Resource Allocation Algorithm Based on Profit Maximization for Crowd Sensing

Kun Gao, Bin Wang, and Xinwu Yu

1Zhejiang Wanli University, Ningbo 315100, China
2Shanghai Key Laboratory of Data Science, Fudan University, Shanghai 200433, China

Correspondence should be addressed to Xinwu Yu; yuxinwu2015@126.com

Received 2 January 2015; Revised 8 March 2015; Accepted 13 March 2015

Academic Editor: Zheng Xu

Copyright © 2015 Kun Gao et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

The key to realizing the crowd sensing network is to overcome the resource restrictions of energy, bandwidth, computing, and so on. First of all, due to the number of users and sensor availability will be dynamic change over time, crowd sensing system is difficult to accurately predict and allocate resource to accomplish a specific task. Secondly, there is a need to consider how to choose an effective subset from a large number of users with different sensing ability, so as to allocate the sensing devices in communication resources under the constraint conditions. This paper proposes a profit maximization algorithm for resource allocation component in crowd sensing environment. The proposed algorithm not only considers the current profit of crowd sensing service request but also considers the long-term expected profits, so as to ensure long-term maximum profit. The objective function is no longer to minimize the completion time but rather to achieve the target profit maximization. The experimental results show that the new algorithm is feasible and superior to the traditional algorithms.

1. Introduction

Crowd sensing is using all kinds of equipment and location technology to statistic and depth analysis the group positions and behavior patterns and then obtain the behavior patterns, organization structure, and dynamical evolution process. It provides support for all kinds of intelligence sensing service in mobile, dynamic, and heterogeneous environments.

The Internet of Things has entered the stage of development; the demand for physical environment thorough sensing becomes more and more strong. In the meanwhile, with the progress of wireless communication and sensor technology, mobile terminal such as smartphones, tablets, wearable devices, and automotive sensing devices integrates more and more sensors and has more and more powerful computing, sensing, storage, and communication ability. With the explosive popularity of the wireless mobile terminal equipment, the Internet of Things will be using the universal mobile devices to provide a larger, more complex, thorough, and comprehensive sensing service. Thus it has already entered a new era of development [1].

In crowd sensing environment, how to effectively allocate the crowd sensing resources is an issue for determining the performance and efficiency of the whole system because the geographical location of the resources is widely distributed, belonging to different autonomous systems and often heterogeneous, dynamic resources. Thus, with the growing popularity of crowd sensing technology, effective resource allocation models and algorithms will be the key to efficient use of these resources. Since crowd sensing technology is moving from theory to practice, so allocation model of economic utility is more meaningful [2].

Crowd sensing resource allocation algorithm inherits the spirit from grid computing, which began in 1960s, that is, multiprocessor parallel unit allocation research. Its mathematical model is an integer programming problem, taking the shortest task makespan as allocation goal, using heuristic algorithm to solve the suboptimal solution. With the popularization of grid computing and crowd sensing, resource allocation algorithm is increasingly considering the economic profit, adding economic metrics, such as price of the unit and the latest completion time in the model.
However, whether or not the parameters of economic profit are contained, the solution of allocation model is the 0-1 variable matrix: if the job $J$ is allocated to the unit $p$, Resource$_{jp} = 1$; otherwise Resource$_{jp} = 0$.

The resources allocated to crowd sensing users are not exclusive to one or more physical unit because of virtualization technology widely used in crowd sensing environment. With the Amazon EC2 platform as an example, the user can pay to buy “one” computing server for several hours to use. In fact, users are not always exclusively using a real physical unit, and there may be one or multiple physical units to provide a virtual unit resource services. The virtualization technology is an important driving force to promote the development of crowd sensing. A single physical machine can be instantiated multiple virtual machines, while the remaining computing resources of multiple physical machines can also be turned into a virtual machine. With the continuous development of virtualization technology, it can be expected that computing resources virtualization consumption will be reduced. From convenient management and safety point of view, the virtual unit will be widely used by crowd sensing service provider.

Thus, crowd sensing resource allocation may not use integer programming model but allocate the computing resources according to the proportion to realize the optimization, and provide these optimized resources to users through virtual technology to provide these resources to the crowd sensing users. This paper presents a crowd sensing resource allocation profit maximization algorithm, describes the resource optimization problem of crowd sensing, and analyses the theoretical significance of the model from the point of view of economics. Finally the authors present the experiment data and analyse the algorithm performance.

2. Related Works

The essence mathematical model of basic allocation research on multiprocessor and parallel units is integer programming problems: allocation objective is the task completion time as short as possible and the objective function is the minimum makespan. This kind of integer programming model almost uses heuristic algorithm to find suboptimal solutions: such as Min-Min, Max-Min, sufferage, and xufferage classic algorithms [2]. With the rapid development of modern optimization algorithms, such as Tabu search, simulated annealing, genetic algorithm/evolutionary algorithm, ant colony optimization algorithm, artificial neural networks algorithm and so on, are also used by advanced distributed computing platform, but it is necessary to improve those algorithms to overcome the existing deficiencies.

Subsequently, there are also some economic profit models added to the economic parameters, such as unit price, operation budget, deadline, and other parameters [3]. This kind of model adds more constraints on the basis of parallel unit allocation model and puts forward higher requirements for the solution algorithm. The solution is still using mostly genetic algorithm and its derivative algorithm.

The above model studied is off-line scheduling problem, which is aware of all job information in advance, and the processing time for certain jobs on certain units needs to be estimated. The method is as follows: to estimate the number of unit instructions that are included in the job [4], according to different computing nodes performance dealing with million instructions per second to estimate the processing time of the work required in the node. Before all jobs entering the sorting system, they need a preprocessing step to estimate the processing time. Another useful research can be seen in [5, 6], and it uses rough set theory to estimate the application execution time, especially suitable for crowd sensing environment.

However, this kind of scheduling models has the following problems: first, the target solution is defined as 0-1 variable, and its essence is the linear integer programming problem. From the theory perspective, linear integer programming problem can be transformed into a linear programming problem, but from computational perspective to achieve the transformation is quite difficult. Secondly, the model almost uses the heuristic algorithm to search the local optimal solution. According to the different input conditions, the convergence rate of the algorithm may be different. It is difficult to guarantee the allocation efficiency.

With the increasing popularity of crowd sensing especially the development of virtualization technology, computing resource allocation is no longer limited to whether one job assigned to a single physical unit. From the profit maximization perspective, crowd sensing environment allocates the overall computational resource. Some computing resources allocated to a job may be surplus resources coming from multiple physical units. Although the physical unit parallel processing will produce more additional overhead, it makes the computing resources management more effective.

In summary, resource allocation problem is equivalent to the virtual unit placement; how to allocate the virtual unit to physical nodes becomes the key factor influencing crowd sensing performance [2, 7–9].

A few applications based on profit maximization can be seen in [10]. The paper [11] presents a method to help content provider distribute the service, but it does not belong to the resource allocation problems. The paper [12] proposed a resource management method whose goal is to make the profit maximization in the model, but eventually the problem is still NP hard. This paper proposes a profit maximization algorithm for crowd censing resource allocation. Compared with the traditional algorithm, the objective function is no longer to minimize completion time but rather to achieve the target profit maximization.

3. Architecture for Resource Allocation Based on Profit Maximization

Based on Xen virtualization technology, this paper constructs crowd sensing resource management platform which can provide virtual unit resources on demand. The platform takes into account two elements: one is virtual unit cost and the other is resource reconfiguration transformation operation cost. Figure 1 shows the architecture, including web application performance model, web application performance
Figure 1: Architecture for crowd sensing resource allocation.

3.1. Web Application Performance Modeling Based on Queuing Theory. Web application performance modeling uses the method [13] based on session description. A session is a collection of multiple transactions; for example, TPC-W contains 14 basic transaction types [14].

Some literature works present the queuing network performance models based on simple feedback circuits; this paper called FBQM model, in which Web application transaction request arrival rate $\alpha_{\text{income}}$. Transaction request is processed by web server, JEE server, and DB server and then gives out the probability $\rho_{\text{end}}$ of the session lifecycle end.

