Knowledge-Based Systems 115 (2017) 123–132

Contents lists available at ScienceDirect

Knowledge-Based Systems

journal homepage: www.elsevier.com/locate/knosys

A belief propagation-based method for task allocation in open and dynamic cloud environments

Yan Kong[a,*], Minjie Zhang[b], Dayong Ye[b]

[a] Faculty of Computer and Software, Nanjing University of Information, Science, and Technology, Nanjing, 210044, China
[b] School of Computer Science and Software Engineering, University of Wollongong, Wollongong, 2522, Australia

ARTICLE INFO

Article history:
Received 5 November 2015
Revised 12 October 2016
Accepted 15 October 2016
Available online 20 October 2016

Keywords:
Belief propagation
Task allocation
Dynamism and openness

ABSTRACT

We propose a decentralized belief propagation-based method, PD-LBP, for multi-agent task allocation in open and dynamic grid and cloud environments where both the sets of agents and tasks constantly change. PD-LBP aims at accelerating the online response to, improving the resilience from the unpredicted changing in the environments, and reducing the message passing for task allocation. To do this, PD-LBP devises two phases, pruning and decomposition. The pruning phase focuses on reducing the search space through pruning the resource providers, and the decomposition addresses decomposing the network into multiple independent parts where belief propagation can be operated in parallel. Comparison between PD-LBP and two other state-of-the-art methods, Loopy Belief Propagation-based method and Reduced Binary Loopy Belief Propagation based method, is performed. The evaluation results demonstrate the desirable efficiency of PD-LBP from both the shorter problem solving time and smaller communication requirement of task allocation in dynamic environments.

1. Introduction

Task allocation in open and dynamic network environments, especially in cloud computing environments, is an important issue [1–4] because it could be motivated by various contexts, such as supply chain formation [5–8], electronic commerce,[1] RoboCup rescue [9–13], and computation platforms [14,15]. In such contexts, task allocation helps regulate the resource management and utilization in the environments. In specific, task allocation in supply chain formation addresses the problem of determining who (i.e., participants in a supply chain) will exchange which resources with whom [7]. In electronic commerce, task allocation is allocating tasks of consumers to resources provided by resource providers who rent their resources to earn profits. For example, EC2 (Elastic Compute Cloud) [16] is a cloud computing platform of Amazon where Amazon rents its resources to earn profits and resource consumers consume the provided resources to perform their tasks. The tasks of resource consumers need to be allocated to suitable resource providers to meet the goals of both the consumers and the providers. Different from the resource competition in electronic commerce, task allocation in Robocup is always distributed, addressing the coordination among teams of agents that always only have local views in disaster scenarios [10]. GENI [17] is a representative one of open and dynamic computation platforms where people/organizations earn profits through exchanging or renting resources to accomplish tasks.

In cloud environments, many tasks with deadlines require resources from multiple administratively independent resource providers. In addition, the task may consist of more than one subtask which have dependency constraints (i.e., a subtask cannot start to be executed until some other subtask(s) is (are) finished). Considering the dynamism and openness of the environments where both resource providers and consumers can come and leave freely, two main challenges of task allocation arise. First, due to the time constraint, quick online response to and strong resilience from the unpredicted changing in the environment are required [18–20]; the second challenge is how to reduce the communication requirement when changing keeps taking place and thus the allocation has to re-proceed frequently [21–23].

Much study about such a type of task allocation has been done, and many methods have been proposed in the past years, such as multi-resource negotiation-based methods [5,14,15,24–26], double auction-based methods [27,28], combinatorial auction-based methods [6,29–31], belief propagation-based methods [7,32], and evolutionary algorithm-based method [33]. In addition, Jiang et. al. proposed a novel cloud resource auto-scaling scheme at the virtual machine (VM) level for web application providers to achieve both true elasticity and cost-effectiveness in the pay-per-use cloud busi-

* Corresponding author.
E-mail address: kongyan4282@163.com (Y. Kong).
[1] www.ebay.com

http://dx.doi.org/10.1016/j.knosys.2016.10.016
0950-7051/

ness model [34]. In the distributed negotiation mechanism proposed in [15], agents negotiate over both a contract price and a decommitment penalty. The decommitment penalty allows agents to decommit from contracts at a cost, and thus an agent could sign contracts with multiple resource providers to increase the success probability of task allocation. However, it is hard for the agent to decide how many contracts to sign to achieve the optimized solution. To solve the problem that most negotiation strategies cannot assure an equilibrium in real applications, Gatti et. al. proposed an efficient bargaining algorithm [25] to achieve an equilibrium in uncertain environments. Their bargaining algorithm has seen great success but still has some limitations, e.g., the bargaining is carried out in one-sided uncertain environments while real applications are always two-sided uncertain. In [28], Walsh et. al. defined a market protocol based on distributed, myopic, progressive auctions, and non-strategic agent bidding policies to determine prices in supply chain formation. The proposed protocol could always approximate to optimal solutions overall through the locally optimized resource provider selection. In multi-resource negotiation-based methods, the consumer obtains each of the required resources through negotiating with the corresponding providers separately, and this makes the resource obtainment flexible. One drawback of such methods, however, is that the resource consumer risks in not obtaining the complementary resources in follow-up negotiation threads, after successfully obtaining a partial set of the needed resources. Most of the combinatorial auctions require a central controller (i.e., the auctioneer), which not only hinders the scalability of these methods but also is hard to be trusted by other selfish participants. A large number of evaluations have shown that belief propagation-based methods work well [7,32], due not only to the distribution characteristic of belief propagation but also to its resistance characteristic to the dynamism of the environments. Unfortunately, neither of the method in [32] nor [7] pays enough attention to the quick online response to and resilience from the changing of highly dynamic environments, even though belief propagation can work in dynamic environments. A ranking model was proposed in [35] to rank the tasks to be assigned resources through calculating the weights of tasks, and then resources are assigned to the ranked tasks accordingly. This model could work well in quite steady environments, but not desirable for highly dynamic ones because any changing can result in the re-ranking of all of the tasks. The resource allocation method proposed by Piraghaj et al. maps groups of tasks to customized virtual machine types, according to the mapping which is based on the task usage patterns obtained from the analysis of the historical data extracted from utilization traces [36]. However, due to the more and more hierarchies and types of tasks and resources in cloud environments, the task usage patterns need to be updated frequently. This may inhibit the cloud scalability and decrease the time efficiency of task allocation which is important to the task allocation with time constraints. Guo et. al. proposed a workflow task scheduling model, in which the processors are pre-treated by a fuzzy clustering method in order to realize the reasonable partition of processor network, and this can largely reduce the cost in deciding which processor to execute the current task [37].

