Multi-Agent Consensus Optimization in Large-Scale Supply Networks

Niyousha Rahimi

A thesis submitted in partial fulfillment
of the requirements for degree of

Master of Science in Mechanical Engineering

University of Washington

2018

Committee :

Ashis Banerjee

Santosh Devasia

Archis Ghate

Program Authorized to Offer Degree:
Department of Mechanical Engineering

University of Washington

Abstract

Multi-Agent Consensus Optimization in Large-Scale Supply Networks

Niyousha Rahimi

Chair of the Supervisory Committee:

Multi-agent systems are characterized by decentralized decision-making by the (semi)-autonomous agents and localized communication or information exchange among the neighboring agents. Supply-demand networks form the backbones of both services and manufacturing industries, and need to operate as efficiently as possible to yield optimized returns. In this Master's thesis, we bring the notion of multi-agent systems to clustered supply-demand networks such that each supplier acts as an agent. Consequently,

- We adapt consensus-based auction bidding methods to optimize the assignment of demands to the suppliers with known communication pathways and resource constraints.

- Results on moderately large networks are presented, which show promising performance in terms of both assignment quality, as given by the overall demand delivery cost and proportion of assigned demands, and computation time.

TABLE OF CONTENTS

LIST OF FIGURES

ACKNOWLEDGMENTS

I would like to express my gratitude to my advisor, Professor Ashis Banerjee for the endless patience, valuable feedback, engagement and funding through the learning process of this Master's thesis. Furthermore I would like to thank Ilya Buzytsky and Andrey Shishkarev, the CEO and the CTO of BIAS Intelligence company who provided the funding and helped me through the completion of this work. A special thanks to Jundi Liu for helping with the experimentation.

I would like to also thank my loved ones, my mother and my husband, who have supported me throughout entire process, both by keeping me harmonious and helping me putting pieces together. I will be grateful forever for your love.

DEDICATION

to my mother, without whom I would never be where I am today

and to my dear husband, Arshia for his endless love and support

Chapter 1

INTRODUCTION

Supply networks, which form integral components of all services and manufacturing industries, have received a lot of attention over the past decade or so due to the ever-increasing need to design and operate them with maximal agility and efficiency. Representative examples include intelligent transportation services such as car sharing, where the customers do not own the cars but rent them as and when necessary from whichever depot they wish to. In direct logistics services, all kinds of goods deliveries are made to a large number of customers at widely varying locations within fixed time windows. Similarly, in streamlined manufacturing, raw materials and finished products are made available to the production facilities and sellers, respectively, before their supplies run out based on both current and forecasted demands. For all these industries, the common requirement, therefore, is that the suppliers (resource providers) are able to deliver as much of the demands, as quickly as possible, whenever and wherever they show up while minimizing their operational costs. An example of such supply-networks is illustrated in Fig. 1.1.

While a lot of progress has been made in designing and understanding the commercial and societal impacts of such services, real-time schedule optimization remains an extremely challenging problem. Existing services rely mostly on nominal schedules that are optimally generated using hours of computation. As a result, these schedules are either always non-optimal leading to increased costs and lower revenues, or cannot be optimized whenever some form of disruption happens that may lead to non-fulfillment of user/customer demands. To address these issues, we developed a cloud-based software solution using distributed multi-agent consensus (MAC) methods instead of centralized combinatorial optimization methods (e.g., mixed integer linear programs) that are employed commonly. In this project, we scaled

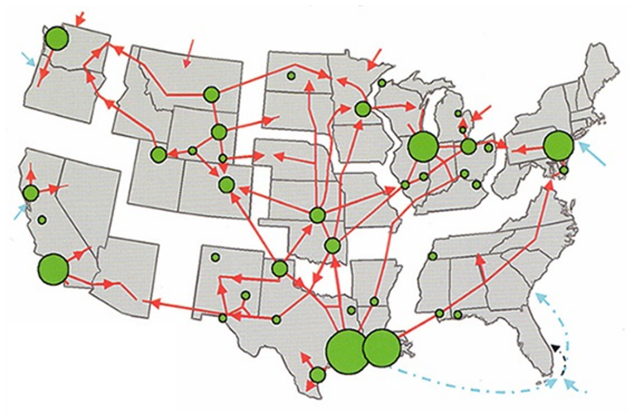

Figure 1.1: Schematic illustration of a supply-network across a country

up these MAC methods to real-world transportation networks, investigated how they can be best combined with widely-used optimization methods, and ensured robustness to a vast majority of service disruptions. While existing MAC methods were evaluated during the initial part of the project, recent advances in topological analysis and control of complex, time-varying networks were leveraged to design a novel method in the latter half of the project. We believe these methods would have a game-changing effect in realizing truly meaningful schedule optimization for real-world transportation services on an asper-demand basis.

Chapter 2
LITERATURE REVIEW

The goal of this thesis is to develop effective multi-agent consensus methods for (near) real-time optimization of schedules in next-generation transportation services. Such services may include car sharing programs, where the users do not own the cars but rent them as and when necessary, and delivery vehicle fleet programs, where consumer deliveries have to be made on time or within fixed time windows. Any such service, therefore, requires effective scheduling of resources (vehicles, drivers, etc.) to satisfy user/customer demands.

Existing methods for (near) real-time optimal assignment of the demands to the suppliers, rely mostly on mathematical programming, especially heuristics [1], Linear Programs (LPs) [2], and the widely-used Mixed Integer Linear Programs (MILPs) [3]. However, these methods often do not incorporate the communication constraints among the suppliers. Furthermore, they belong to the paradigm of centralized decision-making, where the suppliers are simply allocated specific demands rather than being able to choose which demands to deliver. Correspondingly, the computation times are often large, and scale exponentially with the problem size in the case of MILPs [4].

One related topic is the problem of assigning orders for parts among various suppliers in a supply chain to deliver customer demands for products at low cost. There are many studies on this topic, the focus of which are on evaluating and prioritizing suppliers and assigning demands among the suppliers [5–13]. In [14], a hybrid algorithm is proposed based on MILP and the Technique for Order Preference by Similarity to Ideal Solution (TOPSIS) to assign customer demands to the suppliers. Sawik [15] present a new decision-making problem based on two conflicting objective functions: cost and customer service level. The combinatorial optimization problem is formulated as a stochastic mixed integer program with the ordered weighted averaging aggregation of the two conflicting objective functions. Furthermore, Mohammaditabar et al. [16] make use of cooperative and non-cooperative game theory to analyze supplier selection in decentralized supply chains, where the suppliers have capacity constraints. In this work, the two areas of supplier selection models and coordination mechanisms are combined to analyze the selected suppliers and total supply chain costs. Consequently, it establishes the notion that demand assignment in supply networks is similar to task planning for a team of networked agents.

Here, we build on this similarity notion, and consider each supplier in the network as an independent decision-making agent, which shares limited information with its adjacent (communicating) neighbors. This approach is particularly beneficial in the transportation, logistics, and manufacturing scenarios discussed earlier, where the independent suppliers are *competing* to provide resources for the incoming demands. To retain competitive advantages, they want to share as little information as possible on delivery costs and available resources. A centralized solution would not work well in the absence of complete information sharing among the suppliers. Furthermore, it would not be resilient to failures and uncertainties resulting from equipment breakdowns, traffic delays, adverse weather conditions, etc.

Hence, we adapt optimization methods for networked multi-agent systems, particularly the Consensus-Based Bundle Algorithm (CBBA) presented in [17]. CBBA is a decentralized auction algorithm for task assignment in a fleet of autonomous vehicles. Auctioning is used for decentralized task assignment and a consensus procedure is employed for decentralized conflict resolution. CBBA focuses on finding the best bidder, which provides the lowest cost for each task, while assuming that no constraint is imposed on the number of tasks allocated to each agent. It is further augmented in [18] by adding task decomposition and task elimination protocols to ensure cooperation in a network of heterogeneous agents. Several researchers have also adapted the algorithm for asynchronous communication [19], localized communication [20], heterogeneous robots [21], robot-task clusters [22], and sensor management for tracking space objects [23]. A single-item auction method has also been developed for task assignment with temporal constraints [24], and a distributed auction algorithm has been presented for the generic assignment problem [25].