The actual arrival rate of transaction requests entering web application is $\alpha_{\text{income}}'$.

\begin{equation}
\alpha_{\text{income}}' = \alpha_{\text{income}} + (1 - \rho_{\text{end}}) \cdot \alpha_{\text{income}}'
\end{equation}

According to Ritter's law, system resource utilization is $\mu = \alpha \cdot S$, wherein $S$ is system business hours. The resources utilization rate of web server, JEE server, and DB server is expressed as

\begin{equation}
\begin{align*}
\mu_{\text{web}} &= \alpha_{\text{income}}' \cdot S_{\text{web}}, \\
\mu_{\text{jee}} &= \alpha_{\text{income}}' \cdot S_{\text{jee}}, \\
\mu_{\text{db}} &= (1 - \rho_{\text{end}}) \alpha_{\text{income}}' \cdot S_{\text{db}},
\end{align*}
\end{equation}

where $S_{\text{web}}, S_{\text{jee}},$ and $S_{\text{db}}$, respectively, are Web Server, JEE Server, and DB server service time. Service time is the vector $S = S_{\text{time}_1}, S_{\text{time}_2}, \ldots, S_{\text{time}_M}$ composed of the mean service time of each transaction $S_{\text{time}_i}$. So the relationship between web application in load arrival and response time is

\begin{equation}
\begin{align*}
\text{ResT}_{\text{FBQM}} (\alpha_{\text{income}}) &= \frac{S_{\text{web}}}{1 - \rho_{\text{end}} - \alpha_{\text{income}}' S_{\text{web}}} \\
&+ \frac{S_{\text{jee}}}{1 - \rho_{\text{end}} - \alpha_{\text{income}}' S_{\text{jee}}} \\
&+ \frac{\rho_{\text{end}} S_{\text{db}}}{1 - \rho_{\text{end}} - \alpha_{\text{income}}' \rho_{\text{end}} S_{\text{db}}},
\end{align*}
\end{equation}

3.2. Off-Line Test Based Web Application Performance Attenuation Model. This paper constructs web applications performance degradation table by off-line test method. This method consists of web application component Component, virtual...
unit resource type \( Type \), and classic workload \( Workload \), respectively, executing resource reconfiguration operations \( Operation \), where

\[
Component \in \{ \text{Web Server, JEE Server, DB Server} \},
\]

\[
Type \in \{ \text{Li, La, XL} \},
\]

\[
Workload \in \{ \text{Explore, Purchase, Book} \}.
\]

During the experiment, we performed the following operations: increasing VM resource capacity, reducing VM resource capacity, increasing new VM instance, removing the VM instance and VM live migration, recording the response time \( \text{ResT}(\text{Component, Type, Workload, Operation}) \) and response time \( \text{ResT}^p(\text{Component, Type, Workload, Operation}) \) of execution resource reconfiguration during operation \( Operation \), and recording and calculating the response time attenuation value \( \Delta \text{ResT}(\text{Workload}) = \text{ResT}(\text{Component, Type, Workload, Operation}) - \text{ResT}(\text{Component, Type, Workload}) \).

Table 1 shows live migration operation performance degradation table for JEE server.

| Configuration | 1 core | 2 core | 4 core |
|--------------|--------|--------|--------|
| Workload (request/ms) | \( 1 \times 10^5 \) | \( 3 \times 10^5 \) | \( 5 \times 10^5 \) | \( 1 \times 10^5 \) | \( 3 \times 10^5 \) | \( 5 \times 10^5 \) | \( 1 \times 10^5 \) | \( 3 \times 10^5 \) | \( 5 \times 10^5 \) |
| \( \Delta \text{RT} \) (ms) | 244 | 489 | 2091 | 189 | 462 | 1841 | 139 | 439 | 1391 |

4. Algorithm Based on Profit Maximization

As shown in Figure 1, resource allocation engine consists of 4 components: load monitoring, load forecasting, planning adjustment, and resource capacity adjustment. The load monitoring component selects the tool with the properties of crossing platform, high performance, and scalability. The load prediction component selects the open source software based on statistical methods. Capacity planning component is the core of resource adjustment engine and the decision maker of resource reallocation. Resource adjustment component is the specific performer for resources reconfiguration.

