The Price of Anarchy: Centralized versus Distributed Resource Allocation Trade-offs

Jinhong K. Guo[1], Alexander Karlovitz[2], Patrick Jaillet[3] and Martin O. Hofmann[1]
[1]*Lockheed Martin Advanced Technology Laboratories, 3 Executive Campus, Suite 600, Cherry Hill, NJ 08002, U.S.A.*
[2]*Department of Mathematics, Rutgers University, Piscataway, NJ 08854, U.S.A.*
[3]*Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, Cambridge, MA 02139, U.S.A.*

Keywords: Resource Allocation, Resource Optimization, Auction-based Approach, Decentralized Resource Allocation.

Abstract: Optimizing decision quality in large scale, distributed, resource allocation problems requires selecting the appropriate decision network architecture. Such resource allocation problems occur in distributed sensor networks, military air campaign planning, logistics networks, energy grids, etc. Optimal solutions require that demand, resource status, and allocation decisions are shared via messaging between geographically distributed, independent decision nodes. Jamming of wireless links, cyber attacks against the network, or infrastructure damage from natural disasters interfere with messaging and, thus, the quality of the allocation decisions. Our contribution described in the paper is a decentralized resource allocation architecture and algorithm that is robust to significant message loss and to uncertain demand arrival, and provides fine-grained, many-to-many combinatorial task allocation. Most importantly, it enables a conscious choice of the best level of decentralization under the expected degree of communications denial and quantifies the benefits of approximating status of peer nodes using proxy agents during temporary communications loss.

SCITEPRESS SCIENCE AND TECHNOLOGY PUBLICATIONS

1 OVERVIEW

Teams of autonomous sensor assets are expected to self-organize to perform a variety of spatially distributed sensing tasks, such as search and rescue, surveillance and environmental monitoring. Challenging characteristics of these applications include incomplete and uncertain status (current tasks, asset status, etc.), time-sensitive objectives (re-direction during mission execution, failed asset replacement), and limited and varying communication topologies between assets and command and control nodes. Decentralized command and control promises greater agility and resilience in communication contested environment. To decentralize safely, one has to understand the trade-offs between agility and quality.

The contribution of this work is in characterizing the trade-off of decentralization using a robust, probabilistic, auction-based resource allocation architecture with a variable number of distributed auctioneer nodes, each responsible for optimally allocating a subset of tasks to a subset of assets. We use a many-to-many, combinatorial task allocation and scheduling algorithm that is robust to uncertain future demand and uses expressive cost formulations to model mobile sensors with variable costs that depend in travel distance and sensor quality that varies with environmental conditions. Our algorithm is robust against communication interruptions and minimizes ripple effects due to unexpected tasks and resource failures. To further reduce the effect of intermittent communication failures, we employ proxy agents which bid on behalf of their remote assets. Our experiments show that proxy agents improve allocation success rates. Due to the complexity of our problem, unmatched in other published work, a rigorous and tractable mathematical analysis is impossible. Instead, we derived theoretical bounds on a simplified version and validated our algorithm's performance.

2 RESOURCE ALLOCATION

We assume m heterogeneous agents, each representing an asset that can possess multiple capabilities. n tasks arrive dynamically over time and are located in a given geographical region G. In each area G, one of the agents assumes the role of auctioneer. Upon

Guo, J., Karlovitz, A., Jaillet, P. and Hofmann, M.
The Price of Anarchy: Centralized versus Distributed Resource Allocation Trade-offs.
DOI: 10.5220/0007345701460153
In *Proceedings of the 11th International Conference on Agents and Artificial Intelligence (ICAART 2019)*, pages 146-153
ISBN: 978-989-758-350-6
Copyright © 2019 by SCITEPRESS – Science and Technology Publications, Lda. All rights reserved

arrival of a task, the auctioneer announces the task and agents bid to perform the task. The auctioneer determines the winner and assigns the task to one or several bidders. A task is assigned to multiple agents when the task requires the combined capabilities of multiple agents. The completion of an assigned task incurs a cost and earns a reward for the agent. The goal is to maximize net revenue, the sum of reward minus cost for all agents and tasks, over time.