We extend the CBBA method to assign demands to the supply networks by explicitly modeling *resource constraints* in the form of supplier capacities and demand volumes. The CBBA method provides a collection of action rules for allocating a certain number of tasks to autonomous vehicles, where no constraints are imposed on the number of tasks that can be assigned to each vehicle. On the other hand, in our method, we introduce a new set of action rules for assigning demands, with given volumes, to suppliers with limited capacities. We then combine the modified CBBA method with another optimizer to split up the leftover demand volumes and assign them to the suppliers that have not yet exhausted their capacities and are willing to share additional information.

We evaluate different versions of our method, where they differ as regards how and when the demands are split up, on five sets of randomly generated networks ranging from reasonably small (100 suppliers and 100 demands) to moderately large (500 suppliers and 1400 demands). The results indicate that the most realistic method version yields solutions, which are quite close to a baseline optimizer in terms of demand delivery costs and proportions of feasible demand volumes assigned, reasonably quickly, regardless of the problem size.

Chapter 3
TECHNICAL APPROACH

3.1 Network Structure

The network representing the supply-demand problem consists of agents (suppliers) as the nodes. The network is partitioned into P disjoint clusters based on considerations such as geo-spatial proximities, common customers or capabilities, and similar delivery costs[1]. Each cluster also contains a virtual demand node that acts as the demand manager and the cluster representative. As demands come to the network, the virtual node in each cluster stores the information about the demands' volumes, i.e., the number of resources required by the demands. The agents then decide individually how much resources (capacities) to assign for what demands based on their own demand-specific cost values and the limited information shared by the other communicating agents, which does not include their respective cost values. Note that the costs are determined by each supplier for each unit demand, based on the delivery distance/time and the maximum order quantity (demand volume). The overall objective is to assign the maximum feasible demand volume with the least total delivery cost as given by the sum of the individual supplier delivery costs.

The edges between any two nodes indicate that the corresponding agents are connected, i.e., they can communicate with each other. The virtual demand node in each cluster is connected to all the suppliers in that cluster. We assume that all the clusters representatives (virtual demand nodes) are connected to each other, and, therefore, they form a complete graph. However, the suppliers within a cluster do not necessarily form a complete graph[2].

[1]Identifying these clusters is a problem in its own right, and is outside the scope of this work. Apart from standard clustering techniques, recent works on network community detection [26] might be useful in this regard.

[2]The suppliers in a cluster may also be connected to certain suppliers in other clusters, which is not

Such a network is modeled as a simple undirected graph $G = (V, E)$, with the set of nodes $V = \{v_1, v_2, \ldots, v_n\}$ and the set of undirected edges $E = \{(v_i, v_j)\} \subseteq V \times V$. Each edge $(v_i, v_j) \in E$ represents the connection between nodes v_i and v_j. This graph is partitioned with P clusters, which are subsets of the original graph G:

$$G_1, G_2, \ldots, G_P \subset G \tag{3.1}$$

where

$$G_p = (V_p, E_p) \quad \forall p \in \{1, 2, \ldots, P\}.$$

Here, $V_p \subset V$ includes both the supplier nodes and the representative (virtual demand node v_{dp}) in the cluster p. $E_p \subset E$ includes both the connections among the suppliers and the connections with v_{dp} in the cluster p. The cluster representatives create a complete graph K_P of size P, and is denoted as G_v:

$$\begin{aligned} G_v = \{V_v, E_v\} = K_P \subset G \\ V_v = \{v_{d1}, v_{d2}, \ldots, v_{dP}\}. \end{aligned} \tag{3.2}$$

The adjacency matrix $L_{G_p}(t)$ representing the agents' communication links is defined in (3.3). The agents are not required to communicate all the time; hence, the elements of the adjacency matrix change with respect to time t.

$$[L(G_p)(t)]_{ij} = \begin{cases} 1 & \text{if } (v_i, v_j) \in E_p \quad \& \quad v_i \quad \text{and} \quad v_j \\ & \text{are communicating} \\ 0 & \text{otherwise.} \end{cases} \tag{3.3}$$

The capacity of any cluster is the sum of the capacities of all its suppliers. If $r(v_i)$ denotes the resources of node v_i in cluster $p \in \{1, 2, \ldots, P\}$, the capacity of cluster p, R_p, is given by

$$R_p = \sum_{v_i \in V_p} r(v_i). \tag{3.4}$$

explicitly considered in our demand assignment methods.

When demands come into the network at any time instant t, the demand managers in each cluster are updated with the new demand information. The list of demands for the network is represented as:

$$D(t) = \{D_1, D_2, \ldots, D_{O(t)}\}, \tag{3.5}$$

where $O(t)$ is the number of demands that appear at time t. The list of demands assigned to each cluster is then represented as:

$$d_p(t) = \{D_1^p, D_2^p, \ldots, D_{Z_p}^p\}, \tag{3.6}$$

where $p \in \{1, 2, \ldots, P\}$, $D_i^p \in D(t)$, and Z_p is the number of demands assigned to cluster p.

The goal of the demand assignment problem, therefore, is to find a conflict-free matching of demand volumes (or portions of demand volumes) to the clusters that minimizes a global cost function and forms a demand set for each cluster. The goal, afterward, is to assign demand volumes (or again, portions of demand volumes) from each clusters' demand set to its suppliers. For any demand D_j coming to the network, each cluster p has a unit cost c_{pj} that is the average unit delivery cost of its suppliers for D_j. Hence, the global cost is the total sum of the delivery costs for the demand volumes assigned to all the clusters. Each cluster is assigned a maximum of Z_p demands, such that $r(d_p(t)) \leq R_p$, where $r(d_p(t))$ is the total number of resources that cluster p's demand set (d_p) requires at time t, and R_p is the total number of resources that cluster p possesses.

Note here that the demand volumes may be split among the clusters, and the total volume of each demand D_j assigned to the clusters should be less than or equal to $r(D_j)$, the total number of resources required by D_j. The assignment process is completed either when all the demands in $D(t)$ are assigned to the clusters, or the network runs out of resources. Hence, when the process is complete, a total of $N_{min} = \min\{\sum_{p=1}^{P} R_p, \sum_{j=1}^{O(t)} r(D_j)\}$ resources is assigned to the demands. An identical problem is solved for supplier-level assignment within each cluster. The overall demand assignment problem is illustrated schematically in Fig. 3.1.

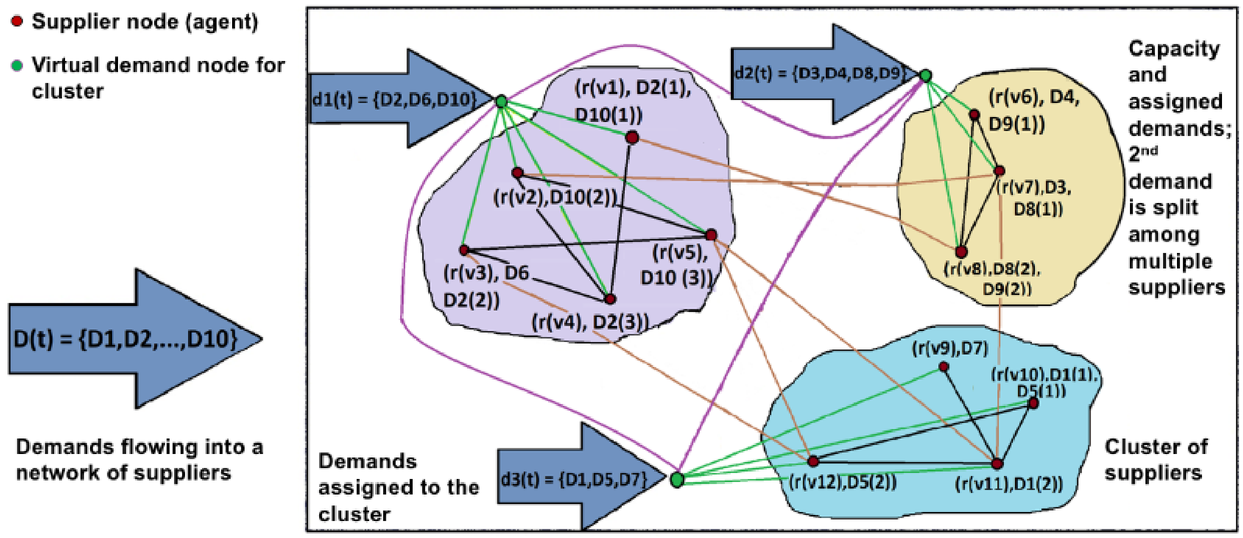

Figure 3.1: Schematic illustration of the optimal assignment problem for supply-demand networks. The problem is modeled as a two-stage process, where the demands are first assigned to the suppliers clusters and then to the individual suppliers within the clusters. The virtual demand nodes serve as the representative clusters nodes that store all the assigned demands so far and communicate among the clusters if necessary. Certain demands are split into multiple components, both among the clusters and the suppliers, to assign as much of the demands as possible while satisfying the suppliers capacities.

3.2 *Multi-Agent Consensus Methods*

We now present our multi-agent consensus methods for assigning the demands in two stages, first to the clusters and then to the suppliers in each of the clusters. In both the stages, we adopt a two-pronged approach: a) an auction bidding mechanism for first assigning complete demands (if possible) to the agents without sharing cost information; b) another optimization process to assign the leftover demands (if possible) only to those agents who are willing to share their cost and remaining capacity information[3].

In networks of agents, the term *consensus* refers to reaching an agreement regarding

[3]It is reasonable to expect that at least some of the agents would be interested in sharing their "sensitive" information to get additional demands. Also, note that the cost values for only the unassigned demands, and remaining rather than total agent capacities are required for this process.

a certain quantity of interest that depends on all the agents states [27]. We adapt the CBBA method [17] to allow the (partially) communicating agents to bid for the demands independently, and arrive at a consensus regarding the winning bid list and the winners list based on the demand delivery costs and available resources. We replace the bundle construction process in the CBBA method with an initialization phase that takes into account the fact that the agents can only provide resources for a limited number of demands as permitted by their respective capacities. Subsequently, we introduce a new set of action rules to modify the conflict resolutions process that explicitly considers the agents resource constraints. The methods are described below for both the demand assignment stages.

3.2.1 First Stage

Consensus optimization for complete demand assignment without cost information sharing by the agents consists of two phases: initialization and conflict resolution.

- **Phase 1: Initialization**

Each cluster first places a bid on each demand. This bid is based on the average lowest cost of delivery for each demand provided by the suppliers in that cluster. The bids comprise the lowest costs that the clusters can provide for each demand. Let $c_{pj} \geq 0$ be the cost of the bid that cluster p places on demand j. Each cluster carries three vectors: a winning bid list $Y_p \in \mathbb{R}_+^{O(t)}$, which is an up-to-date estimate of the winner's bid made for each demand thus far, a winners list $w_p \in \{1, \ldots, P\}^{O(t)}$, where P is the number of clusters in the network, and a cost vector $C_p = [c_{p1}, c_{p2}, \ldots, c_{pO(t)}]^T$, where $O(t)$ is the total number of demands in the network's demand set $D(t)$. For each cluster, Y_p is initialized as the cost vector, i.e., $Y_p = C_p$, and w_p is initialized as an empty list. Algorithm 1 demonstrates the procedure of cluster p's phase 1 at time t.

Algorithm 1: Phase 1 for cluster p:

```
function Build Cost Vector $(D(t), p)$
$j \leftarrow 0$
while $j \leq O$ do
    $C_p(j) \leftarrow \min_c c_{pj} \; \forall \; D_j \in D(t)$
    $w_p(j) \leftarrow \emptyset$
    $j \leftarrow j + 1$
end
$Y_p \leftarrow C_p$
return $(C_p, w_p, Y_p)$
end function
```

The assignment process at time step t consists of a single run of phase 1 (to determine C_p and initialize Y_p, w_p) and several iterations of phase 2 (to reach a consensus on the winning bids list Y_p and the winners list w_p). Note that each cluster's iteration count can be different, which allows the possibility that each cluster has different iteration periods. In this phase, each cluster determines a minimum cost for each demand regardless of the number of required resources. This information is used in the next phase to determine the winning bid and the winner.

- **Phase 2: Conflict Resolution**

In this phase, clusters make use of a consensus strategy to converge to a winning bid list Y_p and a winners' list w_p. This allows conflict resolution over all the assignments while not limiting the network to a specific structure. The adjacency matrix for the cluster representatives is defined such that $l_{pk}(t) = 1$ if a link exists between the representatives of cluster p and cluster k at time t, and 0 otherwise (just as a reminder, the representative of each cluster is its demand manager).

Consensus is performed on Y_p and w_p based on Y_k and w_k received from each neighbor for all k such that $l_{pk} = 1$. In each iteration, clusters receive information (Y_k and w_k) from their adjacent neighbors $\{k|l_{pk} = 1\}$, and determine the winning bid and the winner for each demand. The key characteristic of our consensus protocol is that the sequence of demand assignments is based on the *maximum difference in the demand cost* as determined by the two communicating clusters. Therefore, based on the shared information, each cluster adds suitable demands to its demand set $d_p(t)$ while trying to reach a consensus on Y_p and w_p with the other clusters.

A cluster loses its assigned demand if it is outbid by other clusters for the demand it has selected, and reimburses its resources. Apart from Y_p and w_p, each cluster communicates another vector which is $s_p \in \mathbb{R}^P$. This vector represents the time stamp of the last information update from each of the other clusters. Each time a message is passed, the time vector is populated with

$$s_{pk} = \begin{cases} \gamma & \text{if } l_{pk} = 1 \\ \max\limits_{\{m: l_{pm}=1\}} s_{mk} & \text{otherwise} \end{cases} \tag{3.7}$$

where γ is the message reception time.

The algorithm uses w_p and s_p to determine which clusters information is the most up-to-date for each demand, every time it receives a message from cluster k. In the consensus process, we need to use a decision rule to determine the winning bid and the winning cluster for each demand. This decision rule is explained in Table 3.1. Cluster p is the receiver and cluster k is the sender. There are five possible actions cluster p can take for demand j:

1. **Update:** $Y_p(j) = Y_k(j)$, $w_p(j) = w_k(j)$; Cluster p will update the jth element of its winning bids list Y_p and its winner list w_p.

2. **Reset:** $Y_p(j) = C_p(j)$, $w_p(j) = \emptyset$; cluster p resets the jth element of its Y_p and w_p to the initial state.

3. **Leave:** $Y_p(j) = Y_p(j)$, $w_p(j) = w_p(j)$;

4. **Add:** $Y_p(j) = C_p(j)$, $w_p(j) = p$, $d_p(t) \oplus D_j$, $R_p \leftarrow R_p - r(D_j)$; here, $d_p(t) \oplus D_j$ means cluster p adds demand D_j to its demand set.

5. **Assign to k:** $Y_p(j) = Y_k(j)$, $w_p(j) = k$

At the beginning of the algorithm, $w_p(t)$ for each cluster p is a vector with $O(t)$ empty elements. Also, for each cluster p, $Y_p(t)$ is equal to $C_p(t)$. This means that the winning bid for each demand is the minimum cost of delivery as determined by each cluster p. In each iteration of phase 2 for cluster $p \in \{1, \ldots, P\}$, when p receives information from $k \in \{k|l_{pk}(t) = 1\}$, it first checks if the two clusters have already reached a consensus on the winning bid list ($Y_p(t)$) and the winners' list ($w_p(t)$), which, of course, at iteration 1, they have not. Then, it starts the process of calculating dC, which is a vector identifying the difference in the cost of delivery for each demand as determined by cluster p and the winner in cluster k's winner's list ($dC(j) = C_p(j) - Y_k(j)$). Cluster p makes a decision for each demand using Table 3.1, starting with the demand that has the maximum difference in the delivery cost (i.e., $J_p = \arg\max_j |dC(j)|$).

