have $P_{A1}(r) \leq P_{A2}(r)$.

□

From Theorem 1 and Definition 1, we conclude that the set with more compact agents produces fewer resource replicas. Therefore, we should seek the set with more compact agents to accomplish the task.

2) Eclipse of Resource Caching: To avoid the congestion of redundant resource replicas in the multiagent network, we can let some longtime unused resource replicas be terminated. This process is called the *eclipse of resource caching*.

While a cached resource replica has not been accessed for long time, we let it go back toward its original inhabitation locality step by step. When it arrives at its original inhabitation locality, it can be terminated. We can set a time period T; after every T time, all cached resource replicas in the multiagent network will withdraw to their respective original inhabitation localities with one hop. The cached resource replicas remaining in the network are those that are accessed by agents most recently and frequently. Idle resource replicas will eclipse step by step.

Let $\boldsymbol{N}(i)$ be the maximum number of resource replicas in the system at time i, N_t be the number of tasks arrived at a time unit, N_r be the mean number of resources needed by one task, T be the eclipse time, $|R|$ be the mean number of resources that an agent owns, and $N_{\boldsymbol{c}}(i)$

$$\lim_{i\to\infty} N(i \qquad \lim_{i\to\infty} (N(i) + Nt * Nr - |R| - N(i)) \quad (3)$$

Because the limits of $N(i$ and $N(i)$ due to the eclipse mechanism

$$\lim_{i\to\infty} N(i \qquad \lim_{i\to\infty} N(i)/T$$

$$\lim_{i\to\infty} \qquad r$$

i t

a_i be R_{ai} t is R_t a_i t $\overline{R^t_{a_i}}$ R $_t$ R $_{a_i}$.

a_j a_i a_j is R_{aj} a_j a_i t of a_i t

t A

t a_i and A_t t

---

* a_i / A */

---

1) **Se$_t$** A
2) Queue (Q);
3) **In e $_t$** Queue ($_i$ $_i$
5) $b = 0$;
6) $\frac{A}{t} = \{a_i\}$;
7) $\overline{R^t_a} = Rt - Ra$8) **If**

$N(i$

$\overline{R^t_{a_i}} == \{\}$ $b = 1$;
9) **While** ((!EmptyQueue (Q)) and $(b$) **do**$_{\lim}$

time. Thus, if the principal and assistant agents are executing a task, a new task would have to wait until all required resources are available. Let E_t be the execution time of task t, W_{tj} be the waiting time of task t for resources at agent a_j, and $C(a_t, a_j)$ be the communication time between a_t and a_j. The purpose of the task allocation is to select the agent set A_t to minimize the execution time of t

$$E_t = \sum_{a_j \in A_t} W_{tj} + \sum_{a_j \in A_t, a_j \neq a_t} C(a_t, a_j) \tag{5}$$

under the constraints that agents' resources are limited and each resource can be accessed by only one task at a time. Therefore,

Theorem 2: Given two agents in the network, a_i *and* a_j, a_i *and* a_j *accessed resource* r_k *to execute tasks from time* t_1 *to* t_2. *The accessibility of agent* a_y *to resource* r_k *at time* t_x *is denoted as* $A_y^x(k)$*; the difference between the accessibilities of* a_i *and* a_j *to resource* r_k *at time* t_x *is denoted as* σ_x, $\sigma_x = |A_i^x(k) - A_j^x(k)|$. *Now, we consider the following three situations*.

1) $A_i^1(k) = A_j^1(k)$: If $h_i^{t_1 \to t_2}(k) > h_j^{t_1 \to t_2}(k)$, we have $A_i^2(k) \geq A_j^2(k)$.
2) $A_i^1(k) > A_j^1(k)$: If $h_i^{t_1 \to t_2}(k) > h_j^{t_1 \to t_2}(k)$, we have $\sigma_2 \geq \sigma_1$.
3) $A_i^1(k) < A_j^1(k)$: If $h_i^{t_1 \to t_2}(k) > h_j^{t_1 \to t_2}(k)$, we have $\sigma_2 \leq \sigma_1$ while $A_i^2(k) \leq A_j^2(k)$ or $A_i^2(k) > A_j^2(k)$.