Task: A task requires a number of capabilities to be fulfilled by one or more agents. Each capability has an associated timeline and a minimum proficiency, which allowes suboptimal but acceptable use of less capable resources. For shared capabilities, the proficiency requirement is used to indicate the minimum capacity needed, e.g., the portion of bandwidth needed from a communication channel. Therefore, we represent task j as a tuple $(t_j, l_j, req_j, d_j, r_j, Pr_j, T_j, T_{0j})$, where, $t_j \in \mathbb{R}_+$ is the release time of task j and $l_j \in G$ is the location of task j. $req_j \in [0,1]^{3\times k}$ is a $3 \times k$ matrix, describing the required capabilities for performing task j. k is the total number of capabilities defined for the application, and each specific capability is identified by its column position in the matrix. The values in the first row indicates the required proficiency level for the required capabilities. The second row is a boolean, 0 (false) or 1 (true), that indicates if the capability is shared. If it is shared, then the corresponding value in the third row represents the minimum required capacity level. $d_j \in \mathbb{R}_+^k$ is a k-vector of time durations for task j along its k dimensional requirements req_j. $r_j \in \mathbb{R}_+$ is the reward for the completion of task j. $Pr_j \in \mathbb{R}_+$ is the priority of task j, $T_j \in \mathbb{R}_+^k$ is a k-vector of deadlines before which task j's requirements must be assigned, $T_{0j} \in \mathbb{R}_+^k$ is a k-vector of the requested earliest starting times for task j's required capabilities.

Agent: m agents, each possessing a set of capabilities, are assigned to one or more groups. An agent is defined by (l_i, g_i, cap_i, s_i), where $l_i \in G$ is the location of agent i, g_i is the group identifier of agent i, and $cap_i \in [0,1]^{3\times k}$ is a $3 \times k$ matrix with values of 0 to 1 describing the qualification of agent i along k capability dimensions, similar to the capability requirement in the task representation, with the third row represents an agent's capability capacity. s_i is the speed of the agent.

Objective: The completion of task j requires that the set of agents $i_1, i_2, i_3, \ldots$ assigned to it collectively have qualifications that meet req_j. The objective of the optimization problem is to maximize total net revenue (reward minus cost) over the n tasks. The resource allocation problem can be written as a mixed-integer program:

$$\max_{\mathbf{x},\tau} \sum_{j=1}^{n}\sum_{i=1}^{m}\sum_{p=1}^{k}\sum_{q=1}^{k} R_{ij}^{pq} x_{ij}^{pq}$$

$$s.t. \mathbf{H}(\mathbf{x},\tau) \leq \mathbf{d}$$

$$\mathbf{x} \in \{0,1\}^{n,m,k,k}, \tau \in \{\mathbb{R}^+\}^{n,m,k,k}$$

where $\mathbf{x} \in \{0,1\}^{n,m,k,k}$, is a set of binary decision variables x_{ij}^{pq} with $x_{ij}^{pq} = 1$ indicates that the capability p of agent i will serve the capability requirement q of task j. $\tau \in \{\mathbb{R}^+\}^{n,m,k,k}$ is the set of real-positive decision variables that indicates when agent i capability p will serve task j requirement q. R_{ij}^{pq} is the revenue of serving task j requirement q with agent i capability p. Assuming the total reward associated with the completion of task j is r_j and the cost of performing task j by agent i is c_{ij}, $R_{ij}^{pq} = r_{ij}^{pq} - c_{ij}^{pq}$ with r_{ij}^{pq} and c_{ij}^{pq} defined as a fraction of r_j and c_{ij}. $\mathbf{H}(\mathbf{x},\tau)$ defines a set of linear and possibly non-linear constraints that captures transition dynamics, resource, spatial and temporal limitation, etc., that are bounded by some criteria $\mathbf{d}$. The constraint functions depend on both of the decision variables $\mathbf{x}$ and τ, making this mixed-integer problem even more difficult to solve.

Our algorithm enforces both soft and hard constraints. The hard constraints include the following.

Capability: an agent is only considered for a task if it has the required capability, sufficient proficiency, and the required capacity, if it is a shareable capability. Therefore, this defines three constraints, such that

$$(req_j^1 \wedge I_q) \cdot (cap_i^1 \wedge I_p) > 0$$

$$(req_j^1 \wedge I_q) - (cap_i^1 \wedge I_p) \leq 0$$

$$(req_j^3 \wedge I_q) - (cap_i^3 \wedge I_p) \leq 0 \; if \; req_j^{2,q} = 1$$

req_j^c is the c^{th} row of matrix req_j and cap_i^c is the c^{th} row of matrix cap_i. I_d is a vector with the d_{th} element being 1 while the other elements are 0. $q_j^{2,q}$ is the q^{th} element of the second row of req_j.

