Effect of Alternative Distributed Task Allocation Strategy Based on Local Observations in Contract Net Protocol

Toshiharu Sugawara[1], Kensuke Fukuda[2], Toshio Hirotsu[3], and Satoshi Kurihara[4]

[1] Department of Computer Science and Engineering
Waseda University,
Tokyo 1698555, Japan
sugawara@waseda.jp

[2] National Institute of Informatics
Chiyoda, Tokyo 100-000, Japan
kensuke@nii.ac.jp

[3] Faculty of Computer and Information Sciences
Hosei University, Tokyo, Japan
hirotsu@hosei.ac.jp

[4] Institute of Scientific and Industrial Research
Osaka University
kurihara@ist.osaka-u.ac.jp

Abstract. This paper presents a distributed task allocation method whose strategies are alternatively selected based on the estimated workloads of the local agents. Recent Internet, sensor-network, and cloud computing applications are large-scale and fully-distributed, and thus, require sophisticated multi-agent system technologies to enable a large number of programs and computing resources to be effectively used. To elicit the capabilities of all the agents in a large-scale multi-agent system (LSMAS) in which thousands of agents work concurrently requires a new negotiation strategy for appropriately allocating tasks in a distributed manner. We start by focusing on the contract net protocol (CNP) in LSMAS and then examine the effects of the awardee selection strategies, that is, the task allocation strategies. We will show that probabilistic awardee selections improve the overall performance in specific situations. Next, the mixed strategy in which a number of awardee selections are alternatively used based on the analysis of the bid from the local agents is proposed. Finally, we show that the proposed strategy does not only avoid task concentrations but also reduces the wasted efforts, thus it can considerably improve the performance.

Keywords: Distributed task allocation, Adaptive Behavior, Negotiation, Load-balancing.

1 Introduction

Recent Internet technologies enable for advanced large-scale applications, such as e-commerce, grid computing, distributed computing, and cloud computing. Within these applications, thousands of computational entities, called *agents*, have their own tasks,

© Springer-Verlag Berlin Heidelberg 2012

such as user authentication, stock control, customer recommendation, purchasing management, and shipping control in e-commerce applications, work concurrently and collaborate with each other. In this kind of system, which can be modeled as a *large-scale multi-agent system* (LSMAS), these tasks must be appropriately assigned to the agents based on their abilities. However, they often interfere with each other. For example, if many tasks are allocated to only a few specific agents, this may lead to a delay of one of the task fragments (or subtasks), resulting in the delay of the whole task.

On the other hand, new Internet applications have been and will be more dynamic, agents will have different computational resources/abilities, and new services and new servers will frequently come and go. Of course, the agents' states will also change over time. These facts indicate that agents cannot acquire the most accurate global states of the entire system. Thus, the key issue is how the agents will effectively allocate subtasks to other agents using only locally available information so as to exploit the capabilities of the entire system. For this requirement, contract- and auction-based approaches to task and resource allocations [2] have received a lot of attention for future wide-area distributed network applications.

Although a number of researches on (distributed) task allocations have already been conducted such as Ref. [7], we first focus on a task allocation using a *contract net protocol* (CNP) because it is used in many applications [9,16]. In the CNP, an agent plays one of two roles: *managers* who are responsible for allocating tasks and monitoring processes and *contractors* who are responsible for executing the allocated tasks. A manager agent makes a task known to the contractor agents in the announcement phase, and the contractors tender the bid on the task with certain values, such as the cost, estimated duration to process, or required payment, in the bid phase. In the award phase, the manager awards the contractor (or *awardee*) who tendered the best bid.

The objective of our research is to clarify the characteristics of the CNP in a busy LSMAS and to propose contract strategy, more precisely *awarding strategy* in the award phase, resulting in a more efficient cumulative processing of the entire system than the contract strategy in a conventional CNP. This is a challenging issue because interference among agents is intricately intertwined in this kind of negotiation protocol if many managers have tasks to allocate simultaneously. In a naive CNP, a contractor agent responds to the task announcements one by one, but if many managers announce tasks simultaneously in a busy LSMAS, the managers may have to wait a long time to receive a sufficient number of bids. This significantly reduces the performance of the entire system [6]. In the original conception of CNP [12], the use of multiple bids was proposed as a way to concurrently handle many announcements. If a contractor is awarded multiple bids simultaneously, however, it may not be able to provide the quality or performance it declared in the bids. In fact, managers tend to select highly capable contractor agents. Additionally, if the task has a structure, meaning if the task consists of a number of different subtasks, the situation becomes ever more complex.

In this paper, we propose a novel awarding strategy that leads to a more efficient processing of LSMAS. This is a meta-level strategy that selects one from a set of awarding strategies on the basis of the local observations. References [14,15] already tackled this issue, but they assumed a simplified model in which the tasks have no structure, which means a task is singleton and indivisible, and have the same cost. However, our model is more general, that is, a task is plural consisting of a number of subtasks, and their method in [15] cannot be applied to plural tasks.

In addition, another significant issue, wasted efforts, appears in the plural task structure model. Of course, in both the singleton and plural models, we have to avoid task concentrations that lead to inefficient processing and many drops of subtasks. In addition, the failure (or delay) of only one subtask means the failure (or delay) of the whole task in the plural model. Therefore, the agents' resources used for other subtasks become useless. This also significantly reduces the performance of the entire LSMAS. Thus, the required strategy has to be able to reduce the number of wasted efforts as well as the task concentration so that it can considerably improve the overall performance.

This paper is organized as follows: In the next section, we explain our models, the issues to be addressed, and our simulation environment. Then, we introduce the *probabilistic awardee selection strategy*, under which an awardee is selected with certain fixed probabilities based on the bid values. We show that by changing the award strategies according to the local workload, the overall performance can be considerably improved for a specific task consisting of a number of subtasks. After that, we optimize this award strategy under which a probabilistic award strategy and the conventional award strategy are selected alternatively according to the estimated local workloads of the agents within the environments where certain tasks are blended. We experimentally show that the extended strategy can significantly improve the overall performance. Finally, we try to explain the reason for this improvement.

2 Problem Description

2.1 Model of Agents and Tasks

Let $\mathcal{A} = \{1, \dots, n\}$ be a set of agents, $\mathcal{T}$ be a task, and $\mathcal{F} = \{f^1, \dots, f^d\}$ be the set of skills or functions that agents can perform. We assume that task $\mathcal{T}$ consists of subtasks, $t_1, \dots, t_l$, (i.e., $\mathcal{T} = \{t_1, \dots, t_l\}$ and $|\mathcal{T}| = l$) and that subtask $t (\in \mathcal{T})$ requires the $s(t)$-th skill, $f^{s(t)}$, where $1 \le s(t) \le d$. A subtask is denoted by the lower-case letter t and is simply called a task unless this creates confusion. Agent i is expressed as a tuple, $(\alpha_i, L_i, S_i, Q_i)$, where $\alpha_i = (a_i^1, \dots, a_i^d)$ is the set of agent's capabilities (a_i^h corresponds to the h-th skill, f^h, and $a_i^h \ge 0$; $a_i^h = 0$ indicates agent i does not have skill f^h), L_i is the location of i, and Q_i is the queue where the agent's tasks are stored, which are waiting to be executed one by one. The maximum queue length, Q_i, can be finite or infinite, but was assumed 20 in our experiments. The set $S_i (\subset \mathcal{A})$ is i's scope, i.e., the set of agents that i knows. The *metric* between the agents, $\delta(i, j)$, is based on their locations, L_i and L_j, and is used to define the communication time (or delay) of the messages between i and j.

Subtask t has an associated cost, $\gamma(t)$, which is the cost to complete it. Subtask t can be done by i in $\lceil \gamma(t)/a_i^{s(t)} \rceil$ time units, where $\lceil x \rceil$ denotes the ceiling function. The time it takes to complete t is also called the *execution time* of t by i. $\mathcal{T}$ is completed when all its subtasks are completed.

In every unit time, $\mathcal{L} (\ge 0)$ tasks on average are generated according to a Poisson distribution and are randomly assigned to different managers. The parameter $\mathcal{L}$ is called the *task load* and is denoted by $\mathcal{L}$ tasks per unit time, or simply $\mathcal{L}$ T/t.

2.2 Task Allocations for LSMAS

For CNP, we define $\mathcal{M} = \{m_j\}(\subset \mathcal{A})$ as the set of managers who allocate tasks and $\mathcal{C} = \{c_k\}(\subset \mathcal{A})$ as the set of contractors who execute the allocated tasks. Let us assume that $|\mathcal{A}|$ is large (on the order of thousands); therefore, $|\mathcal{M}|$ and $|\mathcal{C}|$ are also large; Moreover, we shall assume that the agents are widely distributed, like servers on the Internet.

In our experiments, we used a CNP modified for use in a LSMAS for the sake of efficiency. In this CNP, (1) multiple bids and *regret* and *no-bid* messages are allowed, and (2) manager m announces each subtask in $\mathcal{T}$ to the contractors that are selected from its scope, S_m, on the basis of an *announcement strategy*. This procedure can reduce the number of messages. *Regret messages* are sent in the award phase to contractors who were not awarded the contract; *no-bid messages* are sent to managers by contractors who decided not to bid on an announced task. These messages prevent long waits for bids and award messages (e.g., [9,17]).

When manager m receives $\mathcal{T}$, it immediately initiates the modified CNP for each task $\tilde{t}(\in \mathcal{T})$. It first sends announcement messages to the contractors selected from its scope. Each of these contractors sends back a bid message with a certain *bid value*. The bid values might include parameters such as the price for executing the task, the quality of the result, or a combination of these values. Since we are concerned with the efficiency of processing using multiple agents, we assume that their bid values contain the estimated required times for completing the task. Thus, the bid value of contractor c is:

$$\lceil \gamma(\tilde{t})/a_c^{s(\tilde{t})} \rceil + \sum_{t \in Q_c} \lceil \gamma(t)/a_c^{s(t)} \rceil + \beta,$$

where β is the time required to complete the task currently being executed. For multiple bidding, c might have a number of outstanding bids. These bids are not considered because it is uncertain whether they will be accepted. Then, m selects a contractor, the awardee, on the basis of the *award strategy* and sends the awardee a message along with the announced task. Selecting the best bidder is the award strategy in the naive CNP.

When contractor c is awarded a task, it immediately executes it if it has no other tasks. If c is already executing a task, the new task is stored in Q_c, and the tasks in Q_c are executed in turn.

2.3 Performance Measures

Since the queue length of agent i is finite, some allocated subtasks might not be storable in its queue and so they are *dropped*. If all the subtasks in $\mathcal{T}$ are dropped, $\mathcal{T}$ is called a *dropped task*. If $\mathcal{T}$ contains both dropped and not-dropped tasks, it is called a *wasting task*. From the viewpoints of the users and clients in actual applications such as e-commerce, dropped and wasting tasks appear as refused or non-responding requests due to the congestion of servers. From the viewpoints of the servers, and thus, the investors who prepared the equipment to provide the services, the wasting task contains wasted efforts, that is, a number of uselessly executed subtasks, whereas the dropped task does not. So, more wasting tasks lowers the performance of the entire LSMAS.

We assume that manager agents can observe, for each subtask t, the *completion time*, which is the elapsed time from the time the award message is sent, $m_s(t)$, to the time the message indicating that the subtask has been completed is received, $m_e(t)$. The completion time thus includes the communication time in both directions, the queue time, and the execution time. The completion time of $\mathcal{T}$ is defined as $\max_{t\in\mathcal{T}}(m_e(t)) - \min_{t\in\mathcal{T}}(m_s(t))$. A smaller average completion time is better. The *overall performance of a LSMAS*, denoted by $\wp$, is defined as the average of the completion times observed by all managers and used as the system's performance measure. The issues we address are thus the overall performance of a LSMAS under various award strategies and how to improve it by combining the advantages of these award strategies. Moreover, dropped and wasting tasks are counted as a separate performance measure; they are non-zero only if the systems are busy or overloaded.

2.4 Simulation Environment

We set $|\mathcal{C}| = 500$ and $|\mathcal{M}| = 10,000$ in our simulation. The agents were randomly placed on a 150×150 grid with a torus topology, which is denoted by G. The Manhattan distance was chosen as G's metric. The communication time ranged from 1 to 14 (in *ticks*, the time unit in the simulation), in proportion to the value of $\delta(i,j)$.

We express the cost structure of the subtasks by using the superscript of $\mathcal{T}$, if necessary. For example, $\mathcal{T}^{25-5}$ consists of two subtasks, $\{t_1, t_2\}$ such that $\gamma(t_1) = 2500$ and $\gamma(t_2) = 500$.[1] Contractor c_i is assigned different capabilities so that the values of $2500/a^1_{c_i}$ ($c_i \in \mathcal{C}$) will be *uniformly distributed* over the range 20–100; the values of $a^1_{c_i}$ range from 25 – 125. Therefore, for $\mathcal{T}^{25-5} = \{t_1, t_2\}$, c_i can execute t_1 and t_2 within 20–100 ticks and 4–20 ticks, respectively. We assume that the manager agents can not do the tasks themselves ($a^1_m = a^2_m = 0$) forcing them to assign the tasks to agents who can, and that $a^1_{c_i} = a^2_{c_i}$; the latter condition means that a high-performance PC can effectively execute any task if the functions are defined.

The results presented here are the mean values from ten independent trials. In these trials, the maximal numbers of $\mathcal{T}$s being executed every tick, as derived from the cumulative capabilities of all contractors $\sum_{c\in C} a_c$, ranged from 8.15 to 8.30 T/t, with an average of 8.25 T/t. This is the theoretical upper limit, meaning that if the task allocation is ideal, the contractors can execute 8.25 tasks every tick.

Manager m's scope, S_m, consists of the nearest 50 contractors. More precisely, for a positive integer n, let $S_m(n) = \{c \in C | \delta(m,c) \leq n\}$. It follows that $S_m(n) \subset S_m(n+1)$. S_m is defined as the smallest $S_m(n)$ such that $|S_m(n)| \geq 50$. Then, m announces tasks to $N (\leq 50)$ contractors who were randomly selected from S_m. The overall performance varied depending on N and was optimal when N was 20 in our simulation environment [13]. Thus, we assume $N = 20$ in what follows so we can focus on the award strategies.

3 Usage of Probability in the Award Phase

A small number of high-capability agents that receive multiple awards will likely bear an excessive workload whenever many managers simultaneously announce numerous

[1] As another example, $\mathcal{T}^{18-8-4}$ means a task consisting of three subtasks $\{t_1, t_2, t_3\}$ whose costs are 1800, 800 and 400, respectively.

tasks. A simple awarding strategy for alleviating the burden of too many awards is to allocate some tasks to the non-best contractor by introducing a probability in the award phase. In this section, we discuss the effect of this type of probabilistic award by comparing it with that for the naive CNP.

3.1 Effect of Probabilistic Award

Reference [15] reported that some degree of fluctuation in the award phase could improve the overall performance when a task has no structure. The objective of the first experiment was to verify this effect when a task consists of a number of subtasks.

Let $\{c_1, \dots, c_p\}$ be the contractors that bid on the announced task. We denote the bid value from contractor c_i by b_{c_i}. In the naive CNP, m selects the contractor who submitted the best bid (a smaller bid is better). The first award strategy selects the awardee according to the following probability:

$$\Pr(c_i) = \frac{1/(b_{c_i})^k}{\sum_{j=1}^{p} 1/(b_{c_j})^k}. \tag{1}$$

This *probabilistic awardee selection* strategy is denoted by PAS$_k$. Non-negative integer k is a parameter called the *fluctuation factor*, or simply the *f-factor*. The larger the k, the smaller the degree of fluctuation: PAS$_0$ and PAS$_\infty$ respectively correspond to "random selection" and "no randomness." Therefore, PAS$_\infty$ is the award strategy in the naive CNP.

We evaluated the overall performance by gradually increasing $\mathcal{L}$ from 0.1 (idle) to 10 (extremely busy, over the cumulative capabilities) in 5-K ticks and then returning it to 0.1. The total duration was 160-K ticks. We plotted the improvement ratios $\mathcal{I}_{CNP}$ from PAS$_k$ to PAS$_\infty$ every 5-K ticks:

$$\mathcal{I}_{CNP}(\mathrm{PAS}_k) = \frac{\wp(\mathrm{PAS}_\infty) - \wp(\mathrm{PAS}_k)}{\wp(\mathrm{PAS}_\infty)} \times 100, \tag{2}$$

where $\wp(str)$ indicates the overall performance when award selection strategy *str* is used. Note that $\mathcal{I}_{CNP}(\mathrm{PAS}_\infty) = 0$.

We assumed $\mathcal{T} = \{t_1, t_2\}$ and examined $\mathcal{T}^{25-5}$ and $\mathcal{T}^{20-10}$; the results are labeled "PAS$_3$" and "PAS$_6$" in Fig. 1. The graphs also list the task loads over time along the horizontal axis. These curves indicate that when task load $\mathcal{L}$ is small (very few multiple awards occur) or very large (over the theoretical limit of cumulative capability), PAS$_\infty$ performs well (PAS$_k$ is worse up to 34%). When $\mathcal{L}$ is in the middle range, PAS$_k$ ($k = 3$ or 6) improves the overall efficiency by as much as 29%. We can thus expect that when the system is busy but does not reach the theoretical limit, PAS$_k$ can avoid the concentration of the workload and maintain the efficiency. However, this is not always true if the system is too busy to process the given tasks. When the system is extremely busy, $\mathcal{I}_{CNP}(\mathrm{PAS}_6)$ for $\mathcal{T}^{25-5}$ is better than $\mathcal{I}_{CNP}(\mathrm{PAS}_6)$ for $\mathcal{T}^{20-10}$, and $\mathcal{I}_{CNP}(\mathrm{PAS}_6) > 0$ for $\mathcal{T}^{25-5}$ but $\mathcal{I}_{CNP}(\mathrm{PAS}_6) < 0$ for $\mathcal{T}^{20-10}$. After a number of experiments, we observed that $\mathcal{I}_{CNP}(\mathrm{PAS}_k)$ was lower in an extremely busy environment if (1) $|\gamma(t_1) - \gamma(t_2)|$ is small or (2) $\gamma(t_1) + \gamma(t_2)$ is large. For the former situation, we introduce

