An Investigation of Distributed Constraint Optimization with Non-Responding Agents Toward Real-Time Solution Method on Practical Messaging Platforms

Toshihiro Matsui[a]
Nagoya Institute of Technology, Gokiso-cho Showa-ku Nagoya Aichi 466-8555, Japan

Keywords: Distributed Constraint Optimization, Message Loