Large-scale cooperative task distribution on peer-to-peer networks

Daniel R. Karrels [a,*], Gilbert L. Peterson [b,**] and Barry E. Mullins [b]
[a] *Air Force Research Laboratory, Directed Energy Directorate, 3550 Aberdeen Ave SE, Kirtland AFB, NM, USA*
[b] *Department of Electrical and Computer Engineering, Air Force Institute of Technology, 2950 Hobson Way, Wright-Patterson AFB, Ohio, USA*

Abstract. Large-scale systems are part of a growing trend in distributed computing, and coordinating control of them is an increasing challenge. This paper presents a cooperative agent system that scales to one million or more nodes in which agents form coalitions to complete global task objectives. This approach uses the large-scale Command and Control (C2) capabilities of the Resource Clustered Chord (RC-Chord) Hierarchical Peer-to-Peer (HP2P) design. Tasks are submitted that require access to processing, data, or hardware resources, and a distributed agent search is performed to recruit agents to satisfy the distributed task. This approach differs from others by incorporating design elements to accommodate large-scale systems into the resource location algorithm. Peersim simulations demonstrate that the distributed coalition formation algorithm is as effective as an omnipotent central algorithm in a one million agent system.

Keywords: Distributed Multi-Agent System, Hierarchical Peer to Peer, Large-Scale, Command and Control

1. Introduction

Deployed Peer-to-Peer (P2P) systems are now commonly eclipsing one million simultaneous nodes [23]. Significant research efforts continue to optimize system redundancy and speed of querying the data at these scales [5]. As enterprises collect more and more data, the use of datamining to identify trends becomes more attractive [35]. Because the P2P agents store the data, they can be leveraged to also distribute the datamining computation. However, at one million agents, such a large-scale system requires a new method of organizing the agents and algorithms into task agent coalitions. Both resources (computation and data present at the agent) must be included in the tasking process. For scalability, the agents operating under these conditions must be flexible, cooperative and multi-taskable.

This paper addresses the problem of cooperative task Command and Control (C2) of a large-scale Distributed Multi-Agent System (DMAS). The system is composed of cooperative multi-taskable agents. Cooperative agents seek to maximize global utility, rather than personal gains (i.e., not self-interested or greedy). This requirement provides honesty, and enforces the property that any bids or statements of available resources by an agent toward a coalition proposal include all available resources the agent provides. This makes the process scalable, as the models governing negotiations do not include competitive bartering and bidding for tasks.

The primary contribution of this paper is the Distributed Likelihood of Execution (DLoE) algorithm. The DLOE algorithm uses a coalition formation task scheduling model to maximize the work throughput in the system. The DLoE algorithm assigns tasks to agents based on the task's expected Likelihood of Execution (LoE) at the particular agents. The LoE is computed from an agent's scheduling data, and represents

*The authors would like to acknowledge the funding and support of the AFOSR and AFRL. The views expressed in this article are those of the author and do not reflect the official policy or position of the United States Air Force, Department of Defense, or the U.S. Government.
**Corresponding author. E-mail: gilbert.peterson@afit.edu.

potential resource contentions. The scheduling data uses a small amount of overhead at each agent, and is updated periodically to create a general view of agent scheduling data for subgraphs of the Hierarchical Peer-to-Peer (HP2P) topology.

The HP2P overlay leveraged in this paper is the Resource Clustered Chord (RC-Chord) HP2P resource management overlay [17]. RC-Chord provides a robust network organization framework with two hierarchies for organizing and locating agents by both address and available resources. The RC-Chord overlay is extended to maintain the scheduling data for the clusters used by the DLoE algorithm. Because the DLoE algorithm is built on RC-Chord, it inherits the redundancy and the robustness to peer churn of the RC-Chord overlay.

Simulations exercise the DLoE coalition formation algorithm on systems of one million agents. Results are compared against an omnipotent fully centralized optimal algorithm and a greedy algorithm [2]. The greedy algorithm performs worst, forming task coalitions that execute tasks up to 25% slower than the other algorithms. The DLoE algorithm consistently outperforms the greedy algorithm, and yields overall performance to within one standard deviation of the centralized optimal algorithm's results. The results of DLoE testing are encouraging, and lay a framework for future use in large-scale DMAS application suites.

The following section discusses the coalition formation problem definition of Abdallah and Lessher [1]. Section 3 presents related work on coalition formation, multi-robot task allocation, and resource coordination peer-to peer networks. The DLoE algorithm is presented in Section 4, leading into experimental setup in Section 5, results in Section 6, and conclusions and recommendations in Section 7.

2. Coalition Formation

Coalition formation focuses on the construction of teams of agents to execute tasks, with the goal of employing the capabilities and assets of under-utilized agents to achieve larger and more sophisticated tasks. A task is defined as a function, with a desired end state, that requires one or more agents and resources to complete.

Forming optimal coalitions requires input from each agent in the system, and is an $\mathcal{NP}$-complete problem [32]. As defined by Abdallah and Lesser [1], consider the set of tasks $T = \langle T_1, T_2, ..., T_q \rangle$. Each task T_i is defined as $T_i = \langle u_i, rr_{i1}, ..., rr_{im} \rangle$, where u_i is the utility gained for accomplishing task T_i and rr_{ik} is the amount of resource k required by task T_i. The set of agents is $I = \{I_1, I_2, ..., I_n\}$, where each agent $I_i = \langle cr_{i1}, cr_{i2}, ..., cr_{im} \rangle$, and cr_{ik} is the amount of resource k possessed by agent i.

The coalition formation problem is defined as the allocation of the subset of tasks $S \subseteq T$ to agents that maximizes the global utility, U,

$$U = \sum_{i|T_i \in S} u_i. \tag{1}$$

Task allocation algorithms build a set of coalitions $C = \{C_1, ..., C_{|S|}\}$, where $C_i \in I$ is the coalition assigned to task T_i, such that each task coalition provides enough resources of each type to satisfy that task's requirements.

$$\forall T_i \in S, \forall k : \sum_{I_j \in C_i} cr_{j,k} \geq rr_{i,k}. \tag{2}$$

A constraint on the problem is that each agent is capable of only executing a single task:

$$\forall i \neq j : C_i \bigcap C_j = \emptyset. \tag{3}$$

This form of the coalition formation problem assumes single-task agents [11], "all or none" resource allocation, and exponential time coalition formation due to task group enumeration [27]. These properties are modified to provide the ability to scale the tasking of a cooperative coalition on an HP2P network.

3. Related Work

This section summarizes existing approaches to solving the coalition formation problem. It begins with traditional solutions and moves into related forms. Toward the solution developed here, this section also introduces P2P overlays and the RC-Chord HP2P structured overlay.

