Global Journal of Computer Science and Technology: E
Network, Web & Security
Volume 18 Issue 1 Version 1.0 Year 2018
Type: Double Blind Peer Reviewed International Research Journal
Publisher: Global Journals
Online ISSN: 0975-4172 & Print ISSN: 0975-4350

An Agent-based Grouping Strategy for Federated Grid Computing

By Aminul Haque & Md. Tanvir Rahman
Daffodil International University

Abstract- Characterizing users based on their requirements and forming groups among providers accordingly to deliver them the stronger quality of service is a challenge for federated grid community. Federated grid computing allows providers to behave cooperatively to ensure required utility by users. Grouping grid providers under such an environment thus enhance the possibility of more jobs executed whereas a single provider or organization might not be able to do the same. In this paper, we propose an agent-based iterative Contract Net Protocol which supports in building federated grid via negotiating distributed providers. The main focus of this paper is to minimize the number of iterations using a grouping mechanism. Minimizing the number of iterations would produce less communication overhead which results in the minimum queue waiting time for users to publish their jobs. Simulation results further ensure the feasibility of our approach in terms of profit and resource utilization compared to that of the traditional non-grouped market.

Keywords: grid computing, agent technology, economic model, group formation.

GJCST-E Classification: C.1.4, C.1.m

ANAGENTBASEDGROUPINGSTRATEGYFORFEDERATEDGRIDCOMPUTING

An Agent-based Grouping Strategy for Federated Grid Computing

Aminul Haque [α] & Md. Tanvir Rahman [σ]

Abstract- Characterizing users based on their requirements and forming groups among providers accordingly to deliver them the stronger quality of service is a challenge for federated grid community. Federated grid computing allows providers to behave cooperatively to ensure required utility by users. Grouping grid providers under such an environment thus enhance the possibility of more jobs executed whereas a single provider or organization might not be able to do the same. In this paper, we propose an agent-based iterative Contract Net Protocol which supports in building federated grid via negotiating distributed providers. The main focus of this paper is to minimize the number of iterations using a grouping mechanism. Minimizing the number of iterations would produce less communication overhead which results in the minimum queue waiting time for users to publish their jobs. Simulation results further ensure the feasibility of our approach in terms of profit and resource utilization compared to that of the traditional non-grouped market.

Keywords: grid computing, agent technology, economic model, group formation.

I. Introduction

Grid computing is a special kind of network that connects distributed computer resources (such as clusters, supercomputers, and datasets) to provide stronger computation power as well as data warehouse over the Internet in order to solve computationally intensive problems (such as drug design, investigate material properties and weather forecasting). These resources are typically owned by different owners and driven by different rules and policies. Economic models such as Contract Net Protocol (CNP) [1], Double Auction [2], and Commodity market [3] are found suitable in harnessing these distributed resources over different ownership. Recently federated grid has emerged as a new approach that supports coordination of resources through grouping mechanism in order to optimize users quality of service (QoS) (i.e. resource availability, reliability, performance etc.) [4], [5]. However, autonomous coordination of distributed resources is essential to achieve perceived utility by users. However, extreme heterogeneity, dynamic nature, and different ownership of these resources impose challenges to do that.

Agent technology in computer science is well known due to their autonomous actions in making decisions and capability of interacting (such as cooperate, coordinate and negotiate) with other agents like other social beings. Due to the development and application of agent technologies, a surge of interest has been focused on agent-oriented methodologies and modeling techniques. The reason for including agents in grid computing is that grid computing and agent systems have similar objectives. Both aim to achieve *"large-scale open distributed systems, capable to effectively and dynamically deploy and redeploy resources as required, to solve computationally complex problems"* [6]. Similarly, agents representing different grid providers can interact with each other and form groups or teams in order to meet their respective goals (e.g. meeting users QoS, earning profit etc.). However, differentiating among QoS (i.e. typically represented by user's preference values on QoS), and forming groups accordingly to meet their demands are open issues in this field.

In this paper, we study how to characterize different users in terms of their varied utility demands and budget constraints. Perceive different utility is important in order to deliver stronger QoS. In addition, we study how to map appropriate groups with received users to enhance system efficiency (e.g. better profit and resource utilization). We propose an agent-based iterated CNP (*i*CNP) where agents representing users and providers are autonomous to interacting each other and both appear with their respective requests and offers. *i*CNP allows multi-round iterative bidding. In general, under such an economic model, a manager (user) issues/publish the initial call for proposal *(cfp)*/ resource demand. The contractors (providers) then evaluate the proposal and propose their bids. The manager then accepts one or more of the bids or may iterate the process by issuing a revised *cfp*. However, escaping from one round and then waiting for the next round to resubmit the request may cause a long queue waiting time in a large-scale framework which is typically comprised of thousands of users and providers such as grid. Hence, in this paper, we further focus on how to treat a user in a better way from the first round by incorporating grouping mechanism and thus to minimize the number of iterations. Minimize the number of iterations prevents users from the uncertainty to best treat their values and reduces communication delay dramatically since negotiation with users as well as

Author α σ: Department of Computer Science and Engineering, Daffodil International University, Dhaka, Bangladesh.
e-mails: aminul.cse@daffodilvarsity.edu.bd, sajal.it.ju@gmail.com

among providers requires huge communication process. We group providers based on their resource availability so that they can treat the corresponding users the best it is possible by them.

II. Related Work

Chao et al. [4] proposed for grouping grid nodes in terms of nodes' own desires to optimize resource allocation problem. They grouped-up according to compatible users and providers based on catallaxy-based market though how they defined different criteria to do this is not clear. In addition, they conducted the simulation with 100 agents (50 users and 50 providers) but what impact it would have if they conducted the simulation with varying the number of users and providers is not taken into account. We conduct our simulation with thousands of users and a varying number of users and providers.