Table 3.1: Action Rule for Cluster p Based on Communication with Cluster k Regarding Demand j

Cluster k (sender) thinks $w_k(J_p)$ is	**Cluster p (receiver) thinks $w_p(J_p)$ is**	**Receiver's Action (default: leave)**
k	p	if $dC(J_p) > 0$: update and reimburse
	k	update
	$m \notin \{p, k\}$	if $s_{km} > s_{pm}$ or $dC(J_p) \geq 0$: update

	none ($\emptyset$)	if $dC(J_p) < 0$ & $r(D_{J_P}) < R_p$: add
		else : update

p	p	leave

	k	if $r(D_{Jp}) \leq R_p$: add
		else if $c_{pJ_p} \leq Y_p(J_p)$: leave
		else if $c_{pJ_p} > Y_p(J_p)$: reset

	$m \notin \{p, k\}$	if $s_{km} > s_{pm}$: if $r(D_{Jp}) \leq R_p$: add
		else if $c_{pJ_p} < Y_p(J_p)$: leave
		else if $c_{pJ_p} > Y_p(J_p)$: reset

	none ($\emptyset$)	if $r(D_{J_p}) \leq R_p$: add
$m \notin \{p, k\}$	p	if $s_{km} > s_{pm}$ and $dC(J_p) \geq 0$: update & reimburse

	k	if $s_{km} > s_{pm}$: update
		else : reset

	m	if $s_{km} > s_{pm}$: update

	$n \notin \{p, k, m\}$	if $s_{km} > s_{pm}$ and $s_{kn} > s_{pn}$: update
		if $s_{km} > s_{pm}$ and $dC(J_p) > 0$: update
		if $s_{kn} > s_{pn}$ and $s_{pm} > s_{km}$: reset

	none ($\emptyset$)	if $s_{km} \geq s_{pm}$ & $dC(J_p) \geq 0$: update
		else if $s_{km} \geq s_{pm}$ & $dC(J_p) < 0$: add

none ($\emptyset$)	p	if $dC(J_p) \leq 0$: leave
		else assign to k & reimburse

	k	reset

	$m \notin \{p, k\}$	if $s_{km} > s_{pm}$: reset
		else if $dC(J_p) > 0$: assign to k

	none ($\emptyset$)	if $dC(J_p) \leq 0$& $r(D_{J_p}) \leq R_p$: add
		else assign to k

This process continues for the rest of the senders $k \in \{k|l_{pk} = 1\}$ and iterations of phase 2 continues for each cluster p until they all reach a consensus on the winning bid list and the winners' list. Algorithm 2 demonstrate the pseudo code function for the second phase. Note that the number of iterations for each cluster is independent. Phase 2 of the algorithm runs simultaneously for all the clusters until they reach a consensus. In order to define a consensus on the winning bid list, we define a $(P \times P)$ matrix: $\mathcal{X} \in \{0,1\}^{P\times P}$.

Algorithm 2: Phase 2 for cluster p at iteration τ

```
11 Receive $Y_k$ and $w_k$ from $k$ with $l_{pk}(t) = 1$
12 function Resolve Conflicts ($(L_p(t), C_p(t), Y_{k\in\{k|l_{pk}(t)=1\}}, w_{k\in\{k|l_{pk}(t)=1\}}, s_{k\in\{k|l_{pk}(t)=1\}})$)
13 Send $Y_p$ and $w_p$ to $k$ with $l_{pk}(t) = 1$
14 Receive $Y_k$ and $w_k$ from $k$ with $l_{pk}(t) = 1$
15 if $Y_p(t) = Y_k(t)$ then
16     $\mathcal{X}(p,k) \leftarrow 1$
17 else
18     $\mathcal{X}(p,k) \leftarrow 0$; $dC \leftarrow C_p(t) - Y_k(t)$
19     $i \leftarrow 0$
20     while $i \leq O$ do
21         $J_p \leftarrow \arg\max_j |dC(j)|$
22         Determine $w_p(J_p)$ and $Y_p(J_p)$ according to Table 3.1
23         $dC \ominus dC(J_p)$
24         $i \leftarrow i + 1$
25 return $(w_p, Y_p, \mathcal{X})$
26 end function
```

At the beginning, since none of the clusters agree on the winning bid list, $\mathcal{X}$ is an identity matrix $\mathbb{I}_P$ (each cluster only agrees with itself). When cluster p exchanges data with an adjacent cluster k, it first checks if they agree on the winning bid list. If this is true, cluster p assigns 1 to the element (p,k) of the $\mathcal{X}$ matrix, ends the procedure for cluster k, and goes to the next iteration. If these two clusters do not agree on the winning bid list, cluster p assigns 0 to the element (p,k) of the $\mathcal{X}$ matrix, and goes through the rest of the process.

It is worth noting here that $\mathcal{X}$ may not be always symmetric since each cluster can be at a different iteration. However, after a sufficiently large number of iterations, $\mathcal{X}$ becomes symmetric. The algorithm terminates when all the elements of $\mathcal{X}$ are equal to 1.

The consensus bidding algorithm for the first stage of demand assignment without any form of splitting the demands is explained in Algorithm.3

Algorithm 3: Consensus-based auction bidding for assigning demands to clusters without splitting them

```
procedure Assign Demands Without Splitting
foreach $p \in \{1, \dots, P\}$ do
    $(C_p, w_p, Y_p) \leftarrow$ Build Cost Vector $(D(t), p)$
end
$\mathcal{X} \leftarrow \mathbb{I}_P$
foreach $p \in \{1, \dots, P\}$ do
    while $\mathcal{X} \neq \mathbf{1}$ do
        $(w_p, Y_p, \mathcal{X}) \leftarrow$ Resolve Conflicts $((L_p(t),\ C_p(t),\ Y_{k\in\{k|l_{pk}(t)=1\}},\ w_{k\in\{k|l_{pk}(t)=1\}},$
          $s_{k\in\{k|l_{pk}(t)=1\}})$
    end
end
end procedure

function Build Cost Vector $(D(t), p)$
$j \leftarrow 0$
while $j \leq O$ do
    $C_p(j) \leftarrow \min_c c_{pj}\ \forall\ D_j \in D(t)$
    $w_p(j) \leftarrow \emptyset$
    $j \leftarrow j + 1$
end
$Y_p \leftarrow C_p$
return $(C_p, w_p, Y_p)$
end function

function Resolve Conflicts $((L_p(t),\ C_p(t),\ Y_{k\in\{k|l_{pk}(t)=1\}},\ w_{k\in\{k|l_{pk}(t)=1\}},\ s_{k\in\{k|l_{pk}(t)=1\}})$
Send $Y_p$ and $w_p$ to $k$ with $l_{pk}(t) = 1$
Receive $Y_k$ and $w_k$ from $k$ with $l_{pk}(t) = 1$
if $Y_p(t) = Y_k(t)$ then
    $\mathcal{X}(p, k) \leftarrow 1$
end
else
    $\mathcal{X}(p, k) \leftarrow 0$; $dC \leftarrow C_p(t) - Y_k(t)$
    $i \leftarrow 0$
    while $i \leq O$ do
        $J_p \leftarrow \arg\max_j |dC(j)|$
        Determine $w_p(J_p)$ and $Y_p(J_p)$ according to Table 3.1
        $dC \ominus dC(J_p)$
        $i \leftarrow i + 1$
    end
end
return $(w_p, Y_p, \mathcal{X})$
end function
```

- **Assigning Left-over Demands**

When demands are assigned to the clusters, there may be several demands that are not delivered due to the paucity of resources. Such non-deliveries happen either when the network does not have the resources for these demands, or none of the clusters have enough resources to deliver any of these demands individually. In the former scenario, nothing can be done and the system reports the unassigned demands. However, in the latter scenario, since the clusters together may have enough resources to completely or partially deliver the leftover demands, another optimization method is used to assign them to the clusters. This method enables the network to fully use its resources to respond to the demands.

This process requires some additional information from the clusters, which includes their remaining capacities (CapacityInfo(p)), unit costs for the remaining demands (CostInfo(p, j)), and their willingness to participate in this process. The clusters can either choose to participate or refuse to share their information. A vector, termed as ClusterWill, records the willingness of each cluster p to collaborate. After receiving the additional information from the clusters, vector dc is constructed to store the difference between the maximum and minimum cost of delivery for each demand. Starting from a demand J_p that has the maximum difference, the optimizer searches for a cluster p_{J_p} that provides the minimum cost for this demand. Since none of the clusters can individually provide the resources for any of the leftover demands, the optimizer splits the demand based on cluster p_{J_p}'s remaining capacity, assigns the first part to cluster p_{J_p}, and searches for the next best cluster to assign the second part. If the next best cluster has enough resources to deliver the second part of the demand, it is assigned to this cluster and the optimizer continues to assign the next demand. Otherwise, the second part of this demand is split further based on the cluster's capacity and the optimizer searches for the third best cluster. This process continues until either all the demands are assigned or the system runs out of resources. Algorithm. 4 explains this process. During the cluster level assignment, this optimization method act as a centralized process.

Algorithm 4: Optimized assignment of leftover demands to clusters

```
procedure Assign Leftover Demands by Splitting
foreach $p \in \{1, \dots, P\}$ do
    $(\mathrm{CostInfo}(p,:), \mathrm{CapacityInfo}(p)) \leftarrow$ Get Cluster's Information$(p)$
end
foreach $D_j \in D(t)$ do
    $\mathrm{maxCost}(j) \leftarrow \max_j \mathrm{CostInfo}(p,j)$
    $\mathrm{minCost}(j) \leftarrow \min_j \mathrm{CostInfo}(p,j)$
end
$dc \leftarrow \mathrm{maxCost} - \mathrm{minCost}$
while $D(t) \neq \emptyset$ or $\sum_p \mathrm{CapacityInfo}(p) \neq 0$ do
    Split and Assign Demands
end
end procedure

function Get Cluster's Information$(p)$
if $\mathrm{ClusterWill}(p) == 1$ then
    $\mathrm{CostInfo}(p,j) \leftarrow \min_c c_{pj} \ \forall \ D_j \in D(t)$
    $\mathrm{CapacityInfo}(p) \leftarrow R_p$
end
return $(\mathrm{CostInfo}, \mathrm{CapacityInfo})$
end function

function Split and Assign Demands
$J_p \leftarrow \arg\max_j |dc(j)|$
while $r(D_{J_p}) \neq 0$ or $\sum_p \mathrm{CapacityInfo}(p) \neq 0$ do
    $p_{J_p} \leftarrow \arg\min_p \mathrm{CostInfo}(p, J_p)$
    if $r(D_{J_p}) \leq R_{p_{J_p}}$ then
        $d_p(t) \oplus D_{J_p}$
        $R_{p_{J_p}} \leftarrow R_{p_{J_p}} - r(D_{J_p})$
        $\mathrm{CapacityInfo}(p) \leftarrow R_{p_{J_p}}$
        $D(t) \ominus D_{J_p}$
    end
    else
        $r(D_{J_p}(1)) \leftarrow R_{p_{J_p}}$
        $r(D_{J_p}) \leftarrow r(D_{J_p}) - R_{p_{J_p}}$
        $d_p(t) \oplus D_{J_p}(1)$
        $R_{p_{J_p}} \leftarrow 0$
        $\mathrm{CapacityInfo}(p) \leftarrow R_{p_{J_p}}$
    end
end
end function
```

3.2.2 Second Stage

After assigning the demands to the clusters, the demands in each cluster are assigned to the suppliers in that cluster. The consensus auction bidding optimization procedure to reach a consensus on the winning bid list and the winners list for the cluster's demand set is the same as described above. Only the connected suppliers communicate with each other in reaching the consensus. The problem of leftover demands may also occur in this stage. We then use the optimization method described before to assign the leftover demands to those suppliers that participate in exchanging cost and remaining capacity information. This process is performed by the virtual demand node in each cluster, which receives all the additional information from the participating suppliers.

3.3 Demand assignment methods

We combine the consensus-based auction bidding process with the leftover demand assignment optimizer in several ways to come up with five different demand assignment methods. These methods are described as follows.

Method 1: In this case, full volume demands are assigned to the clusters using our consensus auction bidding process in the first stage. After this step, when the clusters reach a consensus, the optimizer is used to assign the leftover demands to the clusters, assuming that all the clusters agree to share cost and remaining capacity information. By the end of this step, the network has used all its resources to respond to the demands.

In the second stage, to obtain a fast solution for demand assignment and avoid leftover demands as much as possible, at the beginning, all the demands are split into an equal number of parts based on their volumes. Subsequently, as in the first stage, consensus-based auction bidding method is used to assign the split demands to the suppliers of each cluster. After the suppliers have reached a consensus, the optimizer is used to assign the leftover demands to the suppliers as described before.

Method 2: This method is basically the same as the first method with one key difference.

In the second stage, when the leftover demands are assigned to the suppliers in each cluster, some of the suppliers may choose not to share their information, i.e., remaining capacity and delivery cost for the leftover demands. This choice results in assignment of the leftover demands only to those suppliers that are willing to participate in the communication process. Clearly, the network may not use its full capacity to respond to the demands. However, this method considers a more realistic scenario for the demand assignment problem, and may be most useful in practice.

Method 3: In this case, the demands are assigned to the clusters and then to the suppliers in each cluster as described in the first method. The only difference here is that leftover demand assignment is not carried out at the end of the second stage In other words, the optimization method for leftover demand assignment is only used during the first stage when the demands are assigned to the clusters. This may happen in practice when none of the suppliers are willing to directly share their cost and capacity information due to the proprietary nature of the information and/or the competitiveness of the demand bidding process. Note here that sharing such information is less of an issue during the first stage as all the supplier cost and remaining capacity values get combined for the entire cluster, making it difficult to derive specific competitive benefits.

Method 4: Here, again, the consensus-based auction bidding and leftover demand assignment are carried out in the same manner as in the first method to assign the demands to the clusters. The difference from the first method is that at the beginning of the second stage, the demands are not divided into an equal number of parts. Instead, the demands are assigned to the suppliers as full volumes, and after a consensus is reached, the optimizer splits the leftover demands and assigns them to the suppliers. This means that the demands are split up only at the end of the assignment process based on the remaining capacities of the suppliers.

Method 5: In this last case, the demands are assigned to the clusters and then to the suppliers in each cluster using only the consensus-based auction bidding procedure. No attempt is made to assign the leftover demands by splitting them up and requesting exchange of additional information among the agents. It is, therefore, expected that this method, being the simplest one considered, would be most computationally efficient but also least optimal in its assignment.

Chapter 4
IMPLEMENTATION

4.1 Results

We now present the results of applying these five demand assignment methods for different network complexities. For each network complexity, 30 different random configurations were generated to evaluate the methods. The methods were implemented using C# in Microsoft Visual Studio Community 2017, version 15.3.5 and Microsoft .NET framework version 4.7.02556. All the tests were run on a desktop PC with an Intel Xeon E5-1607 v4 3.10 GHz processor, 8 GB of RAM, and Windows 10 Pro as the OS. When needed, the demands in each cluster's demand set were split into four equal parts, and the suppliers had a 50% probability of participating in the leftover demand assignment process[1]. To compare the performance of our methods with a best-case centralized optimizer, we also implemented an integer linear program (ILP) in the widely-used IBM ILOG CPLEX Optimization Studio v12.8 - Student (CJ2IKML) [28]. The ILP modeled the demand assignment problem without accounting for the supplier communication constraints as described in the Appendix.

[1]Although not reported here for the sake of brevity, we tested our methods for various choices of demand volume split and supplier participation probability. The delivery costs are insensitive to the split value, and decreases a little bit as the probability increases to 50% before plateauing out for higher values, indicating the overall robustness of our methods.

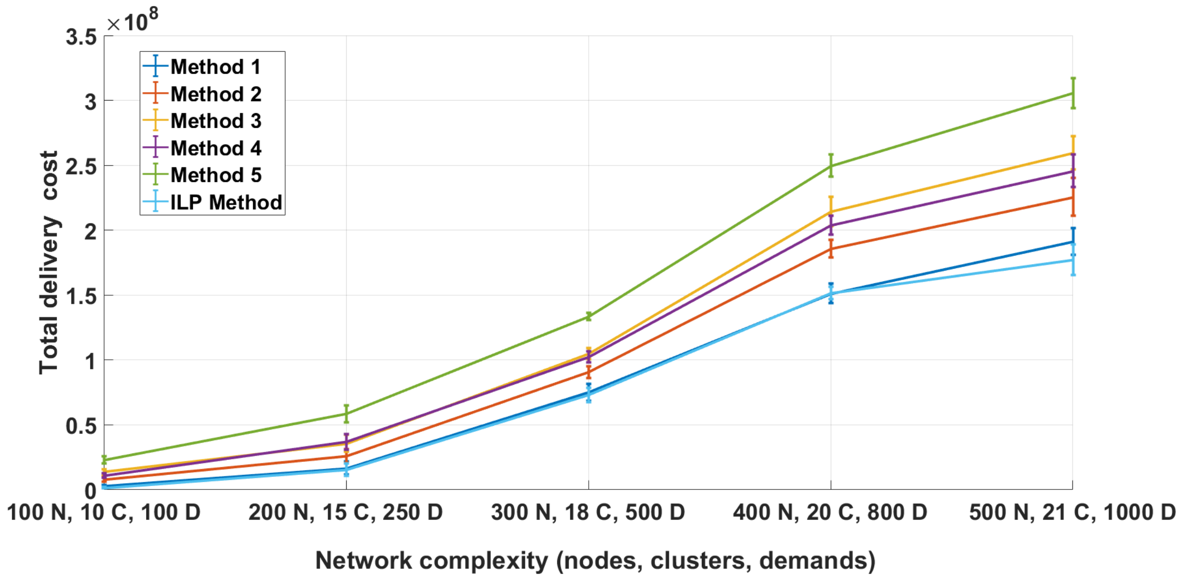

Figure 4.1: Comparisons of delivery costs for the five demand assignment methods under varying supply network complexities with respect to a baseline centralized optimization method.

Figure 4.1 compares the demand delivery costs of all the assignment methods against the baseline costs returned by the CPLEX ILP. For the unassigned demands, the maximum delivery costs are reported. Method 1 consistently performs the best by yielding costs that are statistically not different from the ILP values. This result shows that it is sufficient to split the demands into four equal parts for the networks to deliver all the demands that can be possibly satisfied with near-optimal costs. As the supply-demand network complexity grows, naturally, the total delivery cost increases. It is then worthwhile to note that all our methods perform equally well with respect to the baseline cost values regardless of the network complexity.

Figure 4.2 enumerates the computation times of all the demand assignment methods. We observe that the actual computation times are reasonable, ranging from a few seconds to a few minutes, even without code optimization leading to under-utilization of the available compute resources. Equally promisingly, the increase in computation times appears to be polynomial with respect to network complexity. Furthermore, the time values are similar for all the methods across all network complexities, with Methods 5 and 1 being the most and least efficient, respectively. These results are expected considering how the leftover demands are assigned in these two methods as compared to the rest. Also, as expected, the CPLEX ILP runs faster than our methods since it solves a simpler problem with no supplier communication constraints; encouragingly, the actual times are of the same order of magnitude.

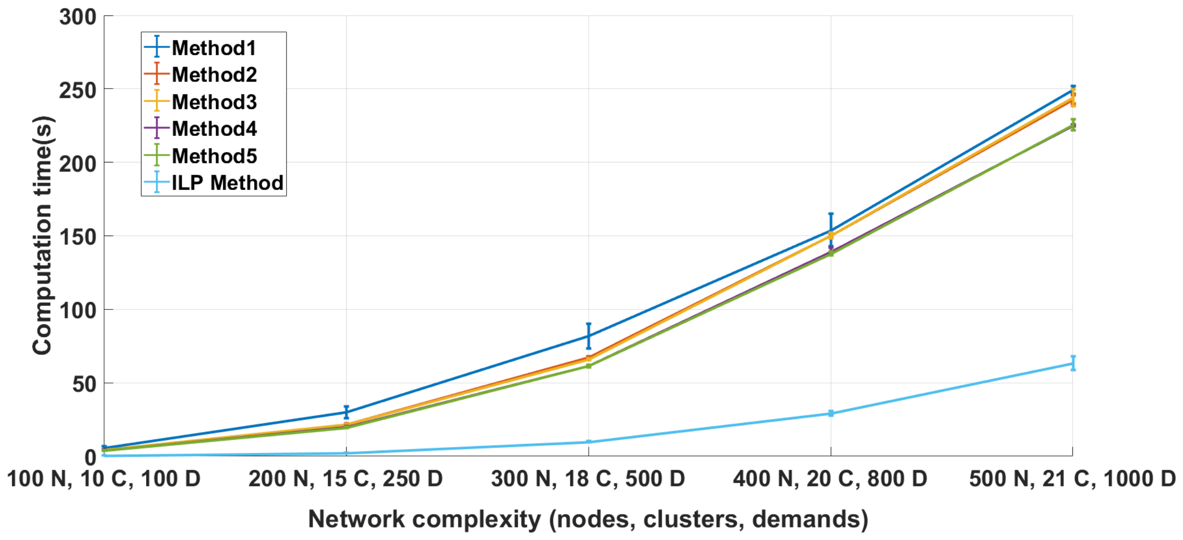

Figure 4.2: Comparisons of computation times for the five demand assignment methods under varying supply network complexities.

Figure 4.3 plots the percentage of assigned demands by the five methods with respect to the overall network capacity. As discussed before, Method 1 uses the entire network capacity to assign the demands by assuming that the agents are always interested in arriving at consensus even when the demands are split up. On the contrary, all the other methods show varying levels of under assignment of the demands, with Method 5 performing the worst by completely ignoring the leftover demands. It is also useful to point out that Method 2, employing the realistic model of assigning the leftover demands only among the interested suppliers, performs quite well in terms of all the evaluation metrics. This result is particularly

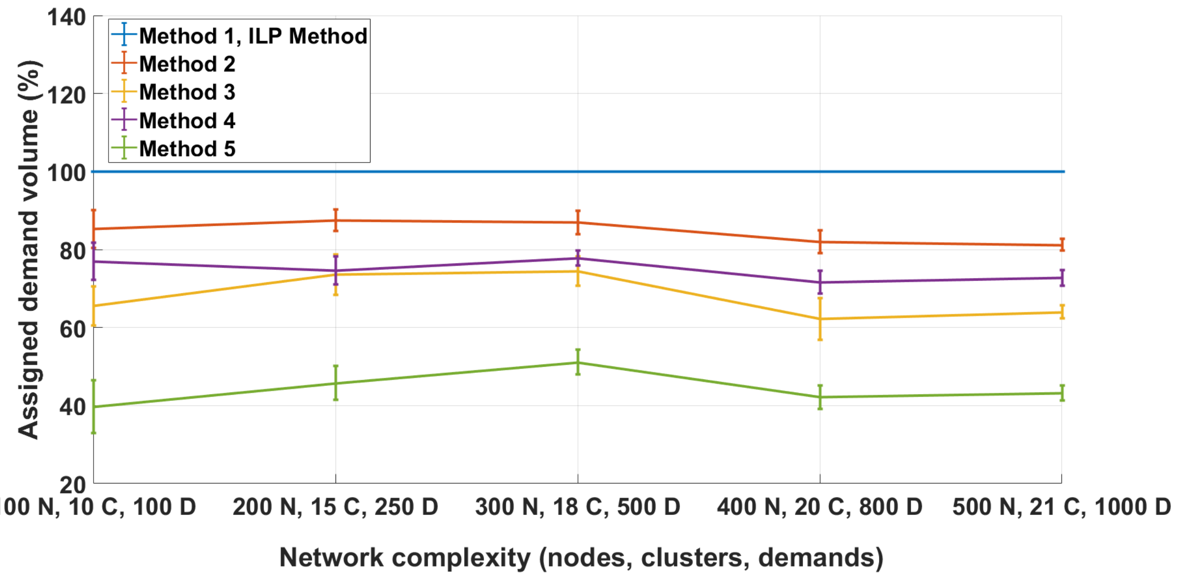

Figure 4.3: Comparisons of assigned demand volume proportions for the five different methods under varying supply network complexities.

promising in terms of providing an overall recommended method to use for the practitioners.