Proof:

1) If $A_i^1(k) = A_j^1(k)$, at time t_1, the minimum length of the shortest path from a_i to any resource in R_k (denoted as r_{ik}) is the same as the one from a_j to any resource in R_k (which is denoted as r_{jk}). According to the resource caching mechanism, the localities of r_{ik} and r_{jk} at time t_2 are $L^{n_i}_{L_{rik} \to t_2}$ and L^{n_j}

Tesoft 9.9406 T(B)(he)ecaus63 474.096 Tm-Tj3855.7(i)255.7(h Tc(and)Tj/F5 1 Tf1.7508 0 TD 0 49(L)Tj/F8 1 T)]TJ/F12Tc()Tj/F8 1 Tf1.00032554 /F1

Based on Theorem 3, an agent will have higher access to a resource if it has a higher *PRF* for that resource. Therefore, the task can be allocated to the agent with the highest *PRF* for the required resources, which can reduce the communication time in (5). After the principal agent is determined based on *PRF*, the remaining process is similar to the *HRF*-based task allocation.

2) C-PRF and C-PRF-Based Task Allocation: As in Section V-A2, for an agent a_i, the resource access situation of its contextual agents' queuing tasks will influence a_i's future access to resources. The nearer a contextual agent is to a_i, the more the contextual agent's resource access situation of the tasks in its queue influences a_i's future access to resources. We present the definition of the contextual PRF (C-PRF) as follows.

Definition 6: The C-PRF of agent a_i for resource r_k is

$$Cp_i(k) = \sum_{a_j \in C_i} \left(\frac{1/d_{ij}}{\sum_{a_j \in C_i}(1/d_{ij})} p_i(k) \right) \tag{12}$$

where C_i is the context of agent a_i, which can be set from its neighboring area to the whole network and d_{ij} is the distance between agent a_i and a_j in the network; $a_i \in C_i$, d_{ii} is less than d_{ij} if $a_j \neq a_i$.

From Definition 6, if an agent's contextual agents' queuing tasks are rich, the agent may have higher future access to such resources even if the agent itself has fewer queuing tasks. Thus, a task can be allocated to the agent that has the highest C-PRF for the required resources, which is called *task allocation based on C-PRF*.

3) On Load Balancing: As mentioned earlier, if an agent possesses more queuing tasks, it may accordingly be assigned more new tasks. However, if too many tasks are crowded on to certain agents, the waiting time may outperform the benefit (reduction in communication time) brought by the resource caching of executing queuing tasks. Thus, *we should apply load balancing in task allocation when the benefit brought by preferential attachment is less than the loss brought by the waiting time of queuing tasks*. The next case study can demonstrate this situation.

Case Study 1: Let a multiagent network be $G = \langle A, E \rangle$ and the communication time between any neighbor agents be t_ε. The resource caching eclipse time period is t_*; now, there is a resource r_k. $\exists a_i \in A$, $s_{ik} = n$; $\forall t \in Q_{ik}$, the execution time of task t is t_t. twill access r_k for once. Now, if a task t_{new} is allocated to a_i, then t_{new} will be executed after the tasks in Q_{ik} are all finished. Then, now, the influence of Q_{ik} on t_{new} includes the following.

1) ***The benefit brought by resource caching: Because $s_{ik} = n$, resource r_k will be cached and migrated toward a_i with n steps; thus; the saved communication time by allocating t_{new} to a_i is nt_ε.***
2) ***The loss brought by resource eclipse within the waiting time: t_{new} will wait for nt_t time to execute, in which the cached resource will migrate back to its original locality for nt_t/t_* steps; thus; the wasted time is $(nt_t/t_*)t_\varepsilon$.***
3) ***The loss brought by waiting time: nt_t.***
4) Therefore, while task t_{new} is allocated to agent a_i for resource r_k and now $s_{ik} = n$, the net benefit is

$$(n - n \cdot t_t/t_*) \cdot t_\varepsilon - n \cdot t_t. \tag{13}$$

When we allocate tasks, we should perform load balancing if the value of (13) is negative. Because t_ε is usually less than t_t in reality, the load balancing should be taken into account when there is a long queue of tasks.