CNP has been used in a cluster environment to optimize utility for users [1]. They have compared the performance of CNP and traditional Round Robin Protocol (RRP) in terms of different job arrival rates and show the advancement of CNP over RRP in terms of utility and computational cost. They have conducted their simulation by incorporating two scenarios; firstly, the scenario that accommodates all the mono-thematic applications (similar kind of applications) and secondly, which accommodates heterogeneous tasks. However, in our work, we focus on *i*CNP and characterize users in terms of their preferences, such that they can be efficiently evaluated.

Goswami adopts [7] CNP to deal with resource heterogeneity and proposes two resource selection policies. One is K-time optimization policy, in which users are sorted in ascending order in terms of their proposed deadlines of finishing their jobs. Another one is, K-cost optimization policy, in which users are sorted in terms of their budgets, they are willing to pay. The value of K refers to whether to switch from K-time to K-cost or not and vice versa. The drawback of this system is, though the failed users have the chance to re-announce/revise their *cfp,* they still resubmit their *cfp* without changing anything (e.g. increase budget or reduce QoS). Hence, the probability of accepting revised *cfp* would be decreasing and produce high communication overhead. We change *cfp* over iterations which maximize to successful SLA establishment in each round.

An agent-based Content Distributed Grid (CDG) is proposed to form VOs [5]. The concept of CDG is borrowed from Content Distributed Network (CDN) in where all the servers are co-operative to each other and belong to the same organization. The CDG is different to the point that all the resources under grid computing are competitive and belong to different owners. Hence, they propose an economic approach to motivate grid providers such that they can be cooperative in contributing their resources in order to maximize utility for users. The failed users can re-negotiate with providers based on their revised *cfp* and this can happen over a certain number of iterations. However, under which condition how many iterations it may have to allow users re-negotiate is not discussed. If the number of iterations is very low, some users might lose their chance to re-negotiate or if it is very high, it might produce high communication delay. We allow the auction to be continued until there are at least one potential user and one potential provider in the market which guarantees all the users making deal with providers. In addition, our grouping strategy helped to minimize iterations while ensuring best treat the users.

Ranjan et al. [8] proposed CNP-based negotiation for meta-scheduling resources in federated grids. Their proposed SLA-based approach is designed to satisfy users by maintaining their job deadlines as well as allows providers to control over their resources. Users in their system are allowed to iterate the negotiation process if they fail in a particular round. However, how users revise their *cfps* is not discussed. Hence, accepting revised *cfp* becomes harder, since the providers keep the resource cost constant throughout an experiment. There could be another way of revising *cfp*, that is, minimizing resource requirements rather than maximizing budget but it may not be applicable to a group of users who define a resource as their optimization. Our group-based optimization strategy minimized the occurring of revising *cfp* and thus decreased the chance of generating such unexpected scenarios.

An enhanced ant colony algorithm combining the technique of Ant Colony System and Mix Ant System for job scheduling for grid computing is proposed in [9]. The proposed algorithm also contains the concept of agent for the purpose of updating the grid resource table.

Laizhi Wei et al. [10] proposed an improved ant algorithm for Grid task scheduling strategy with a new sort of pheromone and node distribution selection rule. The proposed algorithm can measure the performance of resources and tag on it. By dealing with the unsuccessful situations of task scheduling, unnecessary overhead of the system is reduced that results in shortening the total time requirement of a complete task. Sonal Yadav et al. [11] proposed a cost based job grouping and scheduling algorithm that will be beneficial to both user and resource broker. Before allocating resources the algorithms groups the users job which results improved communication to computation ratio and utilization of available resources.

The authors of [12] have proposed a grouping based job scheduling algorithm that uses priority queue and hybrid algorithm to maximize the resource utilization and minimize processing time of the jobs. By considering static restrictions and dynamic parameters of jobs and machines the algorithm selects the best suitable machine for user's job.

III. System Model

We model our framework (Figure 1) with three types of agents, which are user-agent, provider-agent and traffic-agent. Each type of agents is programmed in a way so that it can try its maximum to reach an agreement (Service Level Agreement) while interacting with other agents. We use *i*CNP as the interaction protocol. Since it is iterative, agents in our model can optimize their goals through renegotiation. Details on *i*CNP are described in the following section. The three types of agents in our system try to reach the following individual goal:

Type 1 (User-agents): Try to optimize the preferred values (e.g. storage, budget) as defined by the corresponding users,

*Type 2 (Provider-agent):*Aims to receive more users so that they can maximize their profit,

Type 3 (Traffic-agent): Is designed to receive and evaluate users' request and finally switch to the appropriate groups.

Therefore, it is clear that each type of agents has its own task to accomplish. However, in this work, we only consider users requests (jobs) as tasks. Such a task-oriented domain can be defined as,

P = Provider, *cfp* = call for proposal, OP = Optimization

-·-·-·→ SLA not established and send back to revise *cfp*

←- - - -→ Migration due to unavailable resources

←- -——→ Accessing Virtual Organization (VO)