By the above analysis, the best option for resource reconfiguration is when the benefit/cost maximization constraint is satisfied. Because the cost is related to resource reconfiguration of transformation period and stable period and the profit is related to resource reconfiguration of stable period, this section first presents calculation method for resource reconfiguration whole period, transformation period, and stable period, then describes the construction method for profit function and cost function, and finally presents the nonlinear algorithm for maximization the profit/cost.

4.1. Calculation for Resource Reconfiguration Total Period, Transformation Period, and Stable Period

4.1.1. Prediction for Resource Reconfiguration Total Period.

Resource reconfiguration of the whole period refers to the web application \( \text{WebApp} \) being detected in the load \( \text{Workload}^\text{time}_{\text{WebApp}} \) at a certain time \( time \) and the application load interval duration \( \text{[Workload}^\text{time}_{\text{WebApp}} - b, \text{Workload}^\text{time}_{\text{WebApp}} + b] \). Load change of web application has the property of interdependency, mutability, randomness, and self-similarity; it makes web applications be regarded as stationary random sequence within the range of load \( \text{[Workload}^\text{time}_{\text{WebApp}} - b, \text{Workload}^\text{time}_{\text{WebApp}} + b] \), while the ARMA model [15] is one of the most commonly used models to describe the stationary random sequence. Therefore, this paper uses ARMA model to predict the total period of resource reconfiguration \( Time_{\text{opt}} \).

Web applications with \( \text{Expectation}_i^j \) represents \( j \)th resource reconfiguration mathematical expectation during total period. \( \text{Expectation}_i^j \) represents actual tested value of resource reconfiguration of total period for the \( j \)th time. According to the actual tested value for \( k \) times, the mathematical expectation value of web application resource reconfiguration of total period based on ARMA model for the \( j + 1 \)th time can be represented as follows:

\[
\text{Expectation}_{i+1}^j = \text{ARMA}(1, K - 1) = \psi \ast \text{Expectation}_i^j + \frac{(1 - \gamma)}{k - 1} \sum_{i=1}^{k-1} \text{Expectation}_{j-i}^m,
\]

where the greater the \( \psi \) value, the greater the impact on the next resource reconfiguration estimated based on current web application resource reconfiguration measured values. The determination of parameter \( \psi \) uses the adaptive filter algorithm [16] which has the advantages of fast convergence characteristics. For the previous \( k \) times, the error between the actual detected value and mathematical expectation value of resource reconfiguration is

\[
\xi_j = \psi \ast |\text{Expectation}_i^j - \text{Expectation}_i^m| + \frac{1 - \alpha}{k - 1} \ast \sum_{i=1}^{k-1} \xi_{j-i},
\]

\[
\psi = 1 - \frac{\xi_j}{\text{max}_{i=1,2...K} \xi_i}.
\]
4.1.2. Calculation for Transformation Period. According to the [17] and the above analysis on VM reconfiguration, we present the calculation formula for transformation period as follows:

\[
\text{Time}_{\text{transformation}} = \text{attenuation} \times \frac{\text{memory}}{\text{bandwidth}} + \frac{\text{disk}}{\text{bandwidth}} + b, \tag{7}
\]

where \(\text{disk}\) represents the size of VM template; \(\text{bandwidth}\) represents the available bandwidth; \(b\) is a constant. For example, to represent VM instant startup time, \(\text{memory}\) represents the size of virtual memory; \(\text{bandwidth}\) represents the available bandwidth; \(\text{attenuation}\) represents an attenuation factor used for describing live migration time impact of source VM memory update rate in the process of live migration.

4.1.3. Calculation for Stable Period. Based on the relationship of resource reconfiguration total period, transformation period, and stable period, the calculation for stable period of web application will be represented as follows:

\[
\text{Time}_{\text{stable}} = \text{Time}_{\text{total}} - \text{Time}_{\text{transformation}}. \tag{8}
\]

4.2. Profit Function. Profit function is used to describe the satisfaction degree with QoS constraints under new configuration in the stable period, so as to effectively prevent application from frequent violating QoS constraint in mutation load. This paper describes the profit as Profit = Satisfactor * Time\text{stable}, where Time\text{stable} presents stable period and Satisfactor presents satisfaction factor.