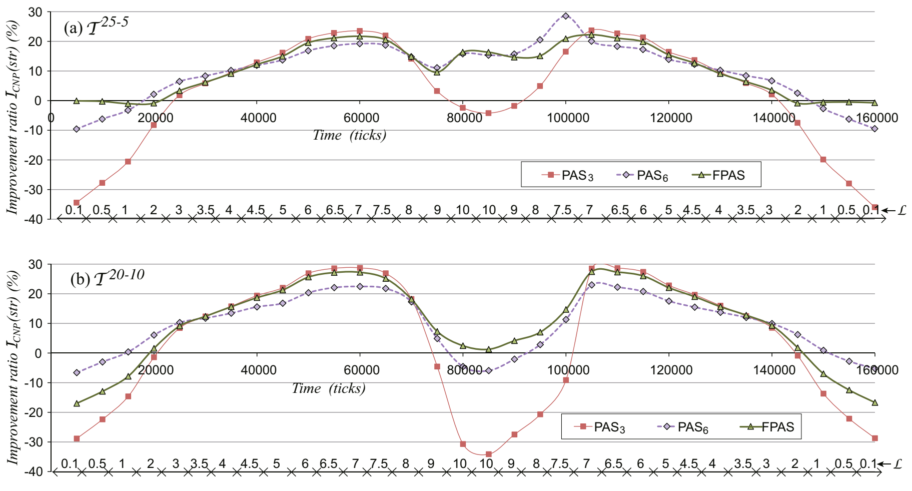

Fig. 1. Ratio of completion times $\mathcal{I}_{CNP}(\text{PAS}_k)$ ($k = 3$ and 6) and $\mathcal{I}_{CNP}(\text{FPAS})$

a *phantom task*, which will be discussed later. Note that the center of the curves in Fig. 1 are shifted slightly to the left because of the effect of the delay in executing tasks queuing during the overload.

3.2 'Flexible' Probabilistic Award

The f-factor of PAS_k should be adaptively controlled according to the system's task loads in order to utilize the full capabilities of a LSMAS from the experimental results in the previous section. However, it is impossible to assess the system's task load, because it is non-local information. Instead, Ref. [14] estimated the task load of the LSMAS from the average queue length of contractors. However, this estimate cannot be easily applied to our case, because if the queue is long but the costs of the queuing tasks are small, the agents cannot conclude whether the system is busy.

Our idea to resolve this issue is to estimate the situations by statistically analyzing the bid values from the local contractors. More precisely, we used the differences between the standard deviations (SDs) of the bid values for different tasks that had different costs. Assume that, for announced task t, manager m received bids whose values were $B_m(t) = \{b_1(t), b_2(t), \dots\}$. Let the SD of $B_m(t)$ be denoted by $SD_m(t)$, and $D_m^{SD}(\mathcal{T})$ be $|SD_m(t_1) - SD_m(t_2)|$ for $\mathcal{T} = \{t_1, t_2\}$. Figure 2 shows how the average values and standard deviations of $D_m^{SD}(\mathcal{T}^{25-5})$ for $\forall m \in \mathcal{M}$ vary every 5000 ticks.

When comparing Figs. 1 and 2, we see that $D_m^{SD}(\mathcal{T}^{25-5})$ can be used as the signal for optimizing the degree of fluctuation; more precisely, the f-factor k can be chosen by using the following strategy,

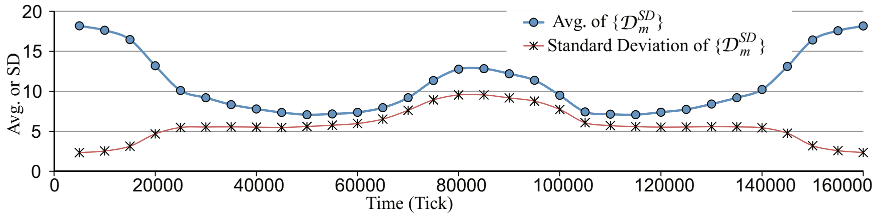

Fig. 2. Average values and SDs of $D_m^{SD}(\mathcal{T}^{25-5})$ over time

$$\begin{aligned} &k = \infty \text{ if } D_m^{SD}(\mathcal{T}) \geq 12.0, \\ &k = 6 \text{ if } 12.0 > D_m^{SD}(\mathcal{T}) \geq 8.8, \text{ and} \qquad \text{(S1)} \\ &k = 3 \text{ if } D_m^{SD}(\mathcal{T}) < 8.8. \end{aligned}$$

This is called the *flexible probabilistic awardee selection* strategy or *FPAS*. The threshold values are determined on the basis of a detailed analysis of each trial from the previous experiments, especially those of $\mathcal{T}^{25-5}$. The aim of this strategy is to combine the best from PAS_∞, PAS_3, and PAS_6.

The results for $\mathcal{I}_{CNP}(\text{FPAS})$ are also plotted in Fig. 1. Figure 1 clearly indicates that FPAS usually provides a better overall performance than the other individual strategies for $\mathcal{T}^{25-5}$ and $\mathcal{T}^{20-10}$. The improvement ratios are particularly large just before the task load reaches the theoretical limit of the LSMAS and right after the contractors surmount the overload caused by the huge number of queued tasks. This is the most important characteristic and will be discussed in Section 5.