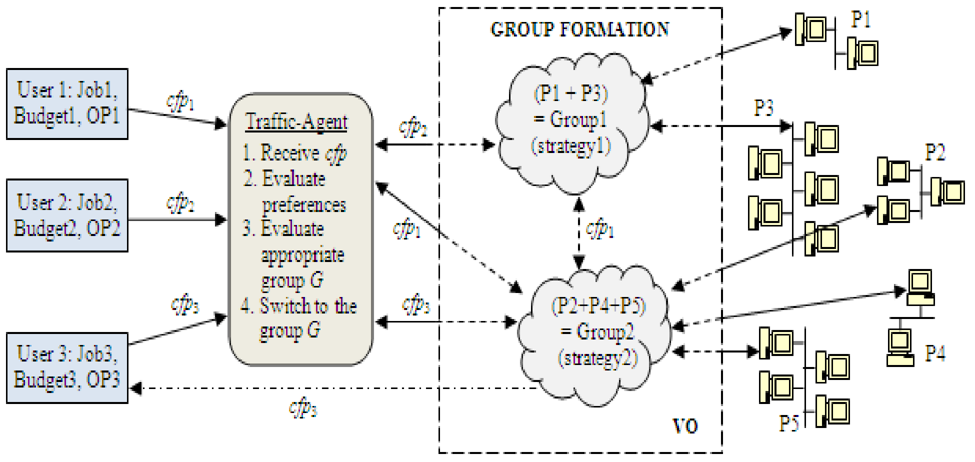

User 1: Job1, Budget1, OP1
User 2: Job2, Budget2, OP2
User 3: Job3, Budget3, OP3
cfp1
cfp2
cfp3
Traffic-Agent
1. Receive cfp
2. Evaluate preferences
3. Evaluate appropriate group G
4. Switch to the group G
GROUP FORMATION
(P1 + P3) = Group1 (strategy1)
(P2+P4+P5) = Group2 (strategy2)
VO
P1
P2
P3
P4
P5

Fig. 1: An overview of our proposed group-based federated grid

$$<T, A, C>$$

Where *T* refers to the set of all tasks. Here the number of tasks (subsets) is equal to the number of users.

$A=(\{A_1,\ldots,A_u\},\ \{A_1,\ldots,A_p\},\ A_t)$ is the set of participating agents. Au, Ap, and At refer the user-agent, provider-agent and traffic agent. Again, $A_u \subset A$, $A_p \subset A$, and *t* are always 1.

C refers to the cost of executing a particular task.

Now, we describe different types of agents in terms of their activity.

a) User-Agent

In our model, a user-agent is represented by A_u where $|u| > 0$ and can be any arbitrary number. Each A_u is set by a few resource requirements, budget, deadline, and preferred optimization. This is called "call for proposal (*cfp*)" and given by,

$$cfp_u = \{R, B, D, Pref\}$$

However, in this work, we set an A_u only with resource requirements (storage and processors) *R*, budget *B* and preferred value *Pref*. The preferred value can be either in resources or budget. The role of an A_u is to try optimizing its *Pref* while considering other associated constraints. Hence, an A_u can easily be characterized based on its cfp_u. User-characterization is mandatory to deliver stronger service. In our work, we consider four different ways to differentiate users.

Firstly, the users set processing as their preferences. In addition, respective budgets are relaxed, that is, willingly to pay whatever price providers impose on *cfp.* An A_u can perceive its preference value automatically based on resource requirements. Hence, a cfp_u can be re-written as,

$$cfp_u = \{CPUs \geq predefined\ CPUs\ and\ storage < predefined\ storage,\ B(relaxed),\ D,\ CPU\}$$

Secondly, the users set storage as their preferences. In this case, the budgets are relaxed as well. Similarly, an Au can perceive its optimization entity by using the following cfp_u,

$$cfp_u = \{CPUs < predefined\ CPUs\ and\ storage \geq predefined\ storage,\ B(relaxed),\ D,\ storage\}$$

Thirdly, the users come with cost optimization. This is recognized by the following cfp_u,

$$cfp_u = \{CPUs < predefined\ CPUs\ and\ storage < predefined\ storage,\ B,\ D,\ cost\}$$

Finally, the users set combined optimization as their preferences, which means, they want more resources with lower costs. This type of cfp_u can be defined as,

$$cfp_u = \{CPUs \geq predefined\ CPUs\ and\ storage \geq predefined\ storage,\ B,\ D,\ combined\}$$

Characterizing users in terms of their *cfp* would help grid providers to treat them better than might otherwise be expected. However, this characterization can be extended in a few more ways (such as both of resources optimization with relaxed budget). We have left either ways for our future work. We use different ranges for different resource requirements in order to dynamically set thousands of users. The average values of the ranges are considered as predefined resources. Please note that if an Au fails in a particular iteration, it revises its cfp_u for the following iteration by increasing its budget until the budget reaches its maximum value. The next step of an Au is interacting with traffic-agent.

b) Traffic-Agent

A traffic-agent is represented by A_t. Only one A_t is designed to deal with all A_u. At first, it receives cfp_u from an A_u. Then it evaluates the cfp_u and detects the preference value. After that, it evaluates appropriate group in order to better serve the user. Details on grouping are described in Section 2.3. Finally, it switches the A_u to the appropriate group which is designed to treat the user best. Hence, it is crucial for an A_t to determine appropriate groups of users otherwise it may cause some extra iterations which ultimately increases communication overhead.

c) Provider-Agent

A provider-agent is represented by A_p. Where, $|p| > 0$ and can be any arbitrary number. An A_p is designed with resource availability and prices for the unit amount of resource consumption. Details on pricing are explained in the Economic model section. In this paper, we focus on the provider side. Therefore, we concentrate on provider strategy rather than user strategy. Although providers in grid computing are known to be self-interested, they can save costs by coordinating their activities among themselves. In a multi-agent paradigm, such a grouping activity is known as characteristic function game *(CFG)*. In such games, the value of each group G is given by a characteristic function v_G. Hence, providers can be grouped in several possible ways based on v_G. This is called group structure *(GS)*. So, for any group, we can say $G \in GS$.