In [12], the user's satisfaction is proportional to the load. The greater the load is, the higher the user satisfaction for the current web application performance is guaranteed. This paper takes the mathematics expectation for resource reconfiguration of total period as a target to construct piecewise satisfaction function:

\[
\text{Satisfactor} = \left\{ \begin{array}{ll}
\left( \frac{\text{Time}_{\text{stable}}}{\text{Expectation (Time)}} \right)^m \times b, & \text{Time}_{\text{stable}} < \text{Expectation (Time)}, m > 0 \\
 b, & \text{Time}_{\text{stable}} \geq \text{Expectation (Time)}, \end{array} \right.
\tag{9}
\]

where

(i) \(b\) represents the satisfaction degree factor constant,
(ii) \(m\) represents the change trend of satisfaction degree,
(iii) \(\text{Expectation (Time)}\) is specified by users to satisfy the demand of web application QoS.

4.3. Cost Function. According to the above analysis, resource reconfiguration cost \(C_{\text{total}}\) can be divided into transformation cost \(C_{\text{transformation}}\) and stable cost \(C_{\text{stable}}\); that is \(C_{\text{total}} = C_{\text{transformation}} + C_{\text{stable}}\). The former considers the cost during the transformation period; the latter considers the cost during the stable period.

4.3.1. Transformation Period Cost. The transformation period is the operation completion time of resource reconfiguration. The guarantee for the process of the web application of QoS is uncertain, and the transformation cost of resource reconfiguration operation is the main cause of uncertainty. Therefore, \(C_{\text{transformation}} = \text{PenaltyFactor} \times \text{Time}_{\text{transformation}}\), where \(\text{Time}_{\text{transformation}}\) represents the transformation period and \(\text{PenaltyFactor}\) represents the penalty factor. In [18, 19], in the process of resource reconfiguration, if the web application QoS cannot be guaranteed, the penalty factor is proportional to the load; that is, the larger the load is, the greater the penalty factor is. This paper constructs the piecewise penalty function as follows:

\[
\text{PenaltyFactor} = \begin{cases} 
 c, & \text{ResponseTime (transformation)} > \text{QoS}, \\
 k > 0, \text{Workload}_{\text{current}} < 100 & \left( \frac{\text{Workload}_{\text{current}}}{100} \right)^k \times c, \text{ResponseTime (transformation)} > \text{QoS}, \\
 k > 0, \text{Workload}_{\text{current}} \geq 100 & 0, \text{ResponseTime (transformation)} \leq \text{QoS}, 
\end{cases} \tag{10}
\]

where

(i) \(\text{Workload}_{\text{current}}\) represents the current load,
(ii) \(\text{ResponseTime (transformation)}\) represents the response time of resource reconfiguration in current workload \(\text{Workload}_{\text{current}}\),
(iii) \(k\) is specified by users and is used to describe the change trend for penalty factor. This paper sets \(k = 1\). It represents the penalty factor value and load is linear relationship. Namely, the greater the current load is, the greater the penalty factor is,
(iv) \(c\) represents penalty factor constant. As an extreme case of resource reconfiguration allocation, \(\text{Time}_{\text{total}} = \text{Time}_{\text{transformation}}\), namely, the web application in the continued implementation of resource reconfiguration operation.

4.3.2. Stable Period Cost. Stable period refers to the virtual unit resource for web application service and the period for satisfaction of the QoS constraints. So the \(C_{\text{stable}} = f \times |\Delta \text{price}| \times \text{Time}_{\text{stable}}\), where \(\text{Time}_{\text{stable}}\) represents the stable period; \(|\Delta \text{price}|\) represents the change of resource cost after the resource reconfiguration operation; \(f\) represents the cost factor unit resource cost expenditure, specified by the user. According to the above formula, web application execution capacity adjustment operation of the transformation cost is 0; namely, virtual resource reconfiguration operation is satisfying the constraint \(\text{Time}_{\text{total}} = \text{Time}_{\text{stable}}\).