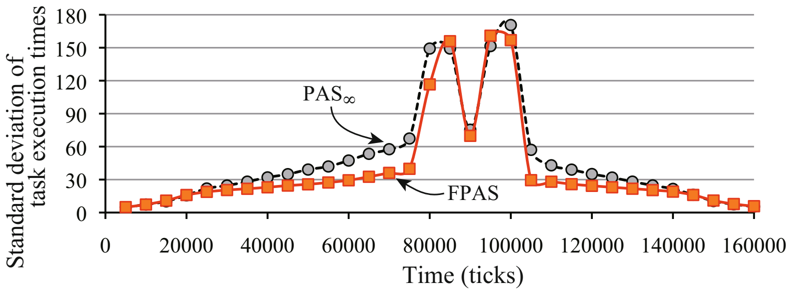

Fig. 3. Standard deviation of completion times under PAS_∞ and FPAS over time

We should also emphasize that FPAS is beneficial to any agent. Figure 3 plots the SDs of the average completion times of individual agents using PAS_∞ and FPAS. The SDs for FPAS are smaller than those for PAS_∞. Therefore, FPAS fairly and impartially performs better for almost all agents.

4 Adaptive Strategy Based on Bid Statistics

4.1 Adaptively Probabilistic Awardee Selection

Since FPAS under strategy (S1) is mainly based on the data for $\mathcal{T}^{25-5}$, it does not necessarily perform well in tasks that have other cost structures. For example, Fig. 1 (b), which indicates the improvement ratio for $\mathcal{T}^{20-10}$, shows that FPAS did not result in a better performance, especially when the task load was low. In this section, we propose a new strategy whereby managers learn how they should determine the f-factor in their environments. The aim of this strategy is to perform in a way that is comparable to that of FPAS for $\mathcal{T}^{25-5}$ and that is generally better than that of PAS$_\infty$ for all tasks.

The algorithm for selecting the f-factor is listed in Fig. 4. First, manager m calculates the SDs of the bid values for each $t_i \in \mathcal{T}$ and the maximum difference between these SDs (denoted by $D_m^{SD}(\mathcal{T})$). It also retains the maximum and minimum values of $D_m^{SD}(\mathcal{T})$ (denoted by *maxSDdiff*, and *minSDdiff*) that have been obtained thus far. It estimates the current task load using *maxSDdiff*, *minSDdiff*, and $D_m^{SD}(\mathcal{T})$. We call this award strategy *adaptively probabilistic awardee selection*, or *APAS*.

Parameter α and variable *minMaxAv* in Fig. 4 are referred to in order to determine whether *maxSDdiff* and *minSDdiff* should be revised. The SDs of $\{D_m^{SD}\}$ in Fig. 2 indicate that the minimum values of $D_m^{SD}(\mathcal{T})$, *maxSDdiff*, and *minSDdiff*, will likely be over-estimated in busy situations because the SD of D_m^{SD} increases. Condition (1) in Fig. 4 estimates this state of overestimation, so we set $\alpha = 1.5$. The constant ε in the figure is used to define the threshold *Th* to switch between award strategies. In our experiments, we chose $\varepsilon = 0.58$ on the basis of the average D_m^{SD} and the SDs of the preliminary experiment shown in Fig. 2. APAS is quite simple in that only PAS$_3$ or PAS$_\infty$ is alternatively selected. Of course, we can extend this to select the appropriate strategy from the set of awarding strategies $\mathcal{S}$, although we set $\mathcal{S}$ ={PAS$_3$, PAS$_\infty$} in our experiment.

Figure 5 plots the improvement ratios for $\mathcal{T}^{25-5}$ and $\mathcal{T}^{20-10}$ over time. $\mathcal{I}_{CNP}$(PAS$_3$) is also plotted because APAS is a mixed strategy involving PAS$_3$ and PAS$_\infty$. Figure 5 indicates that APAS performs as efficiently as FPAS for $\mathcal{T}^{25-5}$ and excellently performs even for $\mathcal{T}^{20-20}$. Note that APAS performs slightly worse than PAS$_\infty$ only when the system is not busy. The learned *Th* might not have been sufficient in this case. Nevertheless, APAS performs excellently in busier situations; it outperforms both PAS$_3$ and PAS$_\infty$, whereas it is the mixed strategy of these two.

4.2 Performance for Different Task Structures and Phantom Task

We also investigated the effect of APAS in tasks with other cost structures. The results are plotted in Fig. 6. These curves indicate that APAS outperforms PAS$_\infty$ for $\mathcal{T}^{22-8}$, $\mathcal{T}^{18-8-4}$, and $\mathcal{T}^{15-8-5-2}$. FPAS does not perform well. We fixed the sum of the costs of these tasks at 3000 to standardize the theoretical upper limit of the task executions by all agents. We used the same changes in the task loads over time because we only wanted to compare their performances under APAS and PAS$_\infty$.