We assume that providers are aware of the demand curve (a market trend on resource demand) and all available cases users can be characterized on. Based on this, all the providers under our federated grid automatically form into four different groups based on their resource availability. Groups are presented here in terms of their set of characteristic functions:

$$v_G{=}1{:}\ \{processors \geq predefined\ processors\ and\ disk\text{-}space < predefined\ disk\text{-}space\}$$

$$v_G{=}2{:}\ \{processors < predefined\ processors\ and\ disk\text{-}space \geq predefined\ disk\text{-}space\}$$

$$v_G{=}3{:}\ \{processors < predefined\ processors\ and\ disk\text{-}space < predefined\ disk\text{-}space\}$$

$$v_G{=}4{:}\ \{processors \geq predefined\ processors\ and\ disk\text{-}space \geq predefined\ disk\text{-}space\}$$

Here, each *G* is formed to better treat its corresponding cfp_u. For example, *G1* (group1) is designed with that provider who are able to supply more processing power and thus to deliver stronger quality to type1 cfp_u. However, grouping in our model occurs prior to serving a particular cfp_u rather than after receiving the cfp_u. The reason for this is to prevent users from waiting while forming groups over distributed domains. In addition, this would increase the probability of receiving more users by a particular *G*.

For each *G*, there is a group correspondent (typically the first provider), who initiates dealing with a cfp_u and negotiates with other providers within that *G* if requires. However, *G* formation in any *CFG* needs to satisfy the following facts;

Group structure generation: Formation of groups by the agents such that agents within each group coordinate their activities, but agents do not coordinate between groups. Typically, this generation occurs super-additively, which is, any *G* of agents is best off by merging into one. This can be explained in terms of the utility function,

$$UtilityA1 \cup A2 \geq UtilityA1 + UtilityA2$$

For all disjoint agents $A1, A2 \subseteq A$. The utility function of an agent A for a deal δ in order to accomplish a task T can be defined as,

$$\text{UtilityA}\ (\delta) = \text{C (TA)} - \text{costA}\ (\delta)$$

Where *C (TA)* refers to the cost originally assigned to the agent *A* to accomplish the task *T* and *costA (δ)* is the cost spends to process the deal. The agents presented here are all A_p and the utility function is restricted to *G* generation.

However, in many cases, *G* formation may not be super-additive since there are some costs (such as communication cost, security cost) to *G* formation process itself. Therefore, under costly computation, component grouping within a single provider or organization may be better off by not forming a composite grouping with different providers. However, in case of grid computing, most cases, a single provider is not able to meet large-scale resource requirements. Again, due to large-scale resources trading, associated costs would be less in most cases.

Optimization of the group: Here, a particular *G's* objective is to maximize monetary value, that is, to maximize the utility value in combination. This can be achieved by increasing the money received from users or decreasing the cost of using resources.

Payoff division: It divides the generated solution among the provider-agents of a particular *G.* The division should be in a fair and stable way so that the agents are motivated to stay with the *G* rather than move out of it. In our model, the solution of a particular *G* is divided into the providers according to the number of resources they have shared.

i. *Group migration*

Though each cfp_u is supposed to receive in its respective *G* which is appropriate to treat that cfp_u, the corresponding A_u can still be migrated to another *G*, if the designed *G* becomes unable to deliver the resource demands. Under a federated grid, this migration policy would increase the system efficiency, since it tries its maximum to treat a user well. However, there are some restrictions to migrate a cfp_u from one *G* to another. As aforementioned, we are focusing on provider side; hence, we have used some strategies over migration so that the system can produce a better payoff.

Please note that all strategies of migrating a cfp_u from one *G* to another is subject to unavailable resources with the *G* it is migrating from.

Strategy 1: From providers' point of view, *G1* and *G2* users receive priority than others, since these users come with a relaxed budget. Hence, *G1* and *G2* users are allowed migrate to *G3* and *G4* at any iteration while *i*CNP.

Strategy 2: G3 and *G4* users are allowed migrate to *G1* and *G2* at all iterations except the first since the budgets of *G3* and *G4* users are not relaxed and thus get less priority by providers. This is done such that in the first iteration providers can receive more users with relaxed budget and thus to maximize profit.

However, our model supports to define a migration policy by a particular group *G* in either way about when to migrate and migrate to which.

IV. Implementation

We established a simulation environment and implemented the proposed model using a cross-platform multi-agent programmable modeling environment known as Netlogo [13], [14]. We choose Netlogo because:

- Netlogo is a FIFA (Foundation for Intelligent Physical Agent) conformant platform [15].
- It has extensive built-in models to deal with multi-agents.
- It can work as a 'simulated parallel' environment [13].
- It is platform (Mac, Windows, and Linux) independent [14].