Figure4.4 characterizes the robustness of our approach to the network clustering scheme and variations in the number of demands. The results are reported only for Methods 1 and 2 to maintain presentation clarity. We observe that the delivery costs of both the methods are unaffected by changes in the number of clusters (and, therefore, the scale of the assignment problem in any cluster), for a given network size and a fixed set of demands. The delivery costs increase linearly with the number of demands for a fixed network size and clustering scheme, which again shows the capability of our methods to consistently generate near-optimal demand assignments.

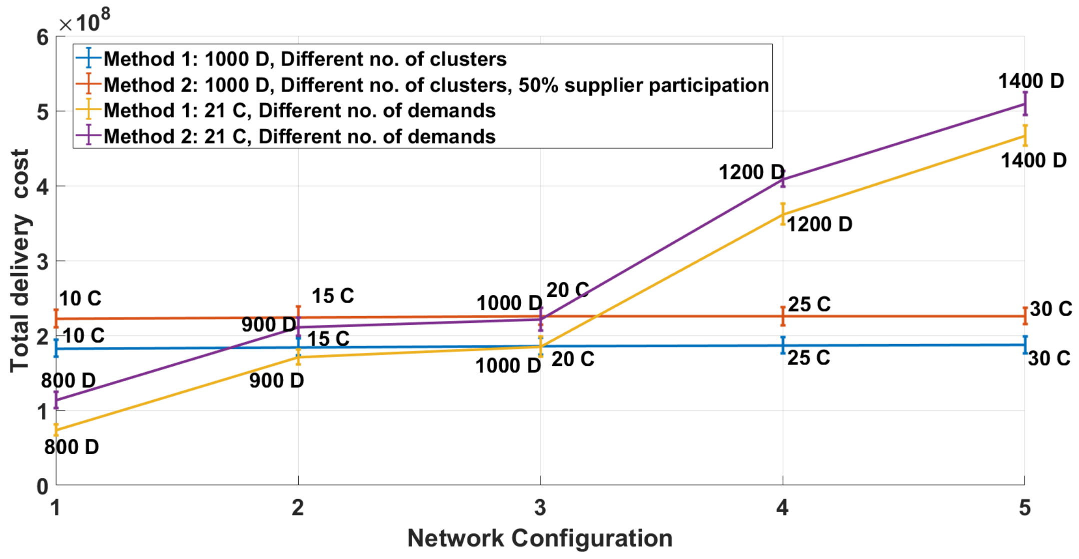

Figure 4.4: Comparisons of the total delivery cost using Methods 1 and 2 for 500 node networks with varying number of clusters (C) and demands (D).

To justify our choice of splitting demands at the beginning of the second stage to 4 parts and to show how the choice of suppliers' participation in the leftover demand assignment affects the total delivery cost, we tested Methods 1 and 2 on a 300 node network with 18 clusters and 500 demands. Method 1 (where all the suppliers participate in the leftover demand assignment) is used with different choices of K (from 2-6) and Method 2 is used with $K = 4$ and different supplier participation percentage ($30\% - 70\%$) within each cluster. Figure 4.5 shows that changing K does not affect the total delivery cost, and 50% participation gives comparable results to higher participation values. As expected, lower participation values yield slightly inferior results, but should be used if they model the network more accurately. We do not include this Figure in the manuscript for the sake of brevity, but summarize the results as footnote 4 in page 6.

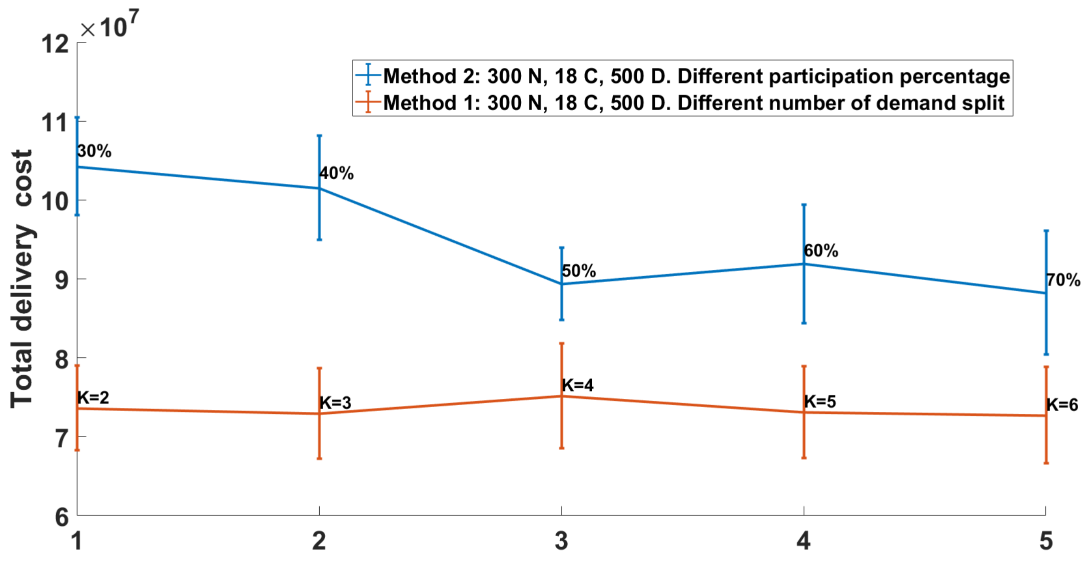

Figure 4.5: Comparison of delivery costs for different supplier participation percentages in Method 2 and different demand splits in Method 1.

Chapter 5
CONCLUSION

5.1 Conclusion

In this paper, we adapt consensus-based auction bidding methods for optimizing demands assignments to networked agents (suppliers) in a decentralized manner, where the assignments are made first to the supplier clusters and then to the individual suppliers within the clusters. Our methods explicitly account for cluster and supplier-level capacities (resources). They also consider splitting up the leftover (initially unassigned) demands into smaller volumes in a few different ways to be able to assign as much of the demands as possible at near-optimal costs within the constraint imposed by the overall network capacity. Simulation results on randomly generated moderate-sized networks show that the methods yield close-to-optimal assignments, successfully assign almost complete feasible demand volumes, are computed reasonably quickly, and are robust to network and model parameters.

We, therefore, believe that our methods would provide the foundations for multi-agent consensus optimization in networked systems with resource constraints. We also hypothesize that our methods are guaranteed to converge to solutions within some acceptable bounds of the optimal assignments. In fact, we should be able to build on the analogous proofs that exist for the CBBA method, although the extensions would be non-trivial to account for the two-stage assignment, resource constraints modeling, and demand volume splitting. Such extensions should be possible since we preserve the sequential nature of the consensus-building process, our delivery costs are non-negative, and bid selection is done by ranking the demands according to the maximum cost difference between the communicating agents, similar to the policy of assigning the tasks with maximum discounted rewards to the agents in the CBBA method.

5.2 Future Work

Future work includes modeling additional network characteristics such as the presence of multiple communication routes between two suppliers, supply flow capacity restrictions in the communication pathways, and time windows to deliver specific demands. Moreover, we plan to investigate the effects of network failures, as characterized by communication breakdowns and non-functional agents, on the computation time, quality, and robustness of the generated assignments. Similar investigation is planned to analyze the impacts of structural changes in networks due to the introduction and/or removal of suppliers or communication routes. Last, we intend to implement our methods on parallel computing paradigms for rapid assignment generation in the case of large-scale, dynamic supply networks with continuous inflows of demands. It would also be useful to model and solve the stochastic assignment problem where the demands, costs, capacities, failure modes and rates, and even the network structures vary randomly.

BIBLIOGRAPHY

[1] T. M. Hansen, R. Roche, S. Suryanarayanan, A. A. Maciejewski, and H. J. Siegel,, "Heuristic Optimization for an Aggregator-Based Resource Allocation in the Smart Grid," *IEEE Trans. Smart Grid*, vol. 6, no. 4, pp. 1785-1794, Jul. 2015.