Furthermore, we examined situations in which a number of different tasks occur. For the sake of convenience, let

```
Initialize:
  $\mathit{maxSDdiff} = 0$, $\mathit{minSDdiff} = \mathit{minMaxAv} = \infty$.
for each $\mathcal{T}$
  Manager $m$ announces all tasks $t_1, \ldots, t_l, (\in \mathcal{T})$ to
  the local contractors[2], and $m$ calculates the average
  value, $Av_m(t_i)$, and the SD, $SD_m(t_i)$, of bid values for
  $t_i$.

  /* Then, it calculates some statistical values. */
  $\overline{Av}_m(\mathcal{T}) \overset{\leftarrow}{=} \max_{t_i \in \mathcal{T}} Av_m(t_i)$;
  $\overline{SD}_m(\mathcal{T}) \overset{\leftarrow}{=} \max_{t_i \in \mathcal{T}} SD_m(t_i)$;
  $\underline{SD}_m(\mathcal{T}) \overset{\leftarrow}{=} \min_{t_i \in \mathcal{T}} SD_m(t_i)$;
  $D^{SD}_m(\mathcal{T}) \overset{\leftarrow}{=} \overline{SD}_m(\mathcal{T}) - \underline{SD}_m(\mathcal{T})$;
  $\mathit{minMaxAv} \overset{\leftarrow}{=} \min(\mathit{minMaxAv}, \overline{Av}_m(\mathcal{T}))$;

  /* If the system is not so busy, */
  if $(\mathit{minMaxAv} \times \alpha > \overline{Av}_m(\mathcal{T}))\{$ /* Condition (1) */
    $\mathit{maxSDdiff} \overset{\leftarrow}{=} \max(\mathit{maxSDdiff}, \overline{SD}_m(\mathcal{T}))$;
    $\mathit{minSDdiff} \overset{\leftarrow}{=} \min(\mathit{minSDdiff}, \underline{SD}_m(\mathcal{T}))$;
  }

  /* Defining threshold values: */
  $\mathit{Th} = \varepsilon \cdot \mathit{maxSDdiff} + (\varepsilon - 1) \cdot \mathit{minSDdiff}$;
  /* where $0 < \varepsilon < 1$. */

  /* Then output PAS$_k$ by following the rule: */
  if $(D^{SD}_m(\mathcal{T}) \geq \mathit{Th})$ $k = \infty$;
  else                     $k = 3$;
```

Fig. 4. Outline of APAS strategy

$$C1 = \{\mathcal{T}^{25-5}, \mathcal{T}^{22-8}, \mathcal{T}^{20-10}, \mathcal{T}^{18-12}\}$$
$$C2 = \{\mathcal{T}^{25-3-2}, \mathcal{T}^{20-8-2}, \mathcal{T}^{18-8-4}\}.$$

$\mathcal{T}^{C1}$ (or $\mathcal{T}^{C2}$) in Fig. 6 corresponds to a situation in which the tasks in *C1* (or *C2*) are generated with equal probability. The results in the figure show that APAS performs well in these situations.

If $\mathcal{T}$ consisted of a single subtask or a number of subtasks with almost identical costs, $D^{SD}_m(\mathcal{T})$ could not be calculated or would always be small. For such tasks, we can introduce a *phantom task*, which is announced but is never awarded as a way to estimate the current local workload.

We also investigated the performance of APAS for $\mathcal{T}^{15-15} = \{t_1, t_2\}$ using phantom task t_p, whose cost is 500, and found that APAS outperforms PAS$_k$ and PAS$_\infty$, although the details were omitted here due to space limitations. In this case, managers with $\mathcal{T}^{15-15}$ announce t_1, t_2, and t_p and calculate $D^{SD}_m(\mathcal{T}^{15-15-5})$, but never select an awardee for t_p.

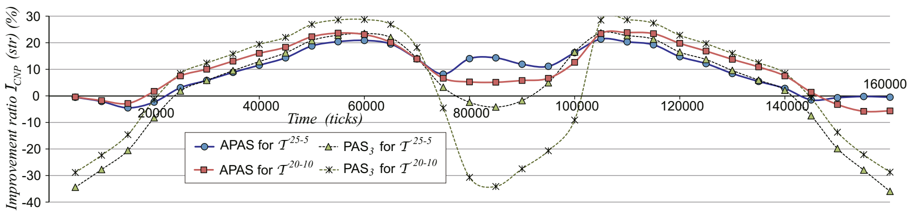

Fig. 5. Improvement ratios of APAS compared with PAS

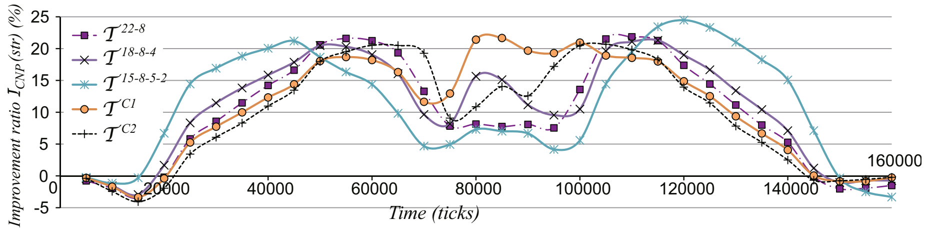

Fig. 6. Improvement ratios of APAS for various tasks

Table 1. Numbers of Dropped and Wasting Tasks

		25-5	18-8-4	15-8-5-2	C1
PAS$_\infty$	dropped	8526.4	4741.3	2538.1	8519.3
	wasting	41517.4	68230	64270.1	40457.5
	total	50043.8	72971.3	66808.2	48976.8
PAS$_3$	dropped	5473.8	1593.6	585.4	5452.1
	wasting	41383.2	66467.7	87807.8	41124.3
	total	46857	68061.3	88393.2	46576.4
APAS	dropped	8542	4655.7	2616.2	9297.9
	wasting	29211.6	47084	62123	29944.3
	total	37753.6	51739.7	64739.2	39242.2

4.3 Analysis of Dropped and Wasting Tasks

In this section we try to analyze why APAS can improve the overall performance. Table 1 lists the numbers of dropped and wasting tasks for different task types in the experiments. We can see that PAS$_3$ clearly reduces the dropped tasks, and that APAS reduces the wasting tasks. APAS also has fewer total numbers of dropped and wasting tasks compared with PAS$_\infty$ and PAS$_3$. Having fewer wasting tasks improves the efficiency of the entire system, because wasting tasks consume more of the contractors' resources.

The main reason for this phenomenon is a small spatial fluctuation in the task load in a busy environment when looking closely at the workloads in each area. As mentioned in the previous section, even in extremely busy cases (near or beyond the theoretical limit), the workload is not spatially uniform although the tasks are randomly assigned to managers. In certain parts of G, agents are overloaded and their queues are full, whereas in other parts the agents have completed some tasks and are somewhat less busy. APAS enables agents to adaptively select PAS_3 or PAS_∞ in accordance with their local conditions. When the workload is beyond the theoretical limit, PAS_∞ (and APAS usually select PAS_∞) results in a large number of dropped and wasting tasks in return for a better performance, $\wp$. In particular, many dropped tasks occur in this case. Unlike PAS_∞, PAS_3 can turn some dropped tasks into wasting ones due to the fluctuation in the award phase. However, wasting tasks are still useless. Thus, PAS_3 is less efficient but has less dropped tasks in extremely busy cases. At the moment of becoming somewhat less busy (APAS usually selects PAS_3), more wasting tasks occur than dropped ones and then PAS_3 can turn some wasting tasks into completed ones. From these results and our analysis, APAS can reduce the number of wasting tasks, comparing with those under PAS_3 and PAS_∞. The analysis in this section implies both a better performance and the presence of less wasting (and dropped) tasks when using the APAS strategy.