3.1. Coalition Formation

Shehory and Kraus [30] describe two methods for coalition formation using reward incentives. In a negotiation-based formation, all single agent coalitions

begin by interacting with other agents to determine if forming a joint coalition can yield a higher payout than remaining alone. In the case that the two agents both determine their profit can be increased by forming a coalition with each other, they negotiate a sharing of the additional payout yielded by forming the coalition. The agents negotiate a fair split of the profits based on greedy [9] or other semantics, and the payout may be different for the two agents. In the negotiation algorithm, this process occurs between all pairs of agents, and each agent attempts to form a coalition with its most profitable partner.

The second algorithm builds upon the Shapley formula [37]. This is a centralized algorithm in which a single agent collects information about the resources and other relevant information from all other agents in the system. The agent then calculates the Shapley value, which involves finding the payout values of all 2^n pairs of agents. These payouts are organized into a prioritized data structure, and all agents are then informed of the new coalition schedules. This centralized algorithm requires $O(n)$ communications (it contacts each agent twice) and $O(2^n)$ computations.

The Contract Net Protocol (CNP) [33] is a contract system to allocate tasks, or portions of tasks, to one or more agents. Given a system of agents, any agent with a surplus of work to perform may start an inverted blind auction (contract proposal) for which other agents with a surplus of resources can bid. The bidder with the most attractive offer (lowest payout) is awarded the contract. Agents form networks of auctions, and may join and part them at will. This concept can be applied in a HP2P structure, where agents are naturally organized into clusters. This method is extended to build upon more modern communications facilities, such as ordered delivery of TCP and higher assumed bandwidth, easing constraints in the original protocol [29]. The CNP is useful in both heterogeneous and homogeneous systems in which agents do not have full information about other agents. Rather, the agents submit themselves as candidates for processing a certain task, based on availability and capabilities, without revealing their full state information.

The coalition formation problem can also be considered a variant of the task allocation problem. The Multi-Robot Task Allocation (MRTA) problem [10] is given m robots, each capable of executing one or more tasks, and n weighted tasks, each requiring one or more robots, the goal is to assign robots to tasks to maximize the overall expected performance, taking into account the priorities of the tasks and the efficiency ratings of the robots [3]. The MRTA problem is $\mathcal{NP}$-hard [9,11].

This research effort examines instantiations of the multi-robot multi-task environment, in which agents are capable of performing tasks requiring either one or more agents, and with each agent capable of performing one or more simultaneous tasks. Tasks will be introduced at runtime (online assignment), and the form and goals of those tasks are not known ahead of time. Application of this paradigm to multi-agent systems is not yet fully understood, and applying it to large-scale multi-agent systems remains an open problem.

These concepts are leveraged and extended in the design of the DLoE algorithm. This algorithm uses the properties of an HP2P overlay, combined with simplifying assumptions and a heuristic to support coalition formation algorithms in a large-scale system. The DLoE algorithm is covered in more detail in Section 4.2.

3.2. Large Scale P2P Overlays

A communications overlay is the set of protocols and algorithms necessary to build and maintain a topology of nodes in such a way as to guarantee a set of performance parameters. In the context of P2P technologies, overlay structures describe the formation of nodes into a system of peers capable of identifying and locating remote nodes without foreknowledge of their exact location or being certain if the requested targets exist. Such a consideration is necessary in many environments where the scale of those systems is large enough to prevent global knowledge. First generation systems solved this problem by query broadcast [8,6], but this solution fails to scale. Newer systems have developed more advanced techniques for locating remote nodes, and the utility of such systems has brought about the emergence of mainstream P2P applications [28,21,34,4,14,13,15].

ML-Chord [22] applies these technologies to create a system organized by available resources. ML-Chord is a two-layer HP2P overlay network, where the top (or bridge) layer joins super peers from each of the clusters at the second layer. These Category Layer clusters each represent a single resource, and agents join one or more clusters based on their available resource(s). Chord is used as the base agent location protocol, and is suitably modified to accommodate search by resource category. Two notable disadvantages of ML-Chord are its fixed size (two layers), and limited scalability for large-scale systems. RC-Chord extends ML-

Chord to address these limitations, and is the framework for the DLoE coalition formation algorithm.

3.3. RC-Chord

RC-Chord [17] is an HP2P overlay network based on the Chord protocol [34]. It incorporates the ability to scale to many levels, with each level composed of one or more clusters. Each cluster is a stand-alone instance of Chord, and connects to a cluster in the next higher level of the hierarchy through a set of super peers. A cluster may have zero or more sub-clusters attached to it, forming a tree from a single super cluster root.

RC-Chord associates each agent with one or more resources, and each cluster, with the exception of the super cluster, represents a single resource. Agents connected to a particular cluster all have that cluster's resource in common, and agents join a cluster for each resource they possess. The super cluster includes multiple agents from each resource sub-graph to form the root of each resource hierarchy. RC-Chord supports searching for agents by global identifier or resource. The hierarchy grows and shrinks dynamically to accommodate network churn and abundant resources. An abundant resource is a resource that many or all agents in a system may possess, such as processor time.

Each cluster consists of a set of super peers, in addition to the larger percentage of normal peers. Cluster super peers are responsible for message routing and maintaining a shared database of resource availability for their leaf node agents. All requests and obligations of resources are processed by cluster super peers. Resource requests implicitly carry an intent to obligate, thus avoiding the need for multi-stage transactions. Instead, each request is examined by a super peer to ensure available resources, and an obligation request is sent to leaf peers for a quantity of the resource. Leaf peers respond with either accept or deny, and the resource is considered obligated. Once the super peer has collected the necessary quantity of the resource, the shared database is updated, and replies are sent to those parties involved. In the case of the requestor, the details of the request are returned (which agents, and how much of each resource at those agents), and the leaf peers receive a session identifier (including obligation duration, requestor, etc.). This is a simplification of the process, as error checking and contention issues are omitted for brevity.

Given the RC-Chord system parameters, each agent knows roughly how many clusters exist in the levels below it. In addition, super peers receive periodic updates about the average quantities for all agents containing specific resources. This value is used to calculate the average LoE for the new task for the local cluster and those clusters in lower levels.

Figure 1 shows an example RC-Chord instance with seven clusters. Three resources are present, with all three represented in the super cluster. When a resource's agent population high-threshold is exceeded at the super cluster, a new cluster for that resource is created at the second level. Once a cluster at the second level is filled, agents joining the system with that resource are attached to a new cluster at the next level of the hierarchy. Figure 1 shows a single level-two cluster for each of the system's three resources. Agents possessing resource three have continued to join the system, and new clusters for that resource were created at level three. This process repeats, with sub-graphs of each resource growing outward from the super cluster to accommodate new agents joining the system.