According to Case Study 1, we should perform load balancing when the queue of tasks is too long. Therefore, we can modify (11) as follows:

$$p_i^*(k) = \alpha(s_{ik}) \cdot p_i(k) = \alpha(s_{ik}) \cdot f(s_{ik}) \tag{14}$$

where $\alpha(s_{ik})$ is an attenuation function, $0 \leq \alpha(s_{ik}) \leq 1$; the value of $\alpha(s_{ik})$ decreases monotonically from 1 to 0 with the increase of s_{ik}.

Therefore, to perform load balancing, the definition of *C-PRF* in (12) should also be modified, shown as follows:

$$\begin{aligned} Cp_i^*(k) &= \sum_{a_j \in C_i} \left(\frac{1/d_{ij}}{\sum_{a_j \in C_i}(1/d_{ij})} p_j^*(k) \right) \\ &= \sum_{a_j \in C_i} \left(\frac{1/d_{ij}}{\sum_{a_j \in C_i}(1/d_{ij})} (\alpha(s_{jk}) \cdot f(s_{jk})) \right). \end{aligned} \tag{15}$$

Obviously, preferential attachment-based task allocation with consideration of load balancing can reduce both the waiting time and communication time in (5).

C. History–Present Combined Preferential Attachment

Now we combine the two preferential attachments, referred to as *history–present combined preferential attachment in task allocation*.

Definition 7: History–present combined resource accessing factor (HP-RF) of agent a_i for resource r_k can be defined as

$$hp_i(k) = \lambda_1 \cdot h_i(k) + \lambda_2 \cdot p_i(k). \tag{16}$$

Contextual HP-RF (C-HP-RF) of agent a_i for resource r_k can be defined as

$$C - hp_i(k) = \lambda_1 \cdot Ch_i(k) + \lambda_2 \cdot Cp_i(k). \tag{17}$$

If load balancing is applied, (16) *and* (17) *can be modified as follows*:

$$hp_i^*(k) = \lambda_1 \cdot h_i(k) + \lambda_2 \cdot p_i^*(k) \tag{18}$$

$$C - hp_i^*(k) = \lambda_1 \cdot Ch_i(k) + \lambda_2 \cdot Cp_i^*(k) \tag{19}$$

where λ_1 and λ_2 are used to determine the relative importance of the two types of preferential attachments, $\lambda_1 + \lambda_2 = 1$.

Then, the task can be allocated to the agent that has the highest *HP-RF* or *C-HP-RF* for the required resources, which is called *task allocation based on history–present combined preferential attachment*.

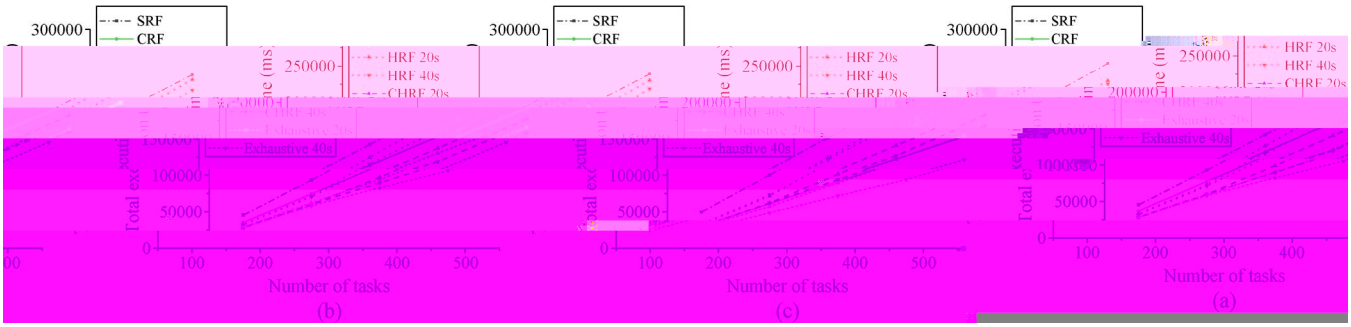


Fig. 1. Execution time comparison between SRF, CRF, exhaustive-trial, *HRF*, and CHRF models. (a) FRFS. (b) MIFS. (c) AAS.