Temporal and spatial: an agent can perform different tasks concurrently only if these tasks (j_1 and j_2) are geographically collocated (within a *radius*), i.e.,

$$\|l_{j_1} - l_{j_2}\| \leq radius$$

An agent can bid for a task only if it can reach the requested location before the task starts, i.e., there is sufficient time (T) between the schedule of j_1 and j_2 by agent i. Note that T is determined by both the task locations and the speed of the agent.

$$|\tau_{\mathbf{i,j1}} - \tau_{\mathbf{i,j2}}| > T, \; if \|l_{j1} - l_{j2}\| > radius$$

Dependency (hard): if some of the requirements (possibly across multiple tasks) are deemed dependent, all those capability requirements must be allocated together or none at all. Let Ψ be the set of pairs (j,q) (task j, capability requirement q) defining such a dependency set,

$$\sum_{i=1}^{m}\sum_{p=1}^{k} x_{ij}^{pq} = \sum_{i=1}^{m}\sum_{p=1}^{k} x_{ij'}^{pq'} \quad \forall pairs\ (j,q),(j',q') \in \Psi$$

Same Agent: a set of capabilities (possibly across multiple tasks) has to be performed by the same agent. Let Ψ define the set of pairs (j,q) (task j, capability requirement q) to be performed by the same agent,

$$\sum_{i}\sum_{p}\sum_{(j,q)\in\Psi} x_{ij}^{pq} \leq 1$$

Soft constraints are implemented as adjustments to the revenue function. To enforce a soft constraint, the revenue is adjusted to be

$$R_{ij}^{pq} = r_{ij}^{pq} - c_{ij}^{pq} + \Delta_{ij}^{pq}$$

where Δ_{ij}^{pq} is calculated depending on the constraints. The soft constraints are designed for:

Honoring Priority: higher priority tasks have a better chance of being assigned by increasing revenue. Therefore, Δ_{ij}^{pq} is chosen to be proportional to a level of priority Pr_j, i.e., $\Delta_{ij}^{pq} \propto Pr_j$

Anticipating Future Needs: reserve assets for future high priority tasks based on an estimated distribution of future task arrivals. The adjustment on the revenue R_{ij}^{pq} uses an expectation $E(y)$ of future high priority tasks, i.e., $\Delta_{ij}^{pq} \propto -E(y)$

Minimizing Ripple Effects: cost of anticipated additional allocation changes due to the allocation change under consideration. Assuming E(z) quantifies the anticipated changes, $\Delta_{ij}^{pq} \propto -E(z)$

Dependencies (soft): increasing the chances of the assignment of associated capabilities if some of them are assigned i.e., $\Delta_{ij}^{pq} \propto \sum_i \sum_p \sum_{(j,q)\in\Psi} x_{ij}^{pq}$

As illustrated above, the mixed-integer problem is very difficult to solve. We employ a single-round auction where one or more auctioneers announce tasks to their respective groups of agents, agents with appropriate capabilities bid (whether they are already busy or not), the auctioneers optimize many-to-many assignments of task capabilities to agent capabilities, and announce the allocation. We use a greedy winner determination algorithm that is based on a constrained clustering algorithm (Tung et al., 2001), similar to the greedy approach developed in (Greene and Hofmann, 2006), but allowing resource sharing over time. Each agent defines a cluster of its capabilities. An additional cluster consists of all the unallocated capability requirements (of all tasks). Task requirements are moved among agent clusters based on revenue improvement, but honoring constraints, until a local optimum is achieved.

Group membership is dynamically determined by geographic proximity within the range of communication. When an agent moves beteween groups, it gives up its unfulfilled previous assignments so that the previous auctioneer can attempt to reassign them to other agents in its group.

3 PROXY AGENTS

Limited ad-hoc and varying communication topologies between assets (agents) and command and control nodes (auctioneer) significantly affect resource allocation performance for both centralized and distributed architectures. To lessen the effect of failed communication preventing agents from participating in an auction, we employed proxy agents. The proxy agents, which reside within the auctioneer, keep status of the agents they represent. The auctioneer tracks the connectivity status of the real agents such that if an agent has repeatedly failed to communicate with the auctioneer, its proxy agent will not be allowed to participate in future auctions until the agent is heard from again. In the experiments described in this paper, the agents send pings every $t_{wait} = 100$ ms. The auctioneer counts every t_{wait} ms if it does not receive a ping from an agent. If this count gets above $maxCnt = 20$ for some agent, the auctioneer does not include that proxy agent in any bids until it receives another ping.

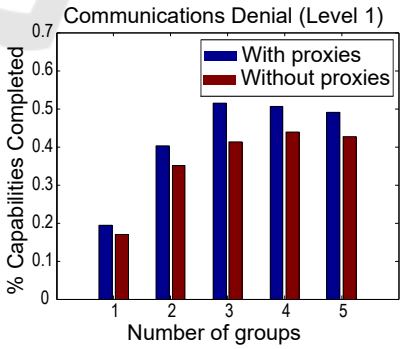


Figure 1: Capability allocation success with and without proxy agents: number of agents = 30, number of pre-assigned tasks = 30, number of new tasks = 30, Task duration = 1000ms, New task arrival Poisson mean = 50ms.

Experiment results have shown (Figure 1) that when probability of communication failure is high, proxy agents improve the performance of task allocation rates. This is implemented by incorporating

the expectation of successful communication into the utility function (for simplicity denoted by R, omitting the indices for tasks and agents): $R' = p \cdot R$, where p is the probability that agent i will receive the assignment should it win the bid for performing capability j. p is calculated using the latest n pings between the proxy agent and its agent. $p = \sum_{k=1}^{n} \alpha_k \times p_k, \quad \sum_{k=1}^{n} \alpha_k = 1$, with $\alpha_{k+1} = f \times \alpha_k, k = 1, \ldots, n-1$ to favor the lastest ping.

$$p_k = \begin{cases} 1 & \text{if } k^{th} \text{ ping succeeded} \\ 0 & \text{otherwise} \end{cases}$$

In our experiments, $\alpha_1 = 0.2$, $f = 0.8$, and $n = 20$.

4 THEORETICAL BOUNDS

Due to the complexity of the problem addressed in this paper, a mathematical formulation is impossible except for a simplified version which we used to derive theoretical bounds and validate algorithms. Theoretical bounds help us understand empirical results and design systems with consideration of performance bounds.

First, given some task arrival intervals and task durations over a given geographical area, we would like to know the smallest number of agents needed for the system to work properly. We consider the simplest possible analytical framework within our overall problem setting.

In particular we will assume that: (1) Task requirements (**req**$_j$'s) and agent qualifications (**cap**$_i$'s) are unidimensional (i.e., $k = 1$). (2) Release times of the tasks are random and follow a Poisson process of parameter λ. (3) The duration of the tasks are i.i.d. exponential with parameter μ (Model 1), or uniform over the interval $[0, 2/\mu]$ (Model 2) [expected task duration is $1/\mu$ under both models]. (4) There are m agents, initially distributed at random over a unit square. (5) At any time, an agent is either busy or available. (6) Upon arrival of a task, if no agents are available then the task is dropped, otherwise it is assigned to one of the free agents uniformly at random. (7) A free agent, upon assignment of a task, can start working on it immediately, irrespective of his/her location. (8) We observe the system over a long period of time so that the observed number of tasks n is very large (the system is in a "steady-state").

Result 1: The probability that all agents are busy when a new task arrives is given by [1]

$$\frac{(\lambda/\mu)^m/m!}{\sum_{i=0}^{m} (\lambda/\mu)^i/i!}$$

For a given λ and μ, this formula allows us to find the minimum number of agents so that the fraction of tasks being allocated is above a given level of service. Result 1 remains valid, under some scaling of the parameters, e.g., for a given ratio λ/μ, the region can be any compact region, and not just the unit square. More interestingly, Result 1 can also be shown to remain valid even when the duration of serving a task also includes moving to the location of the task, as long as it is driven by independent (between tasks) random variables following any general distribution. with mean $1/\mu$ (not necessarily exponential). For example, this would be the case when the task is allocated at random uniformly among any free agents, irrespective of their geographical location. In that case, under the assumption that the overall region of interest is the square $[0, l]^2$, and assuming that any agents move at constant speed s, the previous formula becomes

$$\frac{(\lambda(0.52l/s + 1/\mu))^m/m!}{\sum_{i=0}^{m} (\lambda(0.52l/s + 1/\mu))^i/i!}$$

Result 1 provides a valid upper bound if an allocation based on shortest distance will lead to a smaller number of agents for the same level of service.

Result 2 (first generalizations): Now assume that instead of being dropped, a new task, finding all agents busy, is put on a waiting list and then assigned to the first available agent. Then under Model 1, the system will reach steady-state if $\lambda/m\mu < 1$. We have:

a) The probability p_j that there are j tasks in the system at any time is given by

$$p_j = \begin{cases} p_0(\lambda/\mu)^j/j! & \text{for } 0 \le j \le m-1 \\ p_0(\lambda/\mu)^j/(m^{j-m}m!) & \text{for } j \ge m \end{cases}$$

where

$$p_0 = [\sum_{j=0}^{m-1} (\lambda/\mu)^j/j! + (\lambda/\mu)^m/(m!(1-\lambda/m\mu))]^{-1}$$

b) The average number of tasks waiting in the system is:

$$\overline{n} = p_0(\lambda/\mu)^m(\lambda/m\mu)/(m!(1-\lambda/m\mu)^2)$$

c) The average waiting time of a task is

$$\overline{w} = \overline{n}/\lambda = p_0(\lambda/\mu)^m(\lambda/m\mu)/(m!(1-\lambda/m\mu)^2)/\lambda$$

[1] Due to space limitation, a detailed proof is omitted. Note that this is closely related to the classical Erlang's loss formula, developed during the first decade of the 20th century by Erlang, a Danish engineer who was working on sizing up telephone systems.