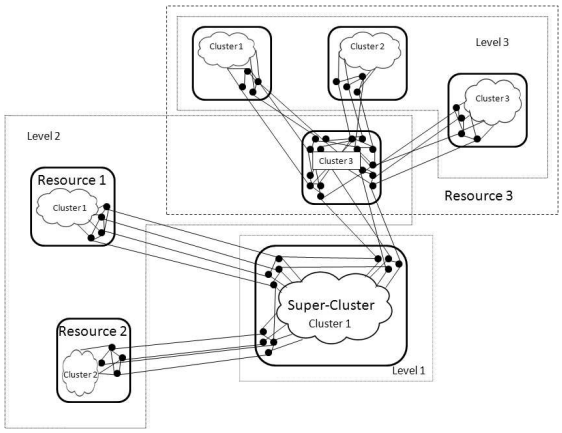

Fig. 1. An example RC-Chord instance with three levels. The super cluster exists as the sole level-one cluster. Its agents serve as super peers for level-two clusters, with a single level-two cluster for each resource. Each resource subgraph may extend downward into additional levels as necessary, with a branching factor proportional to the ratio of peers to super peers and the size of each cluster.

4. Methodology

This section introduces extensions to the coalition formation problem to form the cooperative coalition formation problem. The additions redefine agents to be multi-taskable, allow agents to share full or partial resources between tasks, and allow tasks to split allocations of a resource between multiple agents.

The DLoE algorithm, which is used here to solve the cooperative coalition formation problem in a large-scale DMAS, is also presented. The algorithm is built upon the RC-Chord structured HP2P overlay and borrows concepts from contract protocols. A key distinction between existing contract protocols and the DLoE algorithm is that the DLoE algorithm does not use negotiations. Instead, the algorithm uses knowledge of agent workloads and assigns tasks to agents to maximize work throughput and minimize task duration.

4.1. Peer-to-Peer Task Model

The P2P task model is designed to accommodate the diversity of the tasks expected to be executed within a large-scale multi-agent system. The modifications to Abdallah and Lesser's model are the redefinition of utility as work, the introduction of task priority, and addition of task synchrony to model practical distributed algorithms. These changes reflect the scale of the new environment in which they operate, wherein global knowledge and synchronization are likely no longer achievable [18,39].

Without the possibility of global synchronization and due to the growth rate of the classical coalition formation problem, this task model introduces the idea of work to motivate the decision processes of agents. Rather than spending long periods of time and bandwidth during the coalition formation negotiation process, agents instead focus on forming coalitions quickly and leveraging the scale of the system to achieve maximum useful work throughput. To this end, the idea of utility is replaced by a more tangible unit called work. Each task, T_i, specifies an amount of work, w_i, that must be performed to complete the task.

Tasks are defined as $T_i = \langle w_i, p_i, s_i, rr_{i1}, ..., rr_{im} \rangle$, where w_i is the number of units of work necessary to complete task T_i, p_i is the task priority, and s_i is the task synchrony.

Each agent is capable of executing one unit of work per time unit. Since agents are multi-taskable, they may be members of multiple coalitions and must choose which task to execute at each time step. The task priority, p_i, provides a mechanism for runtime tuning, as well as scheduling fidelity. Task priorities for this model fit within a range of [1,10], with 10 being the highest priority. At each time step, agents with multiple tasks use the priority and a decision process (Section 4.1.1) to choose which task to execute. Maximum work throughput is achieved when each agent has a task to execute at each step, and establishing a local scheduling policy based on task priority ensures that higher priority tasks are completed first.

Not all tasks are completely parallel [20], and may require periodic barrier synchronization points. These barrier synchronization points halt processing on all agents that have reached the barrier until all other agents arrive. This generalized mechanic is used to represent scenarios in which substantial variation exists between the processing capabilities of individual heterogeneous agents, resource contention arises, or to accommodate tasks that require frequent updates or synchronization between execution threads.

Each task is assigned a task synchrony value, s_i, that specifies the number of steps each agent can perform before reaching a barrier. Upon reaching a task synchronization barrier, an agent halts processing on that task until all other agents assigned to the task reach that barrier. During that time, the agent at the barrier removes the task from its ready queue, and instead executes work for other task coalitions of which it is a member. Once the synchronization barrier has been met by all other agents, the task becomes ready to execute by all agents in the task coalition.

To accommodate the allocation of an agent's resources to multiple coalitions, Abdallah and Lesser's model is extended to include cr^i_{jk} as the portion of agent j's supply of resource k that is allocated to coalition i, which is not to exceed the agent's total supply of resource k, defined by $|cr_{jk}|$ (Equation 4). This allows each agent the flexibility to participate in multiple coalitions simultaneously. It also removes the "all or none" property, thus increasing the satisfiability of agent coalition formation by allowing partial amounts of resources, and participation in multiple coalitions.

$$\forall I_j \in I, \forall k : \sum_{T_i \in S} cr^i_{jk} \leq |cr_{jk}|. \tag{4}$$

To maximize performance, each agent can contribute only one resource (or partial resource) to a task. This constraint prohibits the same agent from joining a task more than once. It eliminates the possibility of contention in task scheduling and ensures minimum time between barriers.

The cooperative coalition formation problem is defined as the allocation of the subset of tasks $S \subseteq T$ to agents that maximizes:

$$W(t) = \sum_{i \in I_i} w_i(t). \tag{5}$$

Note that the global work throughput, W, is not resolved with the formation of task coalitions. Rather, W is time-dependent, and increases with the number of agents that are able to execute a unit of work per time. Coalition formation algorithms must therefore focus on the allocation of tasks to agents such that the number of agents per task is globally balanced.

4.1.1. Scheduling

Each agent is capable of receiving and processing multiple tasks, and may only execute a single unit of work per unit time. Agents choose which task to execute from those tasks they possess using a priority based scheduling algorithm. Scheduling follows a sampling process in which each task is weighted based on its priority. Under this roulette wheel sampling, higher priority tasks have a higher likelihood of being selected to receive processor time. This scheduling algorithm ensures fairness and progress, while deconflicting between two optimization parameters: number of tasks on the agent and task priorities.

4.2. Distributed LoE

The main contribution of this paper is the DLoE algorithm. The DLoE algorithm attempts to achieve maximum work throughput by forming coalitions for tasks. Since the algorithm is distributed and designed to operate in large-scale systems, it does not use global knowledge.

Contrary to most coalition formation algorithms, the agents in the DLoE have no decision authority about which task coalitions they join. Rather, the distributed algorithm decides the task allocation strategy that best benefits the system work throughput. The DLoE algorithm borrows from the military command paradigm: centralized authority, decentralized execution. The execution of the algorithm itself is distributed, however the authority it carries is centralized in the sense that agents are less autonomous than other approaches. The objective of this paradigm is to minimize negotiations resulting in coalition formation, thus reducing the overhead of the algorithm, and yielding higher system work throughput.