4.4 Effect of Maximum Queue Length

In the above experiments, we assumed that the maximum queue length of agent $|Q|$ is 20. This affects the overall performance only during extremely busy periods; the number of tasks in the queue in other situations do not increase. We investigate how the maximum queue length affects the improvement ratios in the paragraphs that follow.

Let $|Q| = \infty$ and use task $\mathcal{T}^{25-5}$ as an example. $\mathcal{I}_{CNP}(\text{PAS}_3)$ improves slightly from its value at $|Q| = 20$, whereas $\mathcal{I}_{CNP}(\text{PAS}_6)$ remains better than $\mathcal{I}_{CNP}(\text{PAS}_3)$. In extremely busy situations, PAS_6 was better than PAS_∞ when $|Q| = 20$, as shown in Fig. 1, but it was worse than PAS_∞ when $|Q| = \infty$. Thus,

$$\mathcal{I}_{CNP}(\text{PAS}_3) < \mathcal{I}_{CNP}(\text{PAS}_6) < 0$$

when $|Q| = \infty$. On the other hand, $\mathcal{I}_{CNP}(\text{APAS})$ is nearly 0 in extremely busy situation, because the queues of the contractors become very long, and none of the managers ever become less busy. Therefore, APAS almost always selects PAS_∞.

5 Discussion

First, we want to point out that the improvement elicited by the proposed strategy was at its largest just before and right after the task load reached the theoretical upper limit. We believe that this feature of our strategy is crucial for real applications. If the task load is low, any task allocation strategy can provide a satisfactory service. However, if it is extremely heavy and over the theoretical limit, no strategy leads to an acceptable performance. In other situations, the system should yield a maximum performance and perform at its fullest potential. Our experimental results revealed that our strategy is excellent in these situations.

When comparing the results in this paper with those in [15], we found that there was significant difference in the mechanism for improving the efficiency. In [15], their method improved the efficiency by avoiding excessive concentration. They dealt with a singleton task, so there was no concept of wasting tasks. In real application systems, however, many wasting tasks that heavily impair the performance are likely to actually occur when they are busy. So, our model is more realistic and from the discussion in the previous section, our proposed task allocation method can improve the performance by reducing the number of dropped and wasting tasks when the agents were extremely busy in the situations mentioned above.

To reduce the number of wasting tasks, it may be possible to propose another and more complex protocol that tracks where the tasks are allocated, monitors them, and if one of the subtask is dropped, stops other subtasks by sending messages to the concerned agents. However, a problem still remains in that the efforts by agents before the arrivals of the messages become meaningless. Another solution is to implement the schedulers that have an accurate view of the other agents and that can compass the other agents' workloads. It is, however, almost impossible to know the other agents' situations because computers may be replaced by others, and tasks are allocated and processed in a distributed way. Avoiding wasting tasks is crucial for the system's efficiency.

Our experiments suggest that autonomous local decisions are more essential for the performance improvement of the entire system. For example, APAS can perform better than PAS_3 and PAS_∞ even though it is a mixture of them. We also examined the performance when $\mathcal{S} = \{\text{PAS}_3, \text{PAS}_6, \text{PAS}_\infty\}$, like for FPAS, but we found no major differences in their performances. The possible reason for these phenomena is that the appropriate and adaptive ratios of PAS_3 and PAS_∞ are more influential. If we closely look at the results from our experiments, the tasks do not arrive at mangers uniformly as discussed in Section 4.3. This small variation is only identified by the individual agents, and only the local decisions can reflect it.

We think that the main reason of the phenomenon shown in this paper is the small communication delay that increase the chances of simultaneous awarding. In our experiments, we assume that managers announce its near agents in terms of communication costs. However, in the actual situations, agent's scope is determined the service-level or upper-level relationships. This makes communication cost larger. So we believe that the phenomena described in this paper are more strongly exposed.

6 Related Works

There is a lot of research currently focused on improving the performance and functionality of CNP. For example, reference [10] extends CNP by introducing *levels of commitment*, i.e., making a commitment breakable with some penalty. References [8,10] try to reduce the number of messages and thereby improve the performance. From the theoretical aspect, there are notable researches that discussed the algorithm of the distributed task allocation in the multi-agent contexts, such as in [7]. All these studies assume, however, that the agents are not very busy and that there are not that many of them, making any interference among them insignificant.

Reference [11] discussed the issue of the eager-bidder problem occurring in a LS-MAS, where a number of tasks are announced concurrently so that a CNP with certain levels of commitment does not work well. These authors propose another CNP extension based on statistical risk management. However, their experiments still used fewer agents than in ours. More importantly, the types of resources and tasks considered are quite different; specifically, the resources are exclusive, such as airplane seats, so they should be selectively allocated. In our case, the resources are divisible, e.g., CPUs or network bandwidth, which can accept any number of tasks simultaneously but with reduced quality.

As a result, many agents with many tasks in our experiments cause a floating uncertainty, which affects the learning, statistical estimation, and rational decisionmaking.

From an organizational perspective, reference [5] proposes an agent organizational network and investigates what features are required to effectively make teams perform a large task. In [1], the issue of an adaptive organizational structure to improve the (overall) efficiency was also addressed. However, these studies do not discuss the allocation strategies for the tasks.