VI. Experiment Validations and Analyses

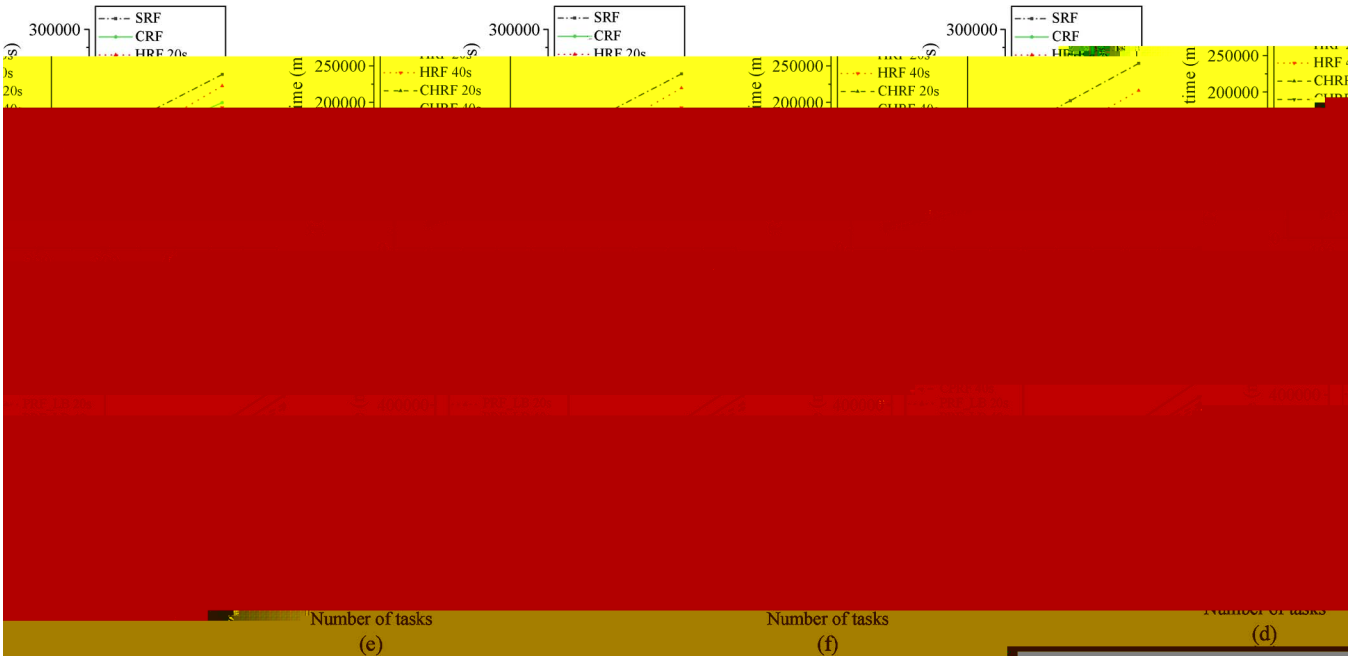


Fig. 2. Execution time comparison between SRF, CRF, exhaustive-trial, PRF, and CPRF models. (a)–(c) do not apply load balancing. (d)–(f) apply load balancing. (a) FRFS. (b) MIFS. (c) AAS. (d) FRFS. (e) MIFS. (f) AAS.

In concl ion, the experiment results prove that the present preferential attachment in task allocation can effectively improve system performance, particularly when the network context situation is considered or the number of tasks is high. Moreover, it is better to compromise between preferential attachment and load balancing while there are too many waiting tasks. Therefore, preferential attachment and load balancing can sometimes be compatible in task allocation.

D. Validation for History–Present Combined Preferential Attachment

Now, we validate the history–present combined preferential attachment in task allocation by comparing the performances of the following models: SRF, CRF, exhaustive-trial, HPRF, and CHPRF. The results are shown in Fig. 3, where HPRF_LB and CHPRF-LB are the HPRF and CHPRF models applying load balancing.