The DLoE algorithm builds on the RC-Chord HP2P structure. At the super peer level, in addition to tracking resource amounts, the super peers track the total priority points (TPP) of the nodes in its cluster and the average TPP for connected clusters. The total priority points are the sum of the priority values of all of the tasks at an agent.

Tasks are introduced to the system at any agent. When a task enters the system, the task is sent to a cluster super peer. If the super peer needs resources that are not available in its clusters or clusters beneath it, it forwards the task to the super cluster. A super cluster super peer then locates agents that can satisfy the task's requirements. The super cluster can be reached by any agent in the system in O($log_m(N)$) steps, where m is the Chord address width per cluster. The DLoE algorithm recursively searches down the most likely subgraphs to locate agents capable of satisfying resource requirements for the new task.

Algorithm 1 shows the execution of the DLoE search algorithm to locate the best agent to the system to which to assign a new task. The distributed recursive algorithm executes on a super peer in a cluster. It begins by initializing the TPP to be the average TPP of all agents in the cluster, and examines the base condition to check if the current cluster contains the best agent for the new task. The DLoE coalition formation algorithm attempts to maximize the likelihood that the new task receives processor time on each agent. The algorithm assigns points to each task, T_i, resident on a target agent, I_j, based on the priorities of those tasks:

$$priority\_points = \sum_{T_i \in I_j} priority(T_i) \qquad (6)$$

The value of each agent's $priority\_points$ is compared, and the agent with the lowest value is assigned the new task. This ensures minimum competition for processing time for the new task on the target agent, thus maximizing its likelihood to execute, and reducing the task's total execution duration.

The algorithm executes on a super peer, and therefore the agent has access to the aggregate TPP information for all agents in its cluster. The algorithm examines the LoE for the task against all agents in the current cluster. This value is computed at a super peer without further inter-agent communications, and is compared to the LoE for the sum of all sub-clusters from the current cluster. A lower LoE value is more desirable, as lower values indicate less competition in the task scheduling algorithm. Upon finding the cluster with the lowest LoE, the algorithm sends the task to a super peer in the cluster. The algorithm then recursively descends the hierarchy until it encounters a leaf cluster, or it finds the agent in the cluster with the lowest LoE.

Algorithm 1 chooseNodeRecursive($cluster, r_i$)

$clusterTPP = cluster.localTPP/cluster.numNodes$
$subNodes = cluster.subNodes(r_i)$
$subTPP = MAX\_INT$
si $subNumNodes > 0.0$ **entonces**
  $subTPP = cluster.subTPP(r_i)$
fin si
si $clusterTPP \leq subTPP$ **entonces**
  $bestAgent = chooseAgentLocal(cluster, r_i)$
fin si
$cluster[]subClusters = getSubClusters()$
$subBestTPPSoFar = max(int)$
$clustersubBestSoFar = nil$
para $i = 0 : subClusters.length$ **hacer**
  $subTest = subClusters[i]$
  $subTPP = subTest.totalTPP(r_i)$
  $subNodes = subTest.nodes(r_i)$
  $subLoE = subTPP/subNodes$
  si $subLoE < subBestTPPSoFar$ **entonces**
    $subBestTPPSoFar = subLoE$
    $subBestSoFar = subTest$
  fin si
fin para
devolver chooseNodeRecursive($subBestSoFar, r_i$)

Implicit in the decision making of the DLoE algorithm is that all holders of a resource are considered equal in quality, although perhaps not quantity. Likewise, agents contribute equal work units to each task they host. Since resources can be shared between tasks and agents execute only a single task per time step, an agent's resources are multiplexed to its task coalitions in the same way that execution time is shared between tasks. These assumptions establish a balance among agents, and reduce the search space of the DLoE algorithm.

Each node in the system periodically passes its resource amounts and TPP to one of its cluster's super peers. This information is aggregated and the TPP is tracked according to resource amount. For example, in a system that permits resource amounts [0,1000], the resource interval is split n times. For each of these n equally divided resource intervals, the count of TPP for the cluster is tracked. Along with the number of nodes in the cluster, again divided by resource interval, this vector of TPP per resource interval is all that is passed upward between clusters. This information continues an upward ascent through the levels of the HP2P network until it reaches the super cluster. The DLoE algorithm uses this information to build an approximate picture of the TPP for each sub-graph. When searching for agents to satisfy resource requirements for coalition formation, the DLoE heuristic chooses the route that minimizes the expected scheduling contention by examining the aggregate TPP at each cluster. Finding the node with minimum TPP will yield the highest likelihood of execution for the new task and minimize its expected task duration.

The data collection and dissemination accounts for a small amount of overhead. Given 10 resource intervals on a 64-bit architecture and two data structures (TPP and resource availability), the information passed from a cluster to its next higher level cluster consumes approximately 160 bytes. A system of one million agents, with 1000 agents per cluster, will have roughly 1000 clusters. This entire periodic maintenance process consumes approximately 160KB per update period for a large-scale system. Cluster super peers store the TPP data for members of their cluster, updating as new information becomes available. This results in a maximum of $num\_intervals * 2^m$ entries stored per super peer, or 82KB of memory on a 64-bit machine with 10 resource intervals and a maximum of 1024 agents per cluster.

5. Experimental Setup

To demonstrate the effectiveness of the DLoE algorithm for large-scale DMAS coalition formation, an RC-Chord implementation is created to operate within the Peersim [16] P2P simulator. The coalition formation algorithms in the simulations are built upon the RC-Chord overlay, and experiments evaluate systems of one million simulated nodes. The RC-Chord HP2P structure consists of up to 4096 nodes per cluster ($m = 12$) and one super peer for each 512 peers to handle network churn, both tunable parameters. These numbers were chosen based on experimental results to minimize mean hop length between nodes, and by choosing a maximum cluster size that maintains a balance between internal maintenance overhead and the overall number of clusters in the system[17].

Experiments last 15,000 time units each, which is a suitable length of time for data trends to become stable. At each time step, tasks are allocated according to the desired load for the simulation. Tasks complete when their required number of work units have been executed by its coalition's agents. Each experiment is run 30 times to provide a means for statistical comparison.

During execution of the experiments, the system undergoes a churn rate proportional to the size of the system. At each time step, approximately 1% of the agents in the system, chosen randomly, are removed from the system, and the same number of agents is reintroduced into the system at randomly chosen locations. This churn helps to ensure that the underlying network management protocols are functioning properly, to include super peer nomination and promotion. Any coalitions that are damaged as a result of losing a member agent undergo a repair mechanism, and one or more new agents are chosen to replace the lost agent(s)[17].

These experiments compare three coalition formation algorithms:

- CRP: An agent is chosen at random. If that agent meets the task's resource requirements, then it is added to the task coalition [2].
- CLoE: Tasks are allocated to agents to maximize the probability of the task receiving processor time slots. This centralized algorithm uses global knowledge.
- DLoE: Similar to the CLoE algorithm, except that global knowledge is removed and the algorithm is distributed.

The CRP [2] and CLoE algorithms serve as baseline algorithms. The CRP is a centralized algorithm with full knowledge[19,31,36,38,40]. The CRP algorithm randomly chooses agents to include in a task coalition. However, this process can fail as the targeted agent may not have the proper resource type, or may have an insufficient quantity of the resource. As such, the CRP algorithm consumes additional bandwidth and is the only algorithm that can miss.

The CLoE optimizes a task's LoE with global knowledge. The CLoE algorithm is an adaptation of the CNP [25], modified to operate in an environment where agents are required to volunteer. The advantage of global knowledge for CLoE is that no ratio averaging is used in the decision process. Instead, the algorithm identifies the agent in the system with lowest TPP by examining every agent in the system at each coalition formation point. With this global knowledge, the CLoE algorithm serves as an optimal baseline. Because the CLoE operates in simulation, the global knowledge is available. However, in a real world system the message passing to the central server would cause a denial of service. Although the CLoE algorithm provides accurate results using its global centralized search, the key contribution of the DLoE algorithm is that its performance is close to that of the CLoE algorithm, but uses a tractable algorithm that can be achieved for large scale distribute multi-agent systems.

5.1. Factors

Table 1 describes the factors in these simulations. The objective of testing under these conditions is to exercise the critical parts of the coalition formation algorithms, and examine the resulting effectiveness metrics.

Table 1
Simulation Factors

Variable	Range
Algorithm	CRP, CLoE, DLoE
Task Synchrony (s_i)	-1, 1, 2, 3, 4, 5, 10, 15 (steps)
Load	500, 1000, 1500 (tasks per step)
DLoE Update Interval	0, 5, 10, 25, 50 (steps)

Tasks are allocated uniformly across each simulation time line, with task priorities following the probabilities shown in the bi-modal distribution of Figure 2. The dominant distribution has a mean of 3.5 ($\sigma = 1.4$) and represents the creation of tasks during normal operations. The target application domain for this research is that of network operations, to include network defense. As a result the task priority distribution also includes a second mode of higher priority, and lower frequency, which represents unpredictable and emergent tasks, such as handling a network attack. This second Gaussian distribution has mean 8.0 ($\sigma = 1.5$) and represents real world situations that induce a sudden burst of task creations. Tasks assigned into this category are centered around priority eight, and represent a small percentage of all tasks.

The primary loading factor is the number of tasks created per time step (Tasks Per Step (tps)). The incoming task work load, or work generated, is measured in units of work per unit time, with each task requiring between 750 and 1000 units of total work to complete. Given the optimal work throughput of the system at one million units of work per unit time, the values of 500tps, 1000tps, and 1500tps represent under-loaded, critically-loaded, and over-loaded systems. The objective of these values is to measure the effectiveness of each algorithm in these scenarios.

The DLoE level heuristic algorithm has one additional critical process variable: the update interval.

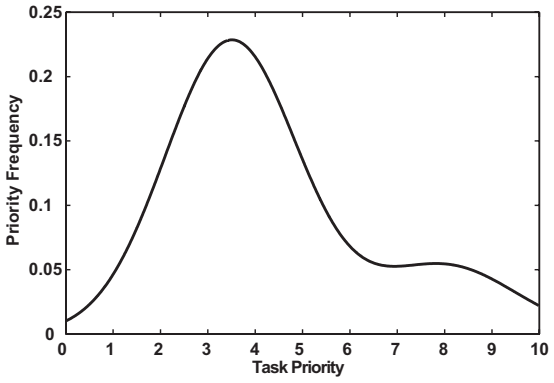

Fig. 2. Simulation task priority distribution. This is a bi-modal mixture model of two Gaussian distributions, one for standard operating missions, and a second for low probability, high priority missions.

The update interval is the duration, in simulation time steps, between updates of DLoE shared resource data (Section 4.2). Unless otherwise noted, the level heuristic update interval is kept at five, and is determined experimentally to minimize network maintenance bandwidth usage, and still provide super peers with accurate approximations of cluster TPP's[17].

5.2. Control Factors

Each agent is assigned one of five resources, with a quantity between 400 and 1000 units. The range of resource quantity was chosen experimentally to ensure that each agent could reasonably be part of multiple task coalitions. This introduces sufficient diversity in the system to validate the model by forcing decision making in coalition formation algorithms, generating multiple resource sub-graphs in the topology construction phase, and exercising task generation by varying the number of required resources required per task.

All experiments simulate a system of one million nodes. Each Chord cluster is given 12 bits of address space, allowing for 4096 agents per cluster, with a minimum of 244 clusters for a system of one million agents.

5.3. Response Variables

The objective of these experiments is to evaluate the effectiveness of the DLoE heuristic algorithm against the baseline uniform (CRP) and optimal (CLoE) algorithms. Primary measures of this effectiveness are the agent solicitation miss rate, the sustained workload performance of the system, and the global balanced utilization of resources.

6. Results and Analysis

This analysis evaluates the effectiveness of the DLoE algorithm by comparing its results to the CRP and CLoE algorithms. The CRP algorithm serves as a baseline, and the CLoE algorithm forms a work optimal algorithm.

Table 2 shows the work throughput of the three algorithms on the critically-loaded scenario across the different task synchrony levels. The results are organized by algorithm type and task synchrony. Task synchrony is the number of time steps between each synchronization barrier, with a larger value indicating that the task reaches a barrier less often. The CLoE algorithm serves as the theoretical best performance for the LoE approach, and DLoE tracks those results competitively for each experiment. The CRP algorithm suffers from using a random decision maker algorithm to choose agents to join tasks, and even though it generates a more uniform distribution of tasks to agents, it neglects the priorities of each task. This results in agents maintaining a uniform number of tasks, however there is more contention in executing tasks on those agents, and the overall work throughput suffers.

Figure 3 shows the work throughput of the three algorithms on the over-loaded scenario with task synchrony disabled. Both the CLoE and DLoE algorithms approach the practical maximum of one million units of work executed per unit time, which matches the best scenario work creation rate. The CRP algorithm is unable to reach this milestone. This is one of the benefits of both the CLoE and DLoE algorithms: they try to allocate tasks to agents that most minimally meet task resource requirements. Without any heuristic, the CRP algorithm simply accepts the first agent that can satisfy a task's requirements, regardless of the excess resource amounts possessed by the agent.

Figure 4 shows the number of tasks allocated per agent for the three algorithms in a critically-loaded system with synchrony disabled. Agents in the systems with the CLoE and DLoE algorithms have a lower number of tasks per agent as a result of the higher overall work throughput. Since tasks complete more quickly, the agents are able to satisfy the incoming workload more easily, and thus have fewer tasks. The CRP algorithm has a lower work execution rate, and therefore maintains more tasks per agent. This continues until a saturation point is reached wherein the randomness of the CRP coalition formation algorithm eventually provides enough tasks to agents with lower amounts of resources to meet the incoming workload.

Table 2

Impact of Task Synchrony on Work Throughput. Shown are work throughput mean (standard deviation) for critically-loaded (1000 tps) systems of varying task synchrony. Task synchrony is the number of steps between each barrier synchronization point, with higher values meaning the barrier is met less often.

Synchrony Level	**DLoE**	**CLoE**	**CRP**
None	876779.98 (2473.38)	877000.25 (463.76)	856961.36 (5824.41)
1	867308.54 (4246.03)	877024.73 (471.87)	631134.59 (2414.76)
5	866507.99 (3521.34)	876975.48 (458.81)	631461.40 (3475.70)
10	870917.79 (2724.06)	877013.30 (457.48)	636517.08 (4115.23)
25	872776.62 (1687.70)	877019.55 (446.20)	649992.75 (3316.01)
50	873027.80 (3166.34)	876959.47 (449.23)	678106.20 (5013.18)

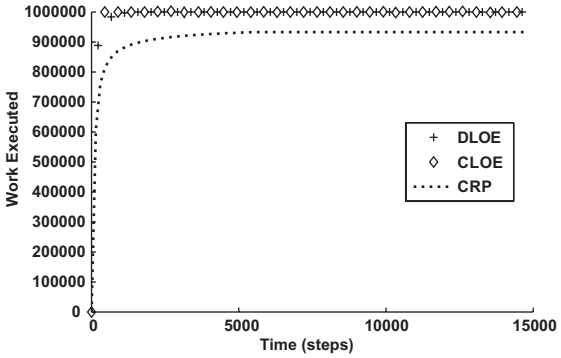

Fig. 3. Comparison of the three algorithms in an over-loaded system (1500tps) with synchrony disabled. The work throughput of the CLoE and DLoE is comparable, whereas the CRP algorithm performs noticeably worse (ANOVA p=0).

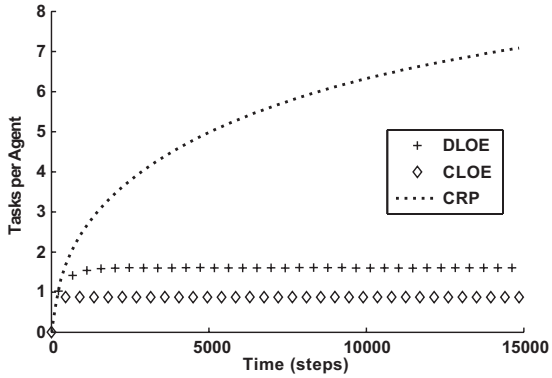

Fig. 4. Comparison of the three algorithms in a critically-loaded system (1000tps) with synchrony disabled. The number of tasks per agent for the CRP algorithm eventually stabilizes at a significantly higher value than the CLoE or DLoE algorithms.

Table 3 shows the impact of the update interval on the mean TPP per RC-Chord level. The results show a strong dependence on the update interval, revealing the impact of rapid task creation and TPP caching in large-scale coalition formation. For systems with one million agents, over 977,000 reside in level three. For the critically-loaded DLoE systems shown in Table 3, level one shows the highest mean variance against the baseline level three due to its small size. Reducing the frequency of DLoE updates reduces the accuracy of TPP balancing because the coalition formation algorithm is forced to use TPP data that has become unrepresentative of the system's current state. Tuning the update frequency significantly improves the coalition formation algorithm accuracy at the expense of increased bandwidth consumption.

Under significant loading, the task duration for the DLoE and CLoE algorithms resembles Figure 5 for all levels of task synchrony. The data show high priority tasks complete with shorter mean duration than lower priority tasks. The excessive loading in these experiments creates a higher number of tasks per agent, and therefore the task scheduler relies primarily on priority to choose which task to execute at each time step.

For under-loaded systems, task durations remain relatively uniform across all task priorities. This occurs because agents in those systems are able to meet the incoming workload, and so the scheduler is rarely forced to choose which task to execute based solely on priority.

A notable exception to this trend is the CRP algorithm, which poorly places its tasks in all cases, leading to a high standard deviation for the number of tasks per agent. This trend is shown in Figure 6. Under the CRP algorithm, many agents are overloaded, while others have zero load. This leads to bottlenecks at those higher loaded agents, creating a net reduction in the work executed per unit time, and forcing

Table 3
Mean TPP per agent at each level measured against the DLoE update interval.

	Update Interval (time steps)					
RC-Chord Level Number	0	1	5	10	25	50
Level 1	5.15	5.28	10.22	15.59	54.87	218.79
Level 2	4.46	4.61	7.65	10.71	25.13	48.03
Level 3	4.22	4.38	6.12	7.74	15.92	28.76

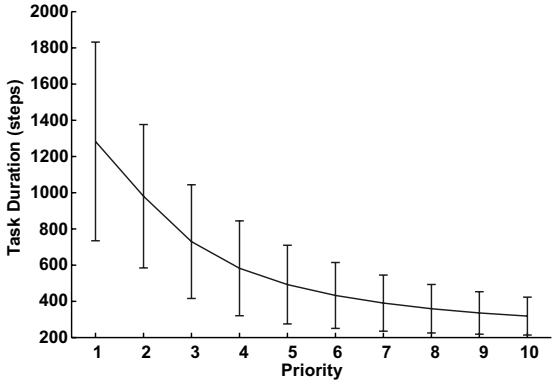

Fig. 5. Task duration versus priority for critically-loaded (1000tps) and over-loaded systems (1500tps). The trends are identical for both the CLoE and DLoE coalition formation algorithms. The duration of tasks reduces with increased priority when the task scheduler is forced to decide based on priority.

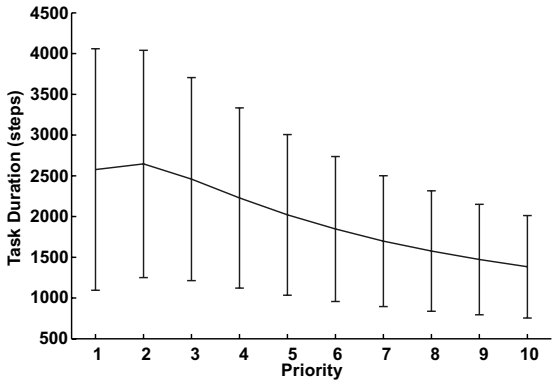

Fig. 6. Task duration versus priority for a critically-loaded system (1500tps) using the CRP coalition formation algorithm. The poor assignment of agents to task coalitions yields bottlenecks in systems with task synchrony, causing delays in processing and eliminating the effectiveness of the task scheduling algorithm.

critically-loaded systems to become unable to meet the incoming task workload.

Figure 7 shows the statistical comparison of the work throughput versus task synchrony for critically-loaded experiments where synchrony is disabled, and Figure 8 shows the same scenario with task synchrony enabled. The difference in performance for both cases between the CRP algorithm and the other two is substantially different, as the CRP performs far worse than the other two algorithms. Both the CLoE and DLoE algorithms have similar median work throughput (within one standard deviation of one another), but the CLoE algorithm yields lower standard deviation with fewer outliers due to higher coalition formation quality as a result of global knowledge.

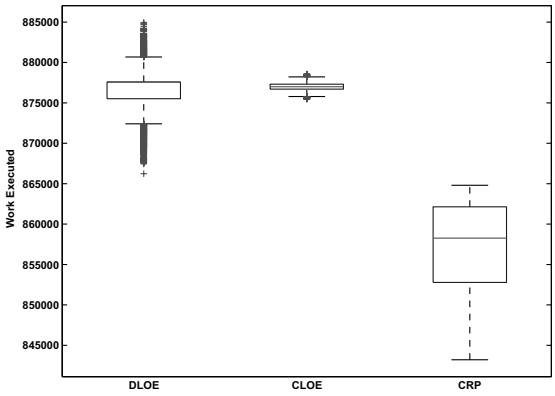

Fig. 7. Boxplot for work throughput versus task synchrony in critically-loaded experiments where synchrony is disabled. The median performance range between the DLoE and CLoE algorithms is similar, however the standard deviation and number of outliers for DLoE is greater than that of the CLoE algorithm. The CRP algorithm yields lower performance than both other algorithms.

The result of these experiments demonstrates that task synchrony plays an important part in the overall performance of coalitions generated using these different algorithms. The CRP algorithm, in particular, suffers from poor performance as a result of its coalition formation process. In addition, the CRP algorithm suffers from the highest miss rate of the algorithms considered, reaching over 80% in some experiments.

The CLoE demonstrates the best overall sustained task execution rate, as well as the lowest standard de-

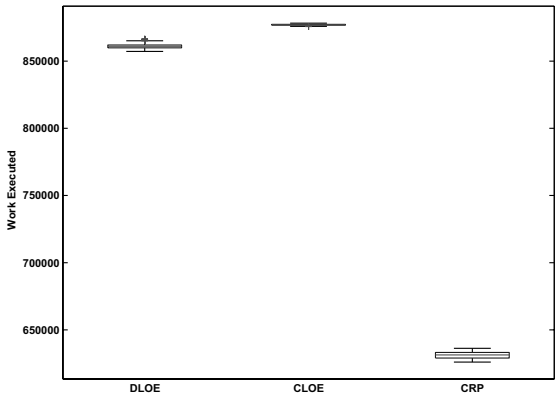

Fig. 8. Boxplot for work throughput versus task synchrony in critically-loaded experiments where synchrony is enabled. The CLoE and DLoE algorithms perform far better than the CRP algorithm due to coalition formation heuristics.

viation. The DLoE algorithm is statistically similar in mean work throughput, with slightly higher variance. The DLoE algorithm successfully tackles the challenge of task synchrony, and allow the task scheduler to make good decisions that reduce the durations of tasks with higher priorities during high workloads.

7. Conclusion

As highly networked entities seek to leverage the power and scale of P2P systems and their data, the difficulty of efficiently sharing the capabilities and assets of the attached systems becomes critically important. This paper presents the Distributed Likelihood of Execution algorithm, which uses the cooperative coalition formation problem as a framework for tasking agents. The DLoE algorithm relies upon the facilities of the RC-Chord structured overlay, specifically the ability to allocate agents into clusters organized by resource or capability. The DLoE algorithm stores task information for agents with each cluster, and passes this information up the network hierarchy to construct a general view of the task loading on each sub-graph. The DLoE algorithm uses this information to decide how far, and in which direction, to descend in search of agents to satisfy task allocation requests. Simulation results indicate that our distributed algorithm performs nearly as well as an omnipotent centralized optimal algorithm, and significantly better than the baseline algorithm.

Potential improvements of the DLoE algorithm include building a new strategy for allocating the number and spacing of the resource intervals for the DLoE algorithm. These intervals are currently static, and only configurable before runtime. This can be improved by incorporating a distributed learning algorithm to tune the number and range of the DLoE tracking intervals at runtime. In addition, work contributed by different agents should ideally contribute to different pools of work within the task. At present, units of work are considered equal, and although this satisfies the target deployment, this may not be the case in all applications.

References

[1] S. Abdallah and V. Lesser. Organization-based coalition formation. In *AAMAS 2004: Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems*, pages 1296–1297, Washington, DC, USA, 2004. IEEE Computer Society.

[2] S. Abdallah and V. Lesser. Organization-based cooperative coalition formation. In *Proceedings of the IEEE/WIC/ACM International Conference on Intelligent Agent Techonology, IAT*, pages 162–168, China, September 2004.

[3] M. Afsharchi, J. Denzinger, and H. Far. Enhancing communication with groups of agents using learned non-unanimous ontology concepts. *Web Intelligence and Agent Systems*, 7:107–121, 2009.

[4] L. O. Alima, A. Ghodsi, and S. Haridi. A framework for structured peer-to-peer overlay networks. *Lecture Notes in Computer Science*, 3267:223–250, 2004.

[5] A. Bonifati, P.K. Chrysanthis, A.M. Ouksel, and K.U. Sattler. Distributed Databases and Peer-to-Peer Databases: Past and Present. *ACM SIGMOD Record*, 27(1):5–11, 2008.

[6] I. Clarke, O. Sandberg, B. Wiley, and T. W. Hong. Freenet: A distributed anonymous information storage and retrieval system. *Lecture Notes in Computer Science*, 2009:46–63, 2001.

[7] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms*. The MIT Press, 2nd edition, 2001.

[8] J. Frankel and T. Pepper. Gnutella. Internet, 2000.

[9] N. Fukuta and T. Ito. Fine-grained efficient resource allocation using approximated combinatorial auctions: A parallel greedy winner approximation for large-scale problems. *Web Intelligence and Agent Systems*, 7:43–63, 2009.

[10] B. P. Gerkey. *On multi-robot task allocation.* PhD thesis, University of Southern California, August 2003.

[11] B. P. Gerkey and M. J. Mataric. A formal analysis and taxonomy of task allocation in multi-robot systems. *International Journal of Robotics Research*, 23:939–954, September 2004.