From b), for a given λ and μ, we can choose m so that the average number of tasks waiting in the system at any time is below a desired level and/or the average waiting time of a task is below a prescribed level.

Result 3 (second generalizations): We now consider the case where the allocation of a new task to a given agent is done proportionally to the shortest distance among all free agents. A new task, finding all agents busy, is put on a waiting list and then assigned to the closest available agent. In that case,

d) Under heavy-traffic approximation, when $\lambda/m\mu \to 1$, the average waiting time of a task is given by

$$\overline{w} \approx (1/\lambda + \lambda\sigma^2/m)/2(1-\lambda/m\mu)$$

where σ^2 is the variance of the service time under the chosen policy (i.e. travel time plus execution time of the task).

Assuming a uniform spatial distribution of free agents and unit speed over the unit square, we would have $\sigma^2 \approx 0.07$. The value in d) shows that the average waiting time is dominated by the term $(1-\lambda/m\mu)^{-1}$ as $\lambda/m\mu \to 1$.

We conducted experiments to co-validate the theoretical bounds and our constraint resource allocation algorithm (setting the configuration to match the simple framework).

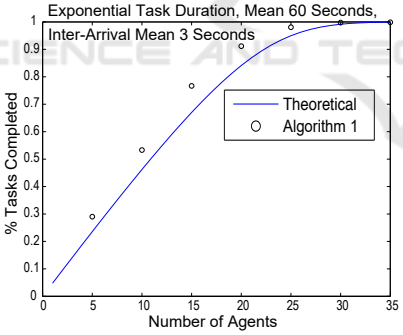
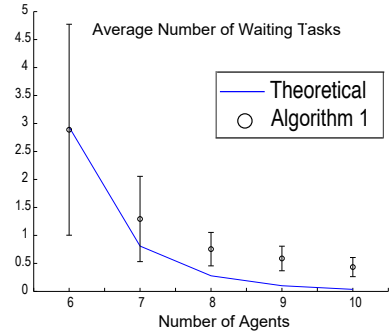


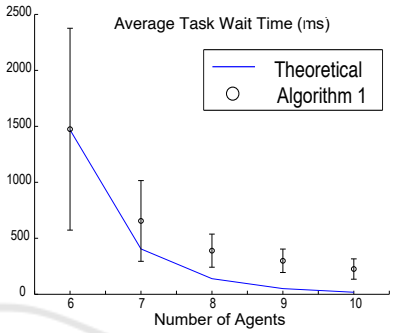
Figure 2: Empirical results and theoretical bound on minimum number agents needed to ensure the fraction of tasks being allocated (i.e. not dropped) is above a given level of service with task duration exponentially distributed with mean 60s; task arrival with Poisson mean 3s and $\lambda/\mu = 20$.

Figure 2 shows results of empirical runs (labeled Algorithm 1) overlaid over theoretical curves of Result 1. As illustrated, the empirical results fit well with the theoretical lower bound. A relaxation of the time threshold in our algorithm implementation after which a task was dropped if not assigned may have contributed to the slightly better performance than the theoretical analysis suggests.

Figure 3 shows the empirical results overlaid over the theoretical curves for Result 2. For empirical data



(a) $\lambda = 1/500, \mu = 1/2500$



(b) $\lambda = 1/500, \mu = 1/2500$

Figure 3: Empirical and theoretical results on (a) average number of waiting tasks; (b) average waiting time per task.

points, the vertical bars show standard deviation about the mean over multiple runs. Note that for the system to reach a steady-state, $\lambda/m\mu < 1$ must hold. Also note that the closer to the boundary condition (m = 1), the larger the standard deviation is.

5 RELATED RESEARCH

Most work on distributed systems focuses on collaboration algorithms and strategies (Spall, 2012), such as routing and positioning UAVs to perform a specific task (Bednowitz et al., 2014). Only a few researchers have investigated the trade-offs of decentralization, (Cicalo et al., 2011), and, in contrast to our work, the performance tradeoff is generally analyzed without considering communications degradation or adversary interference (Tsitsiklis and Xu, 2011). A key reason is that it is extremely difficult to mathematically model the complex aspects of the problem. In this paper, we specifically address communication failure and empirically explore trade-offs.