4.4. Planning Algorithm. The programming process is to select the optimal combination of QoS guarantee and
Input: Operation Set; VM Configuration Set; QoS; VM Configuration Price; workload; Expectation (Time).
Output: Operation: Operation ∈ Operation Set; VM: VM ∈ VM Configuration Set.

1. Estimation workload $workload$ and compute out execution time $ResponseTime = ResponseTime_{\text{QoM}}(workload')$.
2. IF $Workload > Workload + b$ //estimation whole cycle is bigger than Expectation (Time)
3. IF $Time_{\text{total}}^p > Expectation (Time)$
4. Virtual Unit = Minimal Price (Virtual Unit Configuration Set);
5. Operation = Minimal Cost (Virtual Unit, Operation Set);
6. ENDIF
7. ELSE //Estimation whole cycle is shorter than Expectation (Time)
8. Maximum = 0;
9. FOR $\sum Time_{\text{whole}} > Expectation (Time)$
10. IF $(Revenue - Cost)/Cost > Maximum$
11. Virtual Unit = Choose (Virtual Unit);
12. Operation = Choose (Operation);
13. Maximum = $(Profit - Cost)/Cost$;
14. ENDIF
15. ENDFOR
16. ENDIF
17. ELSE IF $Workload < Workload - b$ //estimation whole cycle is bigger than Expectation (Time)
18. IF $Time_{\text{total}}^p > Expectation (Time)$
19. Virtual Unit = Maximum Price (Virtual Unit Configuration Set);
20. Operation = Maximum Cost (Virtual Unit, Operation Set);
21. ENDIF
22. ELSE //Estimation whole cycle is less than Expectation (Time)
23. Operation = Minimum Cost (Operation Set);
24. Virtual Unit = Maximum Price (Operation, Virtual Unit Configuration Set);
25. ENDIF
26. ENDIF
27. ELSE
28. Waiting for whole cycle timeout and go to Line (2);
29. ENDElse

Algorithm 1: Programming algorithm based on profit maximization.

We propose the programming algorithm based on profit maximization as shown in Algorithm 1, where

(i) stable period expectation function $Expectation(Time)$ is constant;

(ii) minimal price function represents the minimal resource cost and maximum price function represents the maximum resource cost;

(iii) minimal cost function represents the transformation cost and maximum cost function represents the maximum transformation cost.

The profit function value is the maximum when the predicted stable period $Time_{\text{total}}$ of web application is greater than the mathematics expectation value of users’ resource reconfiguration total period. The best resource allocation reconfiguration will select the minimum value of cost function.

When the predicted stable period is less than the mathematics expectation value of users resource reconfiguration, the process of resource reconfiguration is shown as line 7–15 in Algorithm 1. For example, the current stable period is $Time_{\text{total}}$ and the prediction of the next stable period of web application is $Time_{\text{total}2}$ until reaching $\sum Time_{\text{total}3} > Expectation(Time)$. According to the above description, the upper limit of web application load during the period $Time_{\text{total3}}$ is $Workload_{\text{current+b}}$. The algorithm selects the minimum cost of transformation resource reconfiguration operation. On this basis, the algorithm selects the maximum cost of virtual unit resource satisfied web application to retrieve.

5. Evaluation for Algorithm Performance

In this section, we use the MATLAB programming event simulator to simulate and evaluate the proposed resource
allocation algorithm for crowd sensing based on profit maximization. In this experiment, we divided 3 quality of service (QoS) levels, respectively, corresponding to the number of VM crowd sensing resources allocated, namely, $k_1 = 1$, $k_2 = 2$, and $k_3 = 3$. If there are no other instructions, the total number of crowd sensing resources of VM used in this chapter is 10 ($K = 10$). The mean rate of crowd sensing service request is 7; the rates of crowd sensing request finishing its service and releasing the resource occupied are, respectively, 5 and 10. In order to ensure the long-term income convergence, we set the discount factor 0.1. In addition, in order to ensure the accuracy of the experiment, the experiment time for each test of the performance is set as 1800 s.