From the experiment results, we conclude the following: 1) HPRF and CHPRF models absolutely outperform the SRF model and are close to the CRF model, which shows that the history–present combined preferential attachment of task allocation can reduce communication time for resource access more than the SRF model; 2) CHPRF outperforms CRF in FRFS and MIFS, which shows the history–present combined preferential attachment is obviously feasible while certain resources are preferential in the task execution; CRF performs very well when AAS is used because CHPRF implements task allocation essentially relying on the unbalanced effects of preferential attachment, but now, AAS averages the overall required resources so that the unbalanced effects in task allocation are relaxed; 3) CHPRF outperforms HPRF, which shows that it is more advanced when the network context is considered; 4) with an increase in the resource caching withdrawal time period or number of tasks, the history–present combined preferential attachment effect becomes more obvious; 5) the HPRF model outperforms both PRF and *HRF* models, and therefore, it is better if we integrate the history and present preferential attachments; and 6) HPRF_LB outperforms HPRF, and CHPRF_LB outperforms CHPRF; therefore, it is better if we can make a compromise between the history–present combined attachment and load balancing while there are many tasks waiting to execute. Moreover, the performance gap between HPRF_LB and HPRF (or between CHPRF_LB and CHPRF) is smaller than that between PRF_LB and PRF (or between CPRF_LB and CPRF in Fig. 2) because HPRF includes the history preferential attachment which does not consider load balancing.

In concl ion, the experiment results prove that the history–present combined preferential attachment in task allocation can effectively improve system performance, particularly when the network context situation is considered and the number of tasks is high. Moreover, the integration of history and present preferential attachments outperforms either history or present preferential attachment alone in task allocation.

E. Comparison With the Idealized Approach

We will compare our approaches with the idealized approach (exhaustive-trial method) by summarizing the related results of the experiments in Section VI-B–D, shown in Table I. The comparison in Table I is measured as follows. Let the task execution time using our approach be x and the time using the idealized approach be y. Their comparison is $c = 1 - ((x - y)/y)$; we can compute the mean of such values in a series of experiments. From Table I, we can see the following: 1) When load balancing is not applied, the mean utilities produced by our approaches are within almost 70%–95% of those produced by