Against the background, the novelty of this paper is that in order to address the challenging issues of openness and dynamism for task allocation in cloud environments, the proposed belief propagation-based task allocation method (i.e., PD-LBP) devises two phases for the task allocation process: pruning and decomposition. The pruning phase prunes some alternative providers to decrease the number of providers involved in belief propagation. The decomposition phase decomposes the whole network into multiple independent sub-networks where belief propagation can be run at the same time and thus converge quickly.

This paper is organized as follows. The problem definition is formulated in Sections 2, and 3 introduces our belief propagation-based task allocation method. Evaluation is presented in Section 4, and we conclude in Section 5.

2. Problem definition

Assume that the task to be allocated is $\mathcal{T} = \{t_1, t_2, \ldots, t_m\}$ ($t_1, t_2, \ldots, t_m$ are the subtasks of $\mathcal{T}$), only when all the subtasks are successfully allocated, can the allocation of $\mathcal{T}$ be considered to succeed. $\mathcal{T}$ has a deadline dl, but there is not deadline for individual subtasks. Similarly, the consumer has a reserve price p_{res} for $\mathcal{T}$ but does not have reserve price for any individual subtask. The provider agents each of which can execute one subtask of $\mathcal{T}$ are called alternatives, and there may be multiple alternatives for each subtask. Task allocation in this paper is to select a provider for each of the subtasks from the corresponding alternatives to make them collaboratively finish the task, aiming at maximizing some pre-defined objective function. In other words, the solution of our problem is a configuration of providers that can optimize the task allocation according to some predefined criterion. It is notable that maybe some alternatives cannot collaborate with each other due to some reasons (e.g., geography, traffic reasons) which are beyond the study of this paper. Due to the dynamism and openness of the environment, the sets of both alternatives and tasks change constantly.

Formally, we use $\mathcal{S} = \{S_1, S_2, \ldots, S_p\}$ to denote the set of all the configurations that can finish $\mathcal{T}$. $S_k \in \mathcal{S}$ ($1 \le k \le p$) is the kth configuration, and $S_k = \{a_k^1, a_k^2, \ldots, a_k^m\}$ where $a_k^j \in S_k$ ($1 \le j \le m$) is the selected alternative for subtask t_j. If the execution time needed by the selected alternatives to finish the related subtasks are t_k^1, t_k^2,..., t_k^m, and the quotes of the selected alternatives are $q_k^1, q_k^2, \ldots, q_k^m$, respectively, then the goal of our problem is to find the best configuration S^* ($S^* \in \mathcal{S}$) according to:

$$S^* = \arg\max_{\mathcal{S}} \sum u_j$$

$$\text{subject to}$$

$$\begin{cases} \sum_{j=1}^{m} q_*^j \le p_{res} \\ t + \sum_{j=1}^{m} t_*^j \le dl \end{cases} \tag{1}$$

where u_j is the utility that can be gained by the consumer from subtask t_j and will be defined later, and t is the time when this equation is calculated. Eq. 1 represents that the quote summation of the selected alternatives for all the subtasks must not be higher than the reserve price, i.e., p_{res}, of the task. Besides, the execution time summation of all the selected alternative must not be longer than the deadline of the task.

The task allocation problem described above is illustrated in Fig. 1 where the task to be allocated is $\mathcal{T} = \{t_1, t_2, t_3, t_4\}$, and the dependency constraint of the subtasks is $t_1 \rightarrow t_2 \rightarrow t_3 \rightarrow t_4$. The set of alternatives for all the subtasks is $\mathcal{A} = \{a_1, a_2, a_3, a_4, a_5, a_6, a_7\}$, and the respective alternative sets for t_1, t_2, t_3, and t_4 are $A_1 = \{a_1, a_2\}$, $A_2 = \{a_3, a_4\}$, $A_3 = \{a_5, a_6\}$, and $A_4 = \{a_7\}$. As shown in Fig. 1, either a_1 or a_2 can execute t_1. If a_1 is selected to execute t_1, it can pass over the task to either a_3 or a_4 to finish t_2 after finishing t_1. If a_3 is selected to execute t_2, it can pass over the task only to a_5 to finish t_3 after finishing t_2. Regardless of the constraints of both reserve price and deadline, the solutions (i.e., configurations) set is $\mathcal{S} = \{\{a_1, a_3, a_5, a_7\}, \{a_1, a_4, a_6, a_7\}, \{a_2, a_3, a_5, a_7\}, \{a_2, a_4, a_6, a_7\}\}$. The purpose of the task allocation is to find the configuration S^* to maximize some objective func-