In order to further validate the proposed resource allocation algorithm for crowd sensing based on profit maximization, we compared the profit rate and blocking rate of the proposed algorithm with greedy algorithm. As long as there are enough crowd sensing resources, resource allocation algorithm based on greedy algorithm is always set to allocate the biggest crowd sensing resources to its request so as to obtain higher efficiency gains of the system.

Figure 2 compares the difference between profit rate maximization algorithm and greedy algorithm. From the figure we may see that, along with the increase of crowd sensing service request arrival rate, the system effectiveness income also increases along with it, and the algorithm we proposed is superior to the performance of the greedy algorithm. This is because of the fact that, when a crowd sensing service requests the crowd sensing resource, the greedy algorithm is always set to allocate the biggest resource to the request. Therefore, there exists such risk: when a crowd sensing service requests crowd sensing resource, because of crowd sensing resources shortage, greedy allocation algorithm can only refuse the next crowd sensing service request. The proposed algorithm in this paper, for a new request, takes into account not only the current benefit by the current request but also the long-term whole system of expected profit.

As shown in Figure 3, with the increase of crowd sensing service request arrival rate, blocking rate of greedy allocation algorithm for crowd sensing service request is also increased rapidly. At the same time, the blocking rate of benefit maximization algorithm increases relatively slow. Therefore, the result of experiments shows that the performance of profit maximization algorithm for crowd sensing resource algorithm is far better than greedy allocation algorithm.

When a crowd sensing service request arrival rate is low, there are more available crowd sensing resources that can be reserved. In this case, the probability of the crowd sensing service domain receiving a new crowd sensing service request will be higher. For example, when a crowd sensing service request arrival rate is 3, the probability of allocating $k_3$ VM crowd sensing resource to its request can be up to 96%. When crowd sensing requests increase, more crowd sensing resources will be occupied, so the reserved resource for crowd sensing request will be greatly reduced so as to the blocking rate of request increased significantly.

Since the benefit maximization algorithm proposed in this paper needs to consider the long-term profit of the whole system, the new crowd sensing service requesting crowd sensing resources will be more cautious. As shown in Figure 4(a), with the increasing of crowd sensing service request, the probability of the new crowd sensing service requests being adopted (decision $a = 3$) is reducing, and the probability of decision $a = 1$ and decision $a = 2$ is increased accordingly. Figure 4(b) indicates the relationship between the probability of all kinds of decisions and total resources of crowd sensing. With the increasing crowd sensing resources VM, the probability of new request obtaining crowd sensing resource is also increasing. As shown in Figure 4(b), when the number of VM increases from 1 to 10, the probability of decision $a = 3$ increases from 0% to 95%.

6. Conclusions

The crowd sensing resource allocation algorithm based on profit maximization is proposed in this paper. Different from the traditional method, this paper uses the thinking of network profit maximization and realizes physical crowd sensing resource allocation in the higher lever: the algorithm
takes the maximum profit as the allocation objective. One or more physical resources are allocated to one or more crowd sensing jobs. Our solution not only considers the current profit of crowd sensing service request but also considers the long-term expected profits, so as to ensure long-term maximum profit. We conduct experiments to validate our hypotheses; experimental results show that this algorithm performance is better than other traditional methods.

The future work will explore the impact of different utility functions on the virtual unit resource allocation efficiency. The crowd sensing resources from the simple measurement is extended to multiple resources common constraints; that is, the description for crowd sensing resource will be developed from one-dimensional to multidimensional description.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

Acknowledgments

This work was supported (in part) by Shanghai Science and Technology Development Funds (13dz2260200, 13511504300), Projects in Science and Technique of Ningbo Municipal (2012B82003), and Key Research Center of Philosophy and Social Science of Zhejiang Province-Modern Port Service Industry and Creative Culture Research Center.