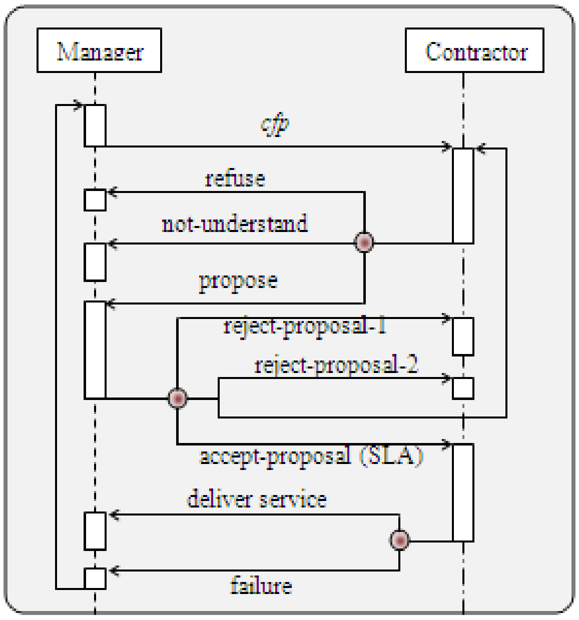


Fig. 2: FIFA Iterated Contract Net Protocol

a) *Economic model*

We implemented FIFA conformant *i*CNP (Iterated Contract Net Protocol) which is an extension of the basic CNP, but it differs by allowing multi-round iterative bidding [16]. *i*CNP supports optimizing a particular user's request by negotiating distributed providers. In such a model (Figure 2) a user is called a manager who issues an initial *cfp*. Providers are known as contractors, who then response with their bids and the manager may then accept one or more of the bids, rejecting the others, or may iterate the process by issuing a revised *cfp*. However, the number of iterations can be based on a time period or potential users (who can still maximize their budgets) and providers (who can still serve at least one standard user). We consider that the iteration continues until there are a potential user and a potential group. Though using our approach seems to increase number of iterations and thus communication overhead, the approach is more consistent in grid perspective and using our grouping strategy, number of iterations could be decreased. In our case, a group G receives a *cfp* via the traffic-agent *(A_t)* with the guarantee that the group *G* receives the correct *cfp*. However, evaluating a particular *cfp* would be different, if a particular group *G* receives the *cfp* from another group *G* (migration) rather than the traffic-agent *(A_t)*.

b) *Bidding policy*

In our system, each user comes with a band of budgets, which are minimum budget and maximum budget the user is willing to pay. Typically, an A_u (user-agent) corresponding to a user starts bidding from its minimum budget to the maximum budget over iterations if it can establish the SLA. The bidding in our mechanism follows linear increment. Though we are not focusing on user-strategy, we would like to change user-bid over iterations rather than linearly in future. If an A_u is unable to establish its SLA even after reaching its maximum budget, it will be considered as a failed job. Resources requested by an A_u are priced by A_p (provider-agent) using the unit price of each particular resource. These unit prices do not change over iterations. However, in future, we would like to change the prices based on supply and demand. The cost *C* of a task, *T* requested by a particular A_u can be formalized as follows:

$$C(T_{A_u}) = \sum_{m=1}^{n} Req_m \times P_m(A_p) \qquad (1)$$

Where *m* refers to the resource type requested by A_u. Typically, this can be storage, CPU, and memory. However, we conduct our simulation only with storage and CPU.

n is the total number of resource types, Req_m means required resource amount of type *m*

P_m is the unit price (e.g. price/GB storage) for type *m*. This is a function of a particular provider A_p

Algorithm 1: Dealing Users with Group-based Iterated Contract Net Protocol

```
1.1   PROCEDURE: ITERATED_CONTRACT_NET_PROTOCOL
1.2   begin
1.3       set job-settled false
1.4       set continue-iteration true
1.5       set number-of-iterations 1
1.6       begin
1.7           SUB-PROCEDURE: RECEIVE_$cfp_u$s
1.8           evaluate $cfp_u$s by $A_t$
1.9           call appropriate groups $G$s
1.10      end
1.11      begin
1.12          SUB-PROCEDURE: INTERACT_GROUP
1.13          while (continue-iteration = true)
1.14              [ foreach cfp-list
1.15                begin
1.16                    SUB-PROCEDURE: EVALUATE_ cfp_BY_G
1.17                    if (job-settled = true)
1.18                        [Remove the cfp from cfp-list]
1.19                    end
1.20                    else Don't remove the cfp from cfp-list
1.21                        call the corresponding $A_u$ for revising cfp
1.22                    end
1.23                end
1.24                increment of number-of-iterations by 1
1.25                begin
1.26                    SUB-PROCEDURE: EVALUATE_POTENTIAL_GROUP
1.27                    if (length of potential-group = 0 or length of cfp-list = 0)
1.28                        [set continue-iteration false]
1.29                    end
1.30                end
1.31              ]
1.32      end
1.33  end
```

c) Optimize user-defined preferences in the group-based federated grid

In our implementation, we distinguished user-defined preferences in two ways. One is resource optimization which includes optimization for storage and CPU and another one is budget optimization. As we are using group-based strategy, it is easier to optimize a particular resource type, since typically a group only receives *cfp* with those preferences which the group is specialized for. For example, if a user's preference is storage, he goes under group 2, since the group is comprised of those providers who have more storage power. Therefore, the optimization of a particular resource is done via negotiating different providers within a particular group. Hence, a task may have to be shared by several providers. A large-scale task can be shared as the following steps:

– *Task decomposition:* involves decomposing large task into subtasks. The task decomposition is typically done by the dispatcher.

– *Task allocation:* refers to assigning the subtasks into different providers.

– *Task accomplishment:* is the completion of the subtasks by the respectively dedicated providers, which could further include decomposition and subtasks assignment.

– *Result synthesis:* includes passing the results from different providers to the corresponding provider (usually who initiates the negotiation). The corresponding provider then composes the results and passes it to the user.

V. Simulation Results and Evaluation

We conduct our simulations according to the resource configuration presented in Table 1. Column 1 of Table 1 represents different parameters that a user and a provider use to set their agents. In our simulation environment, one can accommodate a large number of users as well as providers. To set this large number of users and providers with different requests and offers, we use ranges of values so that each participant can select a value from its respective range. All users' requests are set using the Column 2 ranges and all providers' offers are set using the Column 3 ranges automatically. Since, the provider agents do not change their resource prices over iterations; we use only a single range to define a resource unit price. The first range [1-5] is used to refer to the price for 1 GB storage and the second range [10-20] is used to refer to the price for the processor of 1 MIPS (Million Instructions Per Second) capacity. The deadline parameter might not be consistent in case of simulation and so we are not using time parameter to pricing resource cost (e.g.

\$3/MIPS/hour). In addition, we assume concurrent arrival of different requests and offers.

Table 1: Resource configuration

User/provider-level parameter	User-level-range	Provider-level-range
Storage/diskspace (GB)	200-600	6000-10000
Number of CPUs (MIPS per CPU)	10-30	800-850
Minimum Budget/demand (\$)	500-1000	1-5 (/GB), 10-20 (/MIPS)
Maximum Budge(\$)	4000-5000	Not Considered

a) Evaluation criteria

In the Netlogo framework, three different results can be obtained based on the interaction of A_u (user-agent) and A_p (provider-agent). The first result describes the job rejection rate for an A_p. Job rejection occurs due to scenarios such as disagreement of resource prices or unavailability of resources. This rate is calculated using two parameters - the total number of rejected jobs ($J_{rejected}$) and the total number of requested jobs ($J_{requested}$). The job rejection rate is assumed to range from 0 to 1. The job rejection rate, R_{rate}, is given by:

$$R_{rate} == \frac{J_{rejected}}{J_{requested}} \quad (2)$$

We conduct our experiment with 5000 users and 250 providers and results are compared between group-based *i*CNP and traditional *i*CNP. The traditional approach is, using *i*CNP without applying our group-based strategies. At first, we compare these two approaches in terms of job rejection rate. Job rejected rate is plotted in terms of a number of interactions between users and providers (requests).

Figure 3 demonstrates the job rejection rate patterns between the two approaches. The horizontal axis describes the number of interactions between users and providers and the vertical axis shows the rate. The number of interactions is more than the total number of the user, since the protocol is iterative, which allows failed users re-interact to providers.

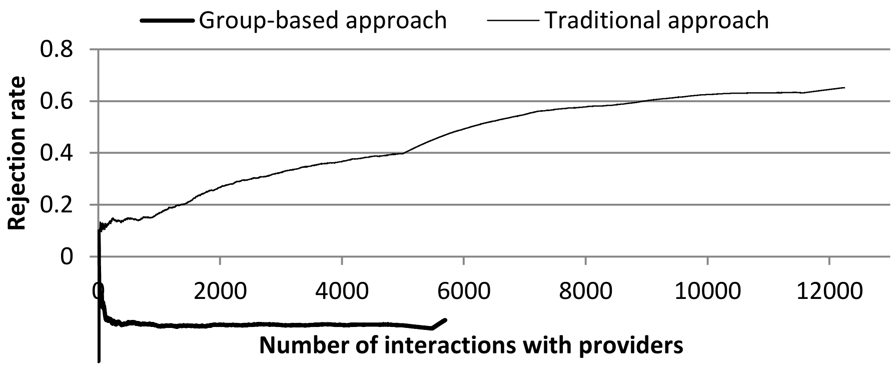


Fig. 3: Job rejection rate comparison

However, in terms of a number of interactions, our group-based approach outperforms than the traditional one, since a high number of interactions would require high communication process, which degrades system performance and keep failed users waiting for the following rounds. Our group-based optimization strategy helps to minimize the number of interactions by switching to appropriate groups based on users' optimizations. Even though the traditional approach uses optimization policy, because of not using characterizing jobs and grouping strategy, any provider can receive any job, which minimizes the probability to meet a job requirement by a single provider. The parallel trend of a particular rejection rate with the horizontal axis refers to accepting jobs and keeps the rejection rate constant. For the group-based approach, initially, the rejection rate fluctuates to 0.5. This happens due to rejecting a few jobs in the beginning. Then it abruptly goes down due to starting accepting jobs and almost keeps constant. At the end, some jobs are rejected. This might occur due to unavailable resources or the users reach their maximum budgets without getting their SLAs established. The second result demonstrates the total revenue earned by a provider or a group. It sums only the prices of the accepted jobs. Hence, the total revenue, Erev, is:

$$E_{rev} = \sum_{l=1}^{j} M_i \quad (3)$$

Where l denotes the executed job number, j denotes a total number of executed jobs and M_i defines agreed price (between a user and a provider) for the l^{th} executed job. For revenue as well group-based approach performs better than the traditional (Figure 4). The variation in a number of interactions can be explained in a similar way as aforementioned. For revenue, it is an upward trend except while rejecting jobs. During rejection the trend keeps constant. For group-based approach, the trend is almost straight, which means jobs are accepted smoothly without many iterations. On the other hand, the trend for traditional approach starts off increasing (accepting jobs) smoothly, then after 5000 interactions (i.e. first round finished by dealing with 5000 users), it stops moving up (rejecting jobs) and gradually going up through a couple of iterations. In the end, for both cases system receives no revenue. The third output illustrates how the resources on provider side are utilized. In this paper, we consider the utilization of storage and CPU. The percentage of utilization, U_m for a resource of type m by a provider A_p can be calculated by using the following formula:

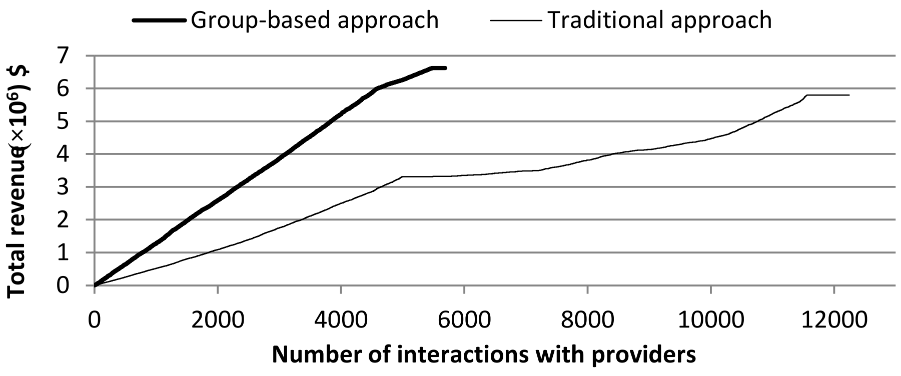


Fig. 4: Total revenue comparison

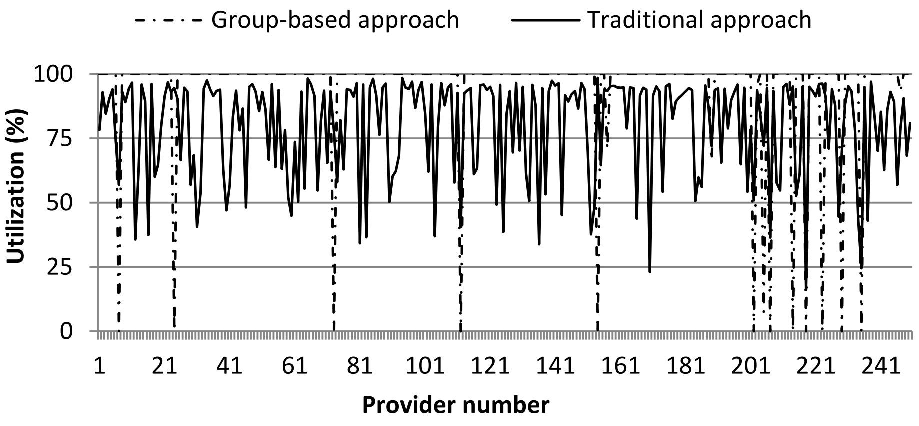


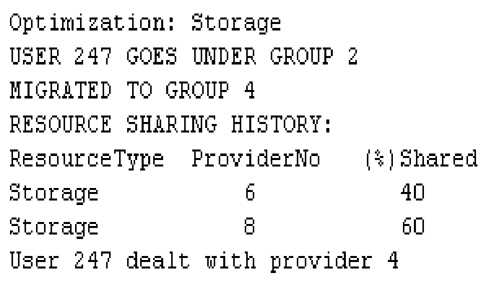
Fig. 5: Resource (disk space) utilization comparison

$$U_m(Ap) = \left[\frac{initial\ resource_m - available\ resource_m}{initial\ resource_m}\right](Ap) * 100 \quad (3)$$

For resource utilization, we obtain similar patterns for both disk space and processors. Hence, we explain the utilization for disk space only (Figure 5). The simulation pattern illustrates the utilization pattern for 250 providers (along x-axis). Unlikely in the group-based approach, traditional approach does not share resources between providers. Hence, a chance to utilize more resources by a single provider decreases. For example, if a provider is unable to fulfill a user's requirements, the provider has to reject the job, since the provider does not support sharing resources with other providers. On the other hand, in our group-based approach, even if a provider is unable to fulfill a user's requirements, the provider still can communicate with other providers within the group and share resources. Hence, the chance of accepting jobs and thus utilization resources increases. For group-based approach, most of the providers achieve maximum utilization (100%) and a few of them are unable to utilize any resources. Typically, these providers are dedicated to optimize budget constraint by users and propose highest resource cost. Hence, it becomes hard to optimize budget by these providers and could not able to utilize any resources. However, the traditional grid providers could contribute their resources in a group-based federated grid, since the chance of utilizing maximum resources is higher in the group-based system than the traditional one. For the traditional approach, the trend is scatted across the figure, which implies the adoption of no optimization strategy.

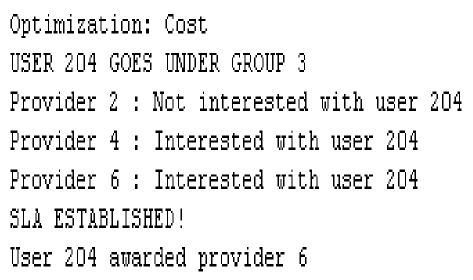
Example 1: Resource Optimization. Figure 6 shows the throughput of provider-provider negotiation within group-4 to optimize storage. Due to unavailable resources, user 247 is migrated from group-2 to group-4. 40% of the user's storage demand is met by provider-6 and rest 60% is shared by provider-8. Provider-4 is the group-4 correspondent here.

Example 2: Budget Optimization. Figure 7 presents the budget optimization process within group-3. Though provider-4 and provider-6, both accept user 204's *cfp,* user 204 awarded provider-6, since provider-6's asking bid was less than that of provider-4. Please note that provider-6 is appeared in both groups, this is because of taking the two shots from different simulations. In practically, one provider cannot exist within different groups.



```
Optimization: Storage
USER 247 GOES UNDER GROUP 2
MIGRATED TO GROUP 4
RESOURCE SHARING HISTORY:
ResourceType  ProviderNo   (%)Shared
Storage           6           40
Storage           8           60
User 247 dealt with provider 4
```

Fig. 6: Resource (storage) optimization



```
Optimization: Cost
USER 204 GOES UNDER GROUP 3
Provider 2 : Not interested with user 204
Provider 4 : Interested with user 204
Provider 6 : Interested with user 204
SLA ESTABLISHED!
User 204 awarded provider 6
```

Fig. 7: Budget optimization

The illustrations presented in this paper with the resource configuration in a way such that *"supply is equal to demand".* However, we conduct simulations with such other scenarios, which are *"supply is greater than demand"* and *"supply is less than demand".*

Table 2: Scenario based comparison between group-based federated grid (GFD) and traditional grid (TG)