[2] F. Zhu and S. V. Ukkusuri, "A Linear Programming Formulation for Autonomous Intersection Control within a Dynamic Traffic Assignment and Connected Vehicle Environment," *Trans. Res. Part C*, vol. 55, pp. 363-378, Jun. 2015.

[3] J. Veintimilla-Reyesa, D. Cattryssec, A. De Meyera, J. Van Orshovena, "Mixed Integer Linear Programming (MILP) Approach to Deal with Spatio-Temporal Water Allocation," *Procedia Eng.*, vol. 162, pp. 221-229, 2016.

[4] C. A. Floudas and X. Lin, "Mixed integer linear programming in process scheduling: Modeling, algorithms, and applications," *Ann. Oper. Res.*, vol. 139, no. 1, pp. 131-162, Oct. 2005.

[5] K. Ghorabaee, M. Maghsoud Amiri, E. K. Zavadskas, and J. Antucheviciene, "Supplier evaluation and selection in fuzzy environments: a review of MADM approaches," *Economic Res.-Ekonomska Istraživanja*, vol. 30, no. 1, pp. 1073-1118, May 2017.

[6] T. Sawik, "Supplier selection in make-to-order environment with risks," *Math. Comput. Model.*, vol. 53, no. 9-10, pp. 1670-1679, May 2011.

[7] E. E. Karsak, and M. Dursun, "An integrated fuzzy MCDM approach for supplier evaluation and selection," *Comput. Ind. Eng.*, vol. 82, pp. 82-93, Apr. 2015.

[8] K.Govindan, M. Fattahi, and E. Keyvanshokooh. "Supply chain network design under uncertainty: A comprehensive review and future research directions." European Journal of Operational Research 263.1 (2017): 108-141.

[9] C. Wu , D. Barnes . "A literature review of decision-making models and approaches for partner selection in agile supply chains". Journal of Purchasing and Supply Management. 2011 Dec 1;17(4):256-74.

[10] W. Ho, X. Xiaowei, and K. D. Prasanta. "Multi-criteria decision making approaches for supplier evaluation and selection: A literature review." European Journal of operational research 202, no. 1 (2010): 16-24.

[11] L. De Boer, , E. Labro, and P. Morlacchi. "A review of methods supporting supplier selection." European journal of purchasing & supply management 7, no. 2 (2001): 75-89.

[12] N. Aissaoui, M. Haouari, and E. Hassini. "Supplier selection and order lot sizing modeling: A review." Computers & operations research 34, no. 12 (2007): 3516-3540.

[13] M. S. Fox, M. Barbuceanu, and R. Teigen. "Agent-oriented supply-chain management." In Information-Based Manufacturing, pp. 81-104. Springer, Boston, MA, 2001.

[14] A. Singh, "Supplier evaluation and demand assignment among suppliers in a supply chain," *J. Purchasing Supply Manage.*, vol. 20, no. 3, pp. 167-176, Sep. 2014.

[15] T. Sawik. "On the fair optimization of cost and customer service level in a supply chain under disruption risks," *Omega*, vol. 53, pp. 58-66, Jun. 2015.

[16] D. Mohammaditabar, S. H. Ghodsypour, and A. Hafezalkotob, "A game theoretic analysis in capacity-constrained supplier-selection and cooperation by considering the total supply chain inventory costs," *Int. J. Prod. Econ.*, vol. 181, pp. 87-97, Nov. 2016.

[17] H. L. Choi, L. Brunet, and J. P. How, "Consensus-based decentralized auctions for robust task assignment," *IEEE Trans. Robot.*, vol. 25, no. 4, pp. 912-926, Jun. 2009.

[18] H. L. Choi, A. K. Whitten, and J. P. How, "Decentralized task assignment for heterogeneous teams with cooperation constraints," In *Proc. American Control Conf.*, Baltimore, MD, 2010, pp. 3057-3062.

[19] L. Johnson, S. Ponda, H. L. Choi, and J. P. How, "Improving the efficiency of a decentralized tasking algorithm for UAV teams with asynchronous communications," In *Proc. AIAA Guidance, Navigation, Control Conf.*, Toronto, ON, Canada, 2010, p. 8421.

[20] T. Mercker, David W. Casbeer, P. Travis Millet, and Maruthi R. Akella, "An extension of consensus-based auction algorithms for decentralized, time-constrained task assignment," In *Proc. American Control Conf.*, Baltimore, MD, 2010, pp. 6324-6329.

[21] D. Paola, D. Naso, and B. Turchiano, "Consensus-based robust decentralized task assignment for heterogeneous robot networks," In *Proc. American Control Conf.*, San Francisco, CA, 2011, pp. 4711-4716.

[22] D. Smith, J. Wetherall, S. Woodhead, and A. Adekunle, "A cluster-based approach to consensus based distributed task assignment," In *Proc. Euromicro International Conf. Parallel, Distributed and Network-Based Processing*, Torino, Italy, 2014, pp. 428-431.

[23] B. Jia, K. D. Pham, E. Blasch, D. Shen, and G. Chen, “Consensus-Based Auction Algorithm for Distributed Sensor Management in Space Object Tracking,” In *Proc. IEEE Aerospace Conf.*, Big Sky, MT, 2017, pp. 1-8.

[24] E. Nunes and M. L. Gini, “Multi-robot auctions for assignment of tasks with temporal constraints,” In *Proc. National Conf. Artificial Intelligence*, Austin, TX, 2015, pp. 2110-2116.

[25] M. M. Zavlanos, L. Spesivtsev, and G. J. Pappas, “A distributed auction algorithm for the assignment problem,” In *Proc. IEEE Conf. Decision Control*, Cancun, Mexico, 2008, pp. 1212-1217.

[26] R. Chen, Y. C. Chen, W. Guo, and A. G. Banerjee, “A note on community trees in networks,” presented at the *Advances Neural Information Processing Conf. Workshop on Synergies in Geometric Data Analysis*, Long Beach, CA, 2017.

[27] R. Olfati-Saber, J. A. Fax, and R. M. Murray, “Consensus and cooperation in networked multi-agent systems,” *Proc. IEEE*, vol. 95, no. 1, pp. 215-233, 2007.

[28] https://www.ibm.com/products/ilog-cplex-optimization-studio, accessed on March 23, 2018.

Appendix A

INTEGER LINEAR PROGRAMMING (ILP) FORMULATION

The ILP formulation for assigning the demands to the clusters without considering the communication pathways among the suppliers is given in (A.1). c_{pj} is the delivery cost for demand D_j provided by cluster p, and α_{pj} is the number of resources cluster p would provide for demand D_j. The ILP minimizes the total delivery cost with respect to α. The constraints are defined as: 1) for each cluster p, the sum of the provided resources for all the demands, $\sum_{j=1}^{O(t)} \alpha_{pj}$, should be less than or equal to the cluster p's capacity R_p; 2) the total number of resources assigned to each demand, $\sum_{p=1}^{P} \alpha_{pj}$, should be less than or equal to the demand volume $r(D_j)$; 3) the total number of resources assigned to the demands should be the minimum of the network's capacity or the sum of the demand volumes. The same formulation is used to assign the demands to the suppliers within each cluster by simply replacing the clusters with the suppliers.

$$\begin{aligned}
\min_\alpha \quad & \sum_{p=1}^{P}\sum_{j=1}^{O(t)} c_{pj}\cdot\alpha_{pj} \\
\text{subj. to} \quad & 0 \le \sum_{j=1}^{O(t)} \alpha_{pj} \le R_p \quad \forall p \in \{1,\dots,P\} \\
& 0 \le \sum_{p=1}^{P} \alpha_{pj} \le r(D_j) \quad \forall D_j \in \{D_1,\dots,D_{O(t)}\} \\
& \sum_{p=1}^{P}\sum_{j=1}^{O(t)} \alpha_{pj} = \min\left\{\sum_{j=1}^{O(t)} r(D_j), \sum_{p=1}^{P} R_p\right\}
\end{aligned} \tag{A.1}$$