A number of market-based algorithms for collaborative multi-agent planning have been developed, benefiting from their simplicity and low computation and communication costs ((Zheng and Koenig,

2009)(Zheng and Koenig, 2010)(Hong and Gordon, 2011)). However, most existing market-based algorithms cannot re-evaluate the existing allocations when new tasks need to be inserted (Mauadi et al., 2011). In contrast to algorithms for homogeneous agents (Amador et al., 2014), our algorithm also applies to heterogeneous agents. Uniquely, our agents are modeled by a set of capabilities with individually variable proficiencies. Their bids are relative to spatial and temporal constraints and dependencies of the roles to be fulfilled. Most research considers only assignments of one agent to one task, one agent to multiple tasks (Liu and Shell, 2011), or many agents to one task (Zhang et al., 2012), while our approach handles many-to-many task assignments with different capability mixes with varying proficiencies. Also, unlike most techniques (Roggendorf and Beltran, 2006)(Bonacquisto et al., 2014) the bidding agents in our auction algorithm represent unselfish agents that share the common goal of achieving overall maximum revenue. Our agents do not lower their bids for highly contended resources, e.g., caching (Wang and Martinez, 2015). Instead, our robust winner determination algorithm accounts for such refinements. Similar to the consensus-based approach in Choi et al. (Choi et al., 2009), our approach is robust to network changes, but also adds proxy agents. Unlike more analytical work (Lagoudakis et al., 2004), our algorithm does not provide a guarantee on the quality of its allocations. Instead, we provided theoretical bounds.

6 EMPIRICAL ANALYSIS

The goal of this empirical analysis is to explore the trade-off between centralized and distributed architectures without the simplification that theoretical analysis is forced to make. To achieve this goal, we simulated various experiment settings under different communication conditions. We varied the given maximum number of different capability types M, collectively required by the tasks (and collectively offered by the agents), and randomized task characteristics, the number of groups that the agents are divided into, and varied communication conditions. We implemented agent communications using the Java Agent Development Framework (JADE) (Bellifemine et al., 2007) and developed a module to add interference to the communication between the agents using a communication model described below.

Our assumption is that the farther a message has to travel, the more likely the message is to fail. So is the closer the sender is to a communication jammer. We model communication failures with a two stage Gilbert-Elliott channel (Kong, 2002). The channel either stays put or transitions between a "good" and a "bad" state. We assume that when the channel is in a bad state, the agent cannot send or receive its message. The agent will repeatedly resend the message until either the message is sent successfully or the message times out (upon which it is dropped). In our experiment, in addition to the reliability of the communication channel, we need to consider adversary interference. Therefore, the communication between the two agents will have to be in a compounded "good" state both in terms of the normal communication channels as well as adversary interference. This is illustrated in Figure 4, with green being good state and red being bad state. The transition probabilities α and β are parameterized with distances d_1 and d_2, which are distances to the closest jammer and of the sender-receiver distance, respectively.

Table 1: Exemplar parameters used in defining different communication levels.

Level	a_1	b_1	e_1	f_1	a_2	b_2	e_2	f_2
1	0.8	0.001	0.1	0.001	0.5	0.01	0.5	0.01
2	0.7	0.001	0.2	0.001	0.5	0.01	0.6	0.01
3	0.6	0.001	0.3	0.001	0.4	0.0075	0.6	0.0075
4	0.5	0.001	0.5	0.001	0.3	0.005	0.6	0.005
5	0.5	0.01	0.5	0.01	0.3	0.005	0.7	0.005
6	0	0	1	0	0	0	1	0

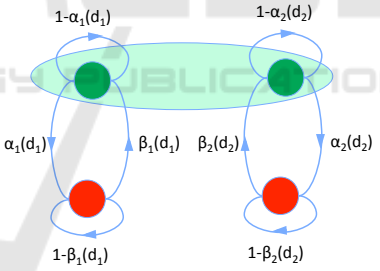


Figure 4: Communication model between two agents.

In the current experiment, the following parameters are used.

$$\alpha_1(d_1) = a_1 - b_1 * d_1, \quad \beta_1(d_1) = e_1 + f_1 * d_1$$
$$\alpha_2(d_2) = a_2 + b_2 * d_2, \quad \beta_2(d_2) = e_2 - f_2 * d_2$$

We divided the communication levels into $1, 2, \ldots, 6$, with 6 indicating perfect communications. The corresponding parameters are shown in Table 1.

The experiment results shown below are averaged over at least 100 runs of the same parameter setup to get statistically significant results. Our experiments have shown that above 80 repetitions, the results reach a stable state.

As communication degrades, messages - including those with task requirements, bidding requests, acknowledgments, etc - are being dropped, affecting

the quality of the results. For the following set of experiments, we used the following set of parameters unless otherwise stated. The number of agents is 60. There were 30 preassigned tasks before 100 pop-up tasks started to arrive in sequence. Each task has a duration requirement of 1000ms, and the pop-up arrival times follow a Poisson distribution with mean 50ms. The axis labeled "comms level" represents the communication condition divided into 6 levels as described above. The other horizontal axis represents the number of groups that the 60 agents are divided into. The vertical axis represents the percentage of tasks (out of all 130) that are successfully allocated.

Figure 5 shows the result of task completion under different communication conditions with different degrees of decentralization. As illustrated in Figure 5, a centralized system works best under perfect and relatively good communication conditions, and performance degrades as the number of groups increases. However, as communication conditions worsen, distributed systems start to outperform the centralized system. It is further clear that an optimal point of decentralization exists beyond which, as the group number increases, the performance starts to degrade again.

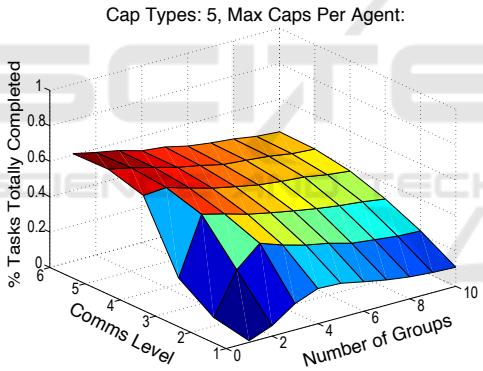


Figure 5: Tasks that were completely executed (all capabilities fulfilled).

Due to capability mismatches of and scheduling conflicts, agents can be idle at times. Communication failures also cause agents to be idle even when their capabilities are needed. Some agents either fail to participate in the auction or fail to receive the winner announcement. Our experiment results show that with imperfect communication, a lower percentage of agents are either assigned to (scheduled to execute) or busy with (executing) tasks. It is further illustrated that the drop in the number of agents that are assigned or busy is larger in centralized systems than in distributed systems when communication failures are introduced.

In the experiments, we placed the agents, tasks, etc. on an $N \times N$ grid, where $N \in \mathbb{R}_+$. If we vary N, but keep all else constant, smaller N will correspond to better results. This is because communications success depends on distance. Our experiments confirm that, given a fixed communication level, the larger grid (larger N) results in a greater optimal point of decentralization. For example, the optimal point is 3 groups for Comms Level 2 when $N = 100$, but 7 groups when $N = 200$. This trend is clear for all communication levels. The main conclusion is that the message drop rate dictates the optimal decentralization point. Similarly, our experiments also showed that the number of agents will *not* affect the optimal decentralization point, since it does not affect the message success rate.

To simulate adversaries attempting to block communications, we included *jammers* in the software. jammers have a location on the grid, and they interfere with communications in their vicinity.

We conjectured that the more jammers there are, the higher the optimal point of decentralization would be. The argument is that the more groups there are, the more likely it is that there are some auctioneers - as well as some of their agents - far away from jammers. This will increase the number of successful communications attempts. We again compute the average percentage of capabilities completed over thousands of runs. The results confirm our hypothesis. There is a marked change in the optimal points of decentralization, since the jammers affect the communications significantly. According to our initial analysis, this should significantly shift the optimal point along the group axis. However, we note that with smaller groups, each group has fewer capabilities it can provide, lessening the effects of message drops on the optimal point of decentralization. We see that on average, the optimal point of decentralization is about one group more when there are five jammers than when there are none.

Another interesting result we found when considering jammers is how well the algorithm performed in the contested-communications environment. At the optimal point of decentralization (4 groups), about 80% of communication attempts get through. At the optimal point with jammers (5 groups), between 0.5% and 1% of communication attempts are successful. Despite this huge loss in communications, the agents still manage to complete just over 30% of the capabilities.

7 CONCLUSION

In this paper, we presented an empirical analysis of the trade-offs between centralized and distributed re-

source allocation in communication contested environments. The results show that there is an optimal degree of decentralization that depends on the level of communications disruption, which opens the possibility of actively managed distributed regimes. We have extended the distributed auctioneer architecture with asset proxy agents and have shown that this extension improves performance by 10 to 20%. To perform empirical analysis, we developed a distributed auction architecture and winner determination algorithm that optimizes a global objective function with constraints. The set of these constraints enables us to apply the algorithm to many problems where typical combinatorial auction algorithms fail to capture the details and nuances of the application. We derived theoretical bounds and conducted experiments to both validate the theoretical analysis and, using the theoretical analysis, to validate the algorithm. Our results show close correspondence between experimental results from our algorithm and the theoretical bounds. Our next step is to enhance asset sharing among distributed auctioneers, modeling probabilistic resource allocation by incorporating knowledge, e.g., a profile of asset capabilities and expected geolocations.