Task allocation to hosts for minimizing the makespan is also one of the central research topics in other domains such as grid computing [3,4]. Programs called mappers or schedulers assign requested tasks to appropriate hosts. However, the costs of the tasks and the capabilities of the hosts are often given so that the mapper accurately knows the processing time of each host. A number of agent-based mapping methods have been proposed; reference [4] uses auction- or contract-based protocols for task allocation. However, these methods are limited to hierarchical mapping structures so they assume geographically close clusters. They also do not take into account the communication delays that may cause uncertainty between the estimated processing status and the actual status.

7 Conclusion

We proposed an optimization method for the probabilistic award strategy in CNP for a large-scale MAS to elicit the potential capabilities of all agents. In a strategy with this optimization, called APAS, a manager agent (a) announces subtasks, (b) statistically analyzes the bids for each of these, (c) estimates the current local task load, and (d) introduces an adaptive degree of fluctuation in the award phase. We experimentally demonstrated that this strategy provides considerably a better performance than the naive CNP.

Although the proposed method performs better than the naive CNP, it still might not be optimal. We must emphasize that the characteristics affecting the overall performance of a LSMAS are complicated and quite different from those of small-scale multiagent systems, so managers should adaptively select the most appropriate strategy. The strategy presented in this paper is simple but can elicit an excellent performance in comparison with the naive CNP. We believe that we can tailor controls to improve the system's performance even further. Moreover, we have to clarify (1) how to vary task types over time and (2) how agent-agent network structures affect the performance under the strategy proposed in this paper. These issues are our future works.

We focused on CNP because it is the well-known and most useful protocol at this time, but CNP is not the only approach to task allocation. Other protocols (with some modification) need to be investigated or a new protocol for busy LSMASs needs to be created. This is also one of our future research topics.

Acknowledgement. This work is, in part, supported by KAKENHI (22300056) and Kayamori Foundation.

References

1. Abdallah, S., Lesser, V.: Multiagent Reinforcement Learning and Self-Organization in a Network of Agents. In: Proceedings of the Sixth International Joint Conference on Autonomous Agents and Multi-Agent Systems, pp. 172–179. IFAAMAS, Honolulu (2007)

2. Buyya, R., Abramson, D., Giddy, J., Stockinger, H.: Economic models for resource management and scheduling in grid computing. Concurrency and Computation: Practice and Experience 14(13-15), 1507–1542 (2003)
3. Casanova, H., Legrand, A., Zagorodnov, D., Berman, F.: Heuristics for Scheduling Parameter Sweep Applications in Grid Environments. In: Proceedings of the 9th Heterogeneous Computing Workshop, pp. 349–363 (2000)
4. Dalheimer, M., Pfreundt, F.-J., Merz, P.: Agent-Based Grid Scheduling with Calana. In: Wyrzykowski, R., Dongarra, J., Meyer, N., Waśniewski, J. (eds.) PPAM 2005. LNCS, vol. 3911, pp. 741–750. Springer, Heidelberg (2006)
5. Gaston, M.E., desJardins, M.: Agent-organized networks for dynamic team formation. In: Proceedings of 4th Int. Joint Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2005), pp. 230–237 (2005)
6. Gu, C., Ishida, T.: Analyzing the Social Behavior of Contract Net Protocol. In: Perram, J., Van de Velde, W. (eds.) MAAMAW 1996. LNCS(LNAI), vol. 1038, pp. 116–127. Springer, Heidelberg (1996)
7. Kraus, S., Plotkin, T.: Algorithms of distributed task allocation for cooperative agents. Theoretical Computer Science 242(1-2), 1–27 (2000)
8. Parunak, H.V.D.: Manufacturing experience with the contract net. In: Huhns, M. (ed.) Distributed Artificial Intelligence, pp. 285–310. Pitman Publishing, Morgan Kaufmann, London, San Mateo (1987)
9. Sandholm, T.: An Implementation of the Contract Net Protocol Based on Marginal Cost Calculations. In: Proceedings of the Eleventh National Conference on Artificial Intelligence, pp. 256–262 (1993)
10. Sandholm, T., Lesser, V.: Issues in automated negotiation and electronic commerce: Extending the contract net framework. In: Lesser, V. (ed.) Proceedings of the First International Conference on Multi-Agent Systems (ICMAS 1995), pp. 328–335. The MIT Press, Cambridge (1995)
11. Schillo, M., Kray, C., Fischer, K.: The Eager Bidder Problem: A Fundamental Problem of DAI and Selected Solutions. In: Proceedings of First International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2002), pp. 599–606 (2002)
12. Smith, R.G.: The Contract Net Protocol: High-Level Communication and Control in a Distributed Problem Solver. IEEE Transactions on Computers C-29(12), 1104–1113 (1980)
13. Sugawara, T., Hirotsu, T., Kurihara, S., Fukuda, K.: Performance Variation Due to Interference Among a Large Number of Self-Interested Agents. In: Proceedings of 2007 IEEE Congress on Evolutionary Computation, pp. 766–773 (2007)
14. Sugawara, T., Hirotsu, T., Kurihara, S., Fukuda, K.: Adaptive Manager-side Control Policy in Contract Net Protocol for Massively Multi-Agent Systems. In: Proceedings of 7th Int. Joint Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2008), pp. 1433–1436. IFMAS (May 2008)
15. Sugawara, T., Hirotsu, T., Kurihara, S., Fukuda, K.: Controling Contract Net Protocol by Local Observation for Large-Scale Multi-Agent Systems. In: Klusch, M., Pěchouček, M., Polleres, A. (eds.) CIA 2008. LNCS (LNAI), vol. 5180, pp. 206–220. Springer, Heidelberg (2008)
16. Weyns, D., Boucké, N., Holvoet, T.: Gradient Field-Based Task Assignment in an AGV Transportation System. In: Proceedings of 5th International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2006), pp. 842–849 (2006)
17. Xu, L., Weigand, H.: The Evolution of the Contract Net Protocol. In: Wang, X.S., Yu, G., Lu, H. (eds.) WAIM 2001. LNCS, vol. 2118, pp. 257–264. Springer, Heidelberg (2001)