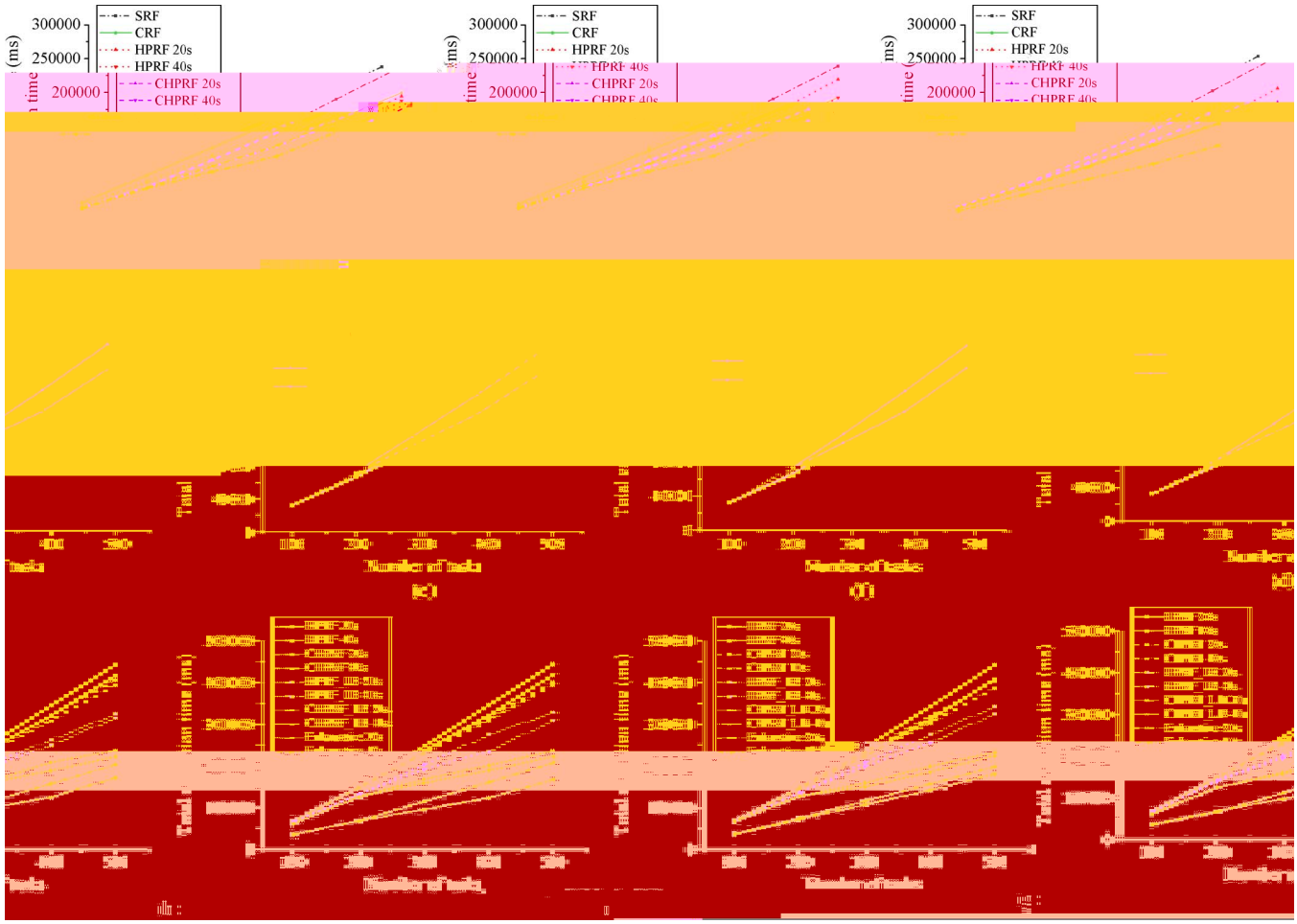


Fig. 3. Execution time comparison between SRF, CRF, exhaustive-trial, PRF, *HRF*, HPRF, and CHPRF models. (a) FRFS. (b) MIFS. (c) AAS. (d) FRFS. (e) MIFS. (f) AAS. (g) FRFS. (h) MIFS. (i) AAS.

TABLE I
COMPARISON BETWEEN OUR APPROACHES AND THE IDEALIZED APPROACH (EXHAUSTIVE-TRIAL METHOD)

Resource satisfaction	Experimental stages	HRF	PRF	HPRF	PRF-LB	HPRF-LB
FRFS	Mean of late stages	0.891718	0.920678	0.898327	0.764212	0.767864
	Mean of all stages	0.927152	0.893712	0.924346	0.422324	0.446571
MIFS	Mean of late stages	0.920648	0.921125	0.923784	0.776029	0.776567
	Mean of all stages	0.719294	0.800006	0.706501	0.802319	0.793195

the idealized approach, which denotes that our approaches are effective; 2) when load balancing is considered, our approaches cannot produce good results in the early stages of experiments; the potential reason for this is that the highly burdened agents do not have absolute dominance over resource access, so the benefit brought by preferential attachment is less than the loss caused by the waiting time of queuing tasks; and 3) when load balancing is applied, our approaches produce mean utilities within 76%–80% of those produced by the idealized approach at the late stages of experiments. The potential reason for this is that, now, the highly burdened agents have absolute dominance over resource access, so the benefit brought by preferential attachment is more than the loss caused by the waiting time of queuing tasks.

VII. CONCLUSION AND DISCUSSION

In practical NMASs such as grids and P2P systems, communication time for accessing required resources is crucial to system performance. In this paper, we are inspired by the idea

simply relying on the agents' experiences of executing tasks and does not need to know the accurate resource distribution information in the system. Thus, our model could be well applied in large-scale dynamic situations in which accurate resource information is difficult to acquire timely.

Regarding the generality and future work of our model, the following are several aspects for discussion.