References

[1] M. Armbrust, A. Fox, R. Griffith et al., “A view of cloud computing,” Communications of the ACM, vol. 53, no. 4, pp. 50–58, 2010.
[2] O. H. Ibarra and C. E. Kim, “Heuristic algorithms for scheduling independent tasks on nonidentical processors,” Journal of the Association for Computing Machinery, vol. 24, no. 2, pp. 280–289, 1977.
[3] P.-Y. Chen and S.-Y. Wu, “The impact and implications of on-demand services on market structure,” Information Systems Research, vol. 24, no. 3, pp. 750–767, 2013.
[4] J.-S. Chang and R.-S. Chang, “A performance estimation model for high-performance computing on clouds,” in Proceedings of the 4th IEEE International Conference on Cloud Computing Technology and Science (CloudCom ‘12), pp. 275–280, IEEE, Taipei, Taiwan, December 2012.
[5] K. Gao, K. Chen, M. Liu, and J. Chen, “Rough set based data mining tasks scheduling on knowledge grid,” in Advances in Web Intelligence, vol. 3528 of Lecture Notes in Computer Science, pp. 150–155, Springer, Berlin, Germany, 2005.
[6] K. Gao, Y. Ji, M. Liu, and J. Chen, “Rough set based computation times estimation on knowledge grid,” in Advances in Grid Computing—EGC 2005, vol. 3470 of Lecture Notes in Computer Science, pp. 557–566, 2005.
[7] R. Özda˘g and A. Karcı, “Sensor node deployment based on electromagnetism-like algorithm in mobile wireless sensor networks,” International Journal of Distributed Sensor Networks, vol. 2015, Article ID 507967, 15 pages, 2015.
[8] M. N. Hindia, A. W. Reza, and K. A. Noordin, “A novel scheduling algorithm based on game theory and multicriteria decision making in LTE network,” International Journal of Distributed Sensor Networks, vol. 2015, Article ID 604752, 8 pages, 2015.
[9] S. M. Jameii, K. Faez, and M. Dehghan, “Multiobjective optimization for topology and coverage control in wireless sensor networks,” International Journal of Distributed Sensor Networks, vol. 2015, Article ID 363815, 11 pages, 2015.
[10] M. D. de Assunção, A. Di Costanzo, and R. Buyya, “Evaluating the cost-benefit of using cloud computing to extend the capacity of clusters,” in Proceedings of the 18th ACM International Symposium on High Performance Distributed Computing (HPDC ’09), pp. 141–150, Garching, Germany, June 2009.
[11] P. Hande, M. Chiang, R. Calderbank, and S. Rangan, “Network pricing and rate allocation with content provider participation,” in Proceedings of the 28th Conference on Computer Communications (INFOCOM ’09), pp. 990–998, IEEE, April 2009.
[12] Q. Zhu and G. Agrawal, “Resource provisioning with budget constraints for adaptive applications in cloud environments,” in
Proceedings of the 19th ACM International Symposium on High Performance Distributed Computing (HPDC ’10), pp. 304–307, ACM, Chicago, Ill, USA, June 2010.

[13] D. Krishnamurthy, J. A. Rolia, and S. Majumdar, ”A synthetic workload generation technique for stress testing session-based systems,” IEEE Transactions on Software Engineering, vol. 32, no. 11, pp. 868–882, 2006.

[14] T. T. P. P. Council, TPC-W benchmark, 2005, http://www.tpc.org/tpcw/.

[15] I. Foster, Z. Yong, I. Raicu, and L. Shiyong, ”Crowd sensing and grid computing 360-degree compared,” in Proceedings of the Grid Computing Environments Workshop (GCE ’08), pp. 1–10, 2008.

[16] R. Buyya, C. S. Yeo, S. Venugopal, J. Broberg, and I. Brandic, ”Crowd sensing and emerging IT platforms: vision, hype, and reality for delivering computing as the 5th utility,” Future Generation Computer Systems, vol. 25, pp. 599–616, 2009.

[17] X. Team, Linux foundation, 2013, http://www.xenproject.org/.

[18] D. Nurmi, R. Wolski, C. Grzegorczyk et al., ”The eucalyptus open-source cloud-computing system,” in Proceedings of the 9th IEEE/ACM International Symposium on Cluster Computing and the Grid (CCGRID ’09), pp. 124–131, May 2009.

[19] L. Youseff, M. Butrico, and D. da Silva, ”Toward a unified ontology of cloud computing,” in Proceedings of the Grid Computing Environments Workshop (GCE ’08), pp. 1–10, Austin, Tex, USA, November 2008.