Scenario	Number of users	Number of providers	Task generation time (sec)		Offer generation time (sec)		Number of rejected jobs		Number of iterations		Net simulation time (sec)	
			GFG	TG	GFG	TG	GFG	TG	GFG	TG	GFG	TG
Supply = Demand	5000	250	4.95	8.59	1.625	1.516	213	737	3	7	132.76	2072.10
Supply > Demand	4000	250	4.96	5.57	5.266	2	0	0	3	9	113.18	1984.23
Supply < Demand	5000	200	4.65	4.17	2.547	1.453	1043	1449	2	5	68.23	706.72

For all three cases, we use the resource configuration according to Table 1. Table 2 demonstrates that our group-based approach outperforms in most cases than the traditional approach. The group-based approach consumes less simulation time, produce less number of iterations, and even rejects fewer jobs in all three scenarios except when supply is greater than demand. Due to more supply compared to demand, the chance of accepting jobs by traditional provider increases.

VI. Conclusions and Future Work

The vision of grid computing is to collaborating computer resources that are distributed. However, due to the dynamic nature and heterogeneity of these resources, seamless collaboration is hindered. Agents are well known for collaborating distributed resources due to their autonomous and proactive nature in building decisions without human intervention. In this paper, we proposed an agent-based Iterated Contract-net-protocol to deal with users' QoS and providers satisfactions. We characterize users in terms of their preferences and switch them to the groups accordingly. A grouping strategy has been proposed for the federated grid, where grouping formed in terms of providers' availability and users' preferences. Our strategy enabled users to receive a stronger QoS without letting them waiting much. The less number of iterations and thus the less time consumption while negotiating between users and providers provided the justification of our approach. The adoption of such a group-based approach would produce less communication delay while dealing with thousands of users as well as providers. In addition, this would minimize execution uncertainty while ensuring better payoff and resource utilization by providers.

In future, we would like to conduct our experiments in real grid scenarios such as Globus, Nimrod in order to test the real-time adaptability and feasibility of our work. Future work would also extend the characteristic functions to distinguish between users in order to maximize the number of delivering required QoSs. We further would like to experience the agent behavior in terms of dealing with distributed environment and adapting accordingly.

References Références Referencias

1. C. Massimiliano, G. Stefano, "Resource allocation in grid computing: an economic model", J. WSEAS Trans. Comp., vol. 3 pp.19-27, 2008.
2. H. Izakian, A. Abraham, B.T. Ladani, "An auction method for resource allocation in computational grids", Future Generation Computer Systems, vol. 26, pp.228-235, 2009.
3. W.-C. Ni, L.-C. Liu, C. Wu, "Services selection based on commodity-market in grid workflow", Journal of Computer Applications, vol.27, pp.2973-2975, 2007.
4. C. Isaac, S. Ramon, A. Oscar, J. Liviu, F.R. Omer, "Optimizing decentralized grid markets through group selection", Int. J. Web Grid Serv., vol. 4, pp. 357-366, 2009.
5. A. Di Stefano, C. Santoro, "An economic model for resource management in a Grid-based content distribution network", Future Generation Computer Systems, vol. 24, pp. 202-212, 2008.
6. A. Foster, C. Kesselman, "The grid blueprint for a future computing infrastructure", Morgan Kaufmann, 1999.
7. G. Kunal, G. Arobinda, "Resource Selection in Grids Using Contract Net", in Proceedings of the 16th Euromicro Conference on Parallel, Distributed and Network-Based Processing (PDP 2008), IEEE Computer Society, 2008.
8. R. Rajiv, H. Aaron, B. Rajkumar, "Sla-based cooperative super scheduling algorithms for computational grids", J. ACM Transaction of Autonomous and Adaptive System, 2008.
9. K. R. Ku-Mahamud and H. J. A. Nasir, "Ant Colony Algorithm for Job Scheduling in Grid Computing," 2010 Fourth Asia International Conference on Mathematical/Analytical Modelling and Computer Simulation, Kota Kinabalu, Malaysia, 2010, pp. 40-45. doi: 10.1109/AMS.2010.21
10. Laizhi Wei, Xiaobin Zhang, Yun Li, Yujie Li, " An Improved Ant Algorithm for Grid Task Scheduling Strategy", in International Conference on Applied Physics and Industrial Engineering, vol. 24, pp. 1974-1981, 2012.
11. Sonal Yadav, Amit Agarwal and Ravi Rastogi, "Cost-based Job grouping and Scheduling Algorithm for Grid Computing Environments", International Journal of Computer Applications vol. 91, no.15, pp. 21-27, April 2014.
12. Pinky Rosemarry, Ravinder Singh, Payal Singhal and Dilip Sisodia, "Grouping Based Job Scheduling Algorithm Using Priority Queue and Hybrid Algorithm in Grid Computing", International Journal of Grid Computing & Applications (IJGCA), vol.3, no.4, December 2012.
13. Y. Sallez, T. Berger, C. Tahon, Simulating intelligent routing in flexible manufacturing systems using NetLogo, in: IEEE International Conference on Industrial Technology, pp. 1072-1077, 2004.
14. R.-C. Damaceanu, An agent-based computational study of wealth distribution in function of resource growth interval using NetLogo, Applied Mathematics, and Computation, vol. 201, pp. 371-377, 2008.
15. J.N. Michael, T.C. Nicholson, R.V. Jerry, Experiences creating three implementations of the repast agent modeling toolkit, ACM Trans. Model. Comput. Simul., vol. 16, pp. 1-25, 2006.
16. FIPA Iterated Contract Net Interaction Protocol Specification, in: Foundation for Intelligent Physical Agents, 2000 http://www.fipa.org/specs/fipa00030/