1) For simplicity, this paper abstracts the distance between two agents as the hops between them and assumes the distances between any two adjacent agents are the same. Such assumption is reasonable in some applications when the hops between two agents are crucial to their communication, such as sensor networks. However, in other situations, distances in terms of resources may not be the same between adjacent agents; they could be a complex function that depends on the type of tasks to be performed. In such a situation, we simply need to modify the concept of distance and revise the resource-searching algorithm by taking the meaning of distance into account. Therefore, our model can be extended into other real situations in which the distances have more complex meanings.
2) In our model, the agents are assumed to differ essentially in their access to resources. Such an assumption is reasonable in some real situations in which all agents are identical. However, in some circumstances, agents may have different computable functions or other resources that cannot be cached, such as CPU power and memory storage. In such a situation, when we perform task allocation, we can select the principal agent that can satisfy the required computable functions and has the highest *HRF* (or PRF) for required resources. Therefore, our presented model can be extended into other situations in which agents have different computable functions.
3) This paper is only concerned with cooperative agents, i.e., all agents can contribute their idle resources and corporately work together to accomplish tasks. However, in reality there are some selfish multiagent systems in which each agent optimizes its own object independently of the others [13], [21]. In the task allocation of such selfish multiagent systems, automated negotiation [13] or noncooperative game [21] is used in related benchmark works, and equilibrium can be obtained by a distributed noncooperative policy. Therefore, in future work, to extend our model into situations in which agents are selfish, we will also introduce the automated negotiation or game theory in the resource search process when the agents can afford the additional computing and communication costs.
4) Our model is designed for NMASs in which resource access can be improved through caching by executing tasks and communication time for accessing resources is crucial to system performance. In fact, the "rich get richer" method can also be generalized to other systems where preferential attachments exist, e.g., agents have self-learning abilities and can improve their capacities through executing tasks. Therefore, our future work will try to improve the generalizability of the "rich get richer" idea in other NMASs.

REFERENCES

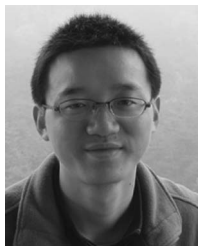
[1] J. Liu, X. Jin, and Y. Wang, "Agent-based load balancing on homogeneous minigrids: Macroscopic modeling and characterization," *IEEE Trans. Parallel Distrib. Syst.*, vol. 16, no. 7, pp. 586–598, Jul. 2005.
[2] L. Yin and G. Cao, "Supporting cooperative caching in ad hoc networks," *IEEE Trans. Mobile Comput.*, vol. 5, no. 1, pp. 77–89, Jan. 2006.
[3] Y. Cardenas, J.-M. Pierson, and L. Brunie, "Uniform distributed cache service for grid computing," in *Proc. 16th Int. Workshop DEXA*, Copenhagen, Denmark, Aug. 22–26, 2005, pp. 351–355.
[4] K. Aberer, M. Punceva, M. Hauswirth, and R. Schmidt, "Improving data access in P2P systems," *IEEE Internet Comput.*, vol. 6, no. 1, pp. 58–67, Jan./Feb. 2002.
[5] H. K. Pillai and S. Shankar, "A behavioral approach to control of distributed systems," *SIAM J. Control Optim.*, vol. 37, no. 2, pp. 388–408, Mar. 1999.
[6] B. Bulka, M. Gaston, and M. des Jardins, "Local strategy learning in networked multi-agent team formation," *J. Autonomous Agents Multi-Agent Syst.*, vol. 15, no. 1, pp. 29–45, Aug. 2007.
[7] S. Abdallah and V. Lesser, "Multiagent reinforcement learning and self-organization in a network of agents," in *Proc. 6th Int. Conf. AAMAS*, Honolulu, HI, May 14–18, 2007, pp. 172–179.
[8] W. Rao, L. Chen, A. W.-C. Fu, and G. Wang, "Optimal resource placement in structured peer-to-peer networks," *IEEE Trans. Parallel Distrib. Syst.*, vol. 21, no. 7, pp. 1011–1026, Jul. 2010.
[9] J. Zhao, P. Zhang, G. Cao, and C. R. Das, "Cooperative caching in wireless P2P networks: Design, implementation, and evaluation," *IEEE Trans. Parallel Distrib. Syst.*, vol. 21, no. 2, pp. 229–241, Feb. 2010.
[10] Y. Jiang, J. Hu, and D. Lin, "Decision making of networked multiagent systems for interaction structures," *IEEE Trans. Syst., Man, Cybern. A, Syst., Humans*, vol. 41, no. 6, pp. 1107–1121, Nov. 2011.
[11] Y. Jiang and Z. Li, "Locality-sensitive task allocation and load balancing in networked multiagent systems: Talent versus centrality," *J. Parallel Distrib. Comput.*, vol. 71, no. 6, pp. 822–836, Jun. 2011.
[12] A. dc77(6h,)-331.6("Decision)-334.]TJang, J("Dechoham,.7(J("Def-)]TJTJM1(ult.334