[12] C. V. Goldman and S. Zilberstein. Decentralized control of cooperative systems: Categorization and complexity analysis. *Journal of Artificial Intelligence Research*, 22:2004, 2004.

[13] A. Gupta, B. Liskov, and R. Rodrigues. Efficient routing for peer-to-peer overlays. In *NSDI 2004: Proceedings of the First Symposium on Networked Systems Design and Implementation*, pages 9–9, Berkeley, CA, USA, 2004. USENIX Association.

[14] I. Gupta, K. Birman, P. Linga, A. Demers, and R. van Renesse. Kelips: Building an efficient and stable P2P DHT through increased memory and background overhead. In *Proceedings of the Second International Workshop on Peer-to-Peer Systems (IPTPS '03)*, 2003.

[15] N. J. A. Harvey, M. B. Jones, S. Saroiu, M. Theimer, and A. Wolman. Skipnet: a scalable overlay network with practical locality properties. In *USITS 2003: Proceedings of the Fourth USENIX Symposium on Internet Technologies and Systems*, pages 9–9, Berkeley, CA, USA, 2003. USENIX Association.

[16] M. Jelasity, A. Montresor, G. P. Jesi, and S. Voulgaris. The Peersim simulator. `http://peersim.sf.net`.

[17] D. Karrels, G. Peterson, and B. Mullins. RC-Chord: Resource clustering in a large-scale hierarchical peer-to-peer system. In *Military Communications Conference, 2009. MILCOM 2009*, Boston, MA, October 2009.

[18] D. Karrels, G. Peterson, and B. Mullins. Structured P2P technologies for distributed command and control. *Peer-to-Peer Networking and Applications*, 2(4): 311–333, 2009.

[19] S. P. Ketchpel. Forming coalitions in the face of uncertain rewards. In *National Conference on Artificial Intelligence*, pages 414–419, 1994.

[20] V. Kumar, A. Grama, A. Gupta, and G. Karypis. *Introduction to Parallel Computing: Design and Analysis of Algorithms*. Benjamin-Cummings Publishing Co., Inc., Redwood City, CA, USA, 1994.

[21] J. Li, J. Stribling, R. Morris, and M. F. Kaashoek. Bandwidth-efficient management of dht routing tables. In *NSDI 2005: Proceedings of the Second Symposium on Networked Systems Design & Implementation*, pages 99–114, Berkeley, CA, USA, 2005. USENIX Association.

[22] E. J. L. Lu, Y. F. Huang, and S. C. Lu. Ml-chord: A multi-layered p2p resource sharing model. *Journal of Network and Computer Applications*, 3:32, May 2009.

[23] MessageLabs. Messagelabs intelligence: Q2/June: Cutwail′s bounce-back; instant messages can lead to instant malware. Technical report, MessageLabs, June 2009.

[24] D. C. Montgomery. *Design and Analysis of Experiments*. Wiley, December 2004.

[25] D. Ouelhadj, P. I. Cowling, A. Meisels, and S. Petrovic. Contract net protocol for cooperative optimisation and dynamic scheduling of steel production. *Journal of Advanced Engineering Informatics*, 18:3, July, 2004.

[26] D. V. Pynadath and M. Tambe. Multiagent Teamwork: Analyzing the Optimality and Complexity of Key Theories and Models, 2002. *AAMAS 2002: First International Joint Conference on Autonomous Agents and Multi-Agent Systems*, pages 873-880, Bologna, Italy.

[27] T. Rahwan. *Algorithms for Coalition Formation in Multi-Agent Systems*. PhD thesis, University of Southampton, August 2007.

[28] S. Rhea, D. Geels, T. Roscoe, and J. Kubiatowicz. Handling churn in a dht. In *ATEC 2004: Proceedings of the Annual Conference on USENIX Annual Technical Conference*, pages 10–10, Berkeley, CA, USA, 2004. USENIX Association.

[29] T. W. Sandholm. An implementation of the contract net protocol based on marginal cost calculations. In *Proceedings of the Twelfth International Workshop on Distributed Artificial Intelligence*, pages 295–308, Hidden Valley, Pennsylvania, 1993.

[30] O. Shehory and S. Kraus. Coalition formation among autonomous agents: Strategies and complexity (preliminary report). In C. Castelfranchi and J.-P. Müller, editors, *From Reaction to Cognition — Fifth European Workshop on Modelling Autonomous Agents in a Multi-Agent World, MAAMAW-93 (LNAI Volume 957)*, pages 56–72. Springer-Verlag: Heidelberg, Germany, 1995.

[31] O. Shehory and S. Kraus. Task allocation via coalition formation among autonomous agents. In *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence (IJCAI-95)*, pages 655–661, Montréal, Québec, Canada, 1995.

[32] O. Shehory and S. Kraus. Methods for task allocation via agent coalition formation. *Artificial Intelligence*, 101(1-2):165–200, 1998.

[33] R. G. Smith. The Contract Net Protocol: High-Level Communication and Control in a Distributed Problem Solver. *IEEE Transactions on Computers, pages 1104–1113, 29:12, December 1980*.

[34] I. Stoica, R. Morris, D. Karger, F. Kaashoek, and H. Balakrishnan. Chord: A scalable peer-to-peer lookup service for internet applications. In *Proceedings of the 2001 ACM SIGCOMM Conference*, pages 149–160, 2001.

[35] P. Strong. Enterprise grid computing. *Queue*, 3(6):50–59, 2005.

[36] C. Theocharopoulou, I. Partsakoulakis, G. A. Vouros, and K. Stergiou. Overlay networks for task allocation and coordination in dynamic large-scale networks of cooperative agents. In *AAMAS 2007: Proceedings of the Sixth International Joint Conference on Autonomous Agents and Multiagent Systems*, pages 1–8, New York, NY, USA, 2007. ACM.

[37] E. Winter. *The Handbook of Game Theory*, The Shapley Value, pages 2026–2052. North-Holldand, 2002.

[38] Y. Xu, P. Scerri, B. Yu, S. Okamoto, M. Lewis, and K. Sycara. An integrated token-based algorithm for scalable coordination. In *AAMAS 2005: Proceedings of the Fourth International Joint Conference on Autonomous Agents and Multiagent Systems*, pages 407–414, New York, NY, USA, 2005. ACM.

[39] X. Xu, L. Chen, and P. He. A novel ant clustering algorithm based on cellular automata. *Web Intelligence and Agent Systems*, 5:1–14, 2007.

[40] M. Yokoo, E. H. Durfee, T. Ishida, and K. Kuwabara. The distributed constraint satisfaction problem: Formalization and algorithms. *Knowledge and Data Engineering*, 10(5):673–685, 1998.