ACKNOWLEDGEMENTS

This research was supported by ONR contract N00014-12-C-0162.

REFERENCES

Amador, S., Okamoto, S., and Zivan, R. (2014). Dynamic multi-agent task allocation with spatial and temporal constraints. In *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence*.

Bednowitz, N., Batta, R., and Nagi, R. (2014). Dispatching and loitering policies for unmanned aerial vehicles under dynamically arriving multiple priority targets. *Journal of Simulation*, 8(1):9–24.

Bellifemine, F. L., Caire, G., and Greenwood, D. (2007). *Developing Multi-Agent Systems with JADE*. Wiley.

Bonacquisto, P., Modica, G. D., Petralia, G., and Tomarchio, O. (2014). A strategy to optimize resource allocation in auction-based cloud markets. In *Proceeding of 2014 IEEE International Conference on Services Computing*.

Choi, H., Brunet, L., How, J. P., and Member, S. (2009). Consensus-based decentralized auctions for robust task allocation. *IEEE Transactions on Robotics*.

Cicalo, S., Tralli, V., and Perez-Neira, A. (2011). Centralized vs distributed resource allocation in multi-cell ofdma systems. In *Proceeding of 2011 IEEE 73rd Vehicular Technology Conference (VTC Spring)*.

Greene, K. and Hofmann, M. O. (2006). Coordinating busy agents using a hybrid clustering-auction approach. In *Proceeding of AAAI 06, Workshop on Auction Mechanisms for Robot Coordination*, Boston, MA.

Hong, S. A. and Gordon, G. J. (2011). Decomposition-based optimal market-based planning for multi-agent systems with shared resources. In *Proceeding of the 13th International Conference on Artificial Intelligence and Statistics (AISTSTS)*, Chia Laguna Resort, Sardinia, Italy.

Kong, P. (2002). Performance of queue in impaired wireless channel. *Electronic letters*, 38(22).

Lagoudakis, M. G., Berhault, M., Koenig, S., Keskinocak, P., and Kleywegt, A. J. (2004). Simple auctions with performance guarantees for multi-robot task allocation. In *Proceedings of the IEEE/RSJ International conference on Intelligent Robots and Systems (IROS)*.

Liu, L. and Shell, D. A. (2011). Assessing optimal assignment under uncertainty: An interval- based algorithm. *International Journal of Robotics Research*, 30(7):936–953.

Mauadi, M., Li, W., and Murata, T. (2011). Combinatorial auction method for decentralized task assignment of multiple-loading capacity agv based on intelligent agent architecture. In *Proceeding of the Second International Conference on Innovations in Bio-inspired Computing and Applications (IBICA)*, pages 207–211.

Roggendorf, M. and Beltran, F. (2006). Flow-based resource allocation in a multiple-access wireless market-setting using an auction. In *Proceeding of the 26th IEEE International Conference on Distributed Computing Systems Workshops (ICDCSW06)*.

Spall, J. C. (2012). Cyclic seesaw process for optimization and identification. *Journal of Optimization Theory and Applications*, 154(1):187–208.

Tsitsiklis, J. and Xu, K. (2011). On the power of (even a little) centralization in distributed processing. In *Proceeding of SIGMETRICS'11*.

Tung, A. K. H., Ng, R. T., Lakshmanan, L., and Han, J. (2001). Constraint-based clustering in large databases. In den Bussche, J. V. and Vianu, V., editors, *Database Theory ICDT 2001*, pages 405–419. Springer Berlin Heidelberg.

Wang, X. and Martinez, J. F. (2015). Xchange: A market-based approach to scalable dynamic multi-resource allocation in multicore architectures. In *Proceeding of 2015 IEEE 21st International Symposium on High Performance Computer Architecture (HPCA)*.

Zhang, K., Collins, Jr., E. G., and Shi, D. (2012). Centralized and distributed task allocation in multi-robot teams via a stochastic clustering auction. *ACM Trans. Auton. Adapt. Syst.*, 7(2):21:1–21:22.

Zheng, X. and Koenig, S. (2009). K-swaps: Cooperative negotiation for solving task-allocation problems. In *Proceeding of the International Joint Conference on Artificial Intelligence (IJCAI)*, pages 373–379.

Zheng, X. and Koenig, S. (2010). Sequential incremental value auctions. In *Proceeding of AAAI Conference on Artificial Intelligence (AAAI)*.