[27] D. Palmer, M. Kirschenbaum, J. Murton, and K. Zajac, "Decentralized cooperative auction for multiple agent task allocation using synchronized random number generators," in *Proc. IEEE/RSJ, Int. Conf. Intell. Robots Syst.*, Las Vegas, NV, Oct. 2003, pp. 1963–1968.
[28] S. Fjuita and V. R. Lesser, "Centralized task distribution in the presence of uncertainty and time deadlines," in *Proc. 2nd ICMAS*, Kyoto, Japan, Dec. 10, 1996, pp. 87–94.
[29] O. Shehory and S. Kraus, "Methods for task allocation via agent coalition formation," *Artif. Intell.*, vol. 101, no. 1/2, pp. 165–200, May 1998.
[30] R. G. Smith, "The contract net protocol: High-level communication and control in a distributed problem solver," *IEEE Trans. Comput.*, vol. C-29, no. 12, pp. 1104–1113, Dec. 1980.
[31] S. Aknine, S. Pinson, and M. F. Shakun, "An extended multi-agent negotiation protocol," *J. Autonom. Agents Multi-Agent Syst.*, vol. 8, no. 1, pp. 5–45, Jan. 2004.
[32] K.-P. Chow and Y.-K. Kwok, "On load balancing for distributed multiagent computing," *IEEE Trans. Parallel Distrib. Syst.*, vol. 13, no. 8, pp. 787–801, Aug. 2002.
[33] S. Dhakal, M. M. Hayat, J. E. Pezoa, C. Yang, and D. A. Bader, "Dynamic load balancing in distributed systems in the presence of delays: A regeneration-theory approach," *IEEE Trans. Parallel Distrib. Syst.*, vol. 18, no. 4, pp. 485–497, Apr. 2007.
[34] D. M. Pennock, G. W. Flake, S. Lawrence, E. J. Glover, and C. L. Giles, "Winners don't take all: Characterizing the competition for links on the web," in *Proc. Nat. Acad. Sci.*, Apr. 2002, vol. 99, no. 8, pp. 5207–5211.
[35] K. Ranganathan and I. Foster, "Design and evaluation of dynamic replication strategies for a high-performance data grid," in *Proc. Int. Conf. Comput. High Energy Nucl. Phys.*, Beijing, China, Sep. 3–7, 2001, pp. 712–715.
[36] J. Xu, A. Y. S. Lam, and V. O. K. Li, "Chemical reaction optimization for task scheduling in grid computing," *IEEE Trans. Parallel Distrib. Syst.*, vol. 22, no. 10, pp. 1624–1631, Oct. 2011.
[37] A. Padovitz, S. W. Loke, and A. Zaslavsky, "Multiple-agent perspectives in reasoning about situations for context-aware pervasive computing systems," *IEEE Trans. Syst., Man, Cybern. A, Syst., Humans*, vol. 38, no. 4, pp. 729–742, Jul. 2008.
[38] X. Fan, M. McNeese, B. Sun, T. Hanratty, L. Allender, and J. Yen, "Human-agent collaboration for time stressed multi-context decision making," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 40, no. 2, pp. 306–320, Mar. 2010.
[39] Eclipse.org home, 2011. [Online]. Available: http://www.eclipse.org/

Yich an Jiang (M'07) received the Ph.D. degree in computer science from Fudan University, Shanghai, China, in 2005.

He is currently a Professor at the School of Computer Science and Engineering, Southeast University, Nanjing, China. He has published more than 70 scientific articles in refereed journals and conference proceedings, such as IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS, *Journal of Parallel and Distributed Computing*, IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICS PART A: SYSTEMS AND HUMANS, IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICS PART C: APPLICATIONS & REVIEWS, *Proceedings of the International Joint Conferences on Artificial Intelligence*,and *Proceedings of the International Conference on Autonomous Agents and Multiagent Systems*. His main research interests include multiagent systems, social networks, and complex distributed systems.

Dr. Jiang is a Senior Member of China Computer Federation and Chinese Institute of Electronics, a member of the editorial board of Advances in Internet of Things, an Editor of the International Journal of Networked Computing and Advanced Information Management, an Editor of Operations Research and Fuzziology, and a member of the editorial board of the Chinese Journal of Computers.



Zhich an H ang received the B.E. degree in information engineering from Southeast University, Nanjing, China, in 2009, where he is currently working toward the M.E. degree.

His main research interests include social networks and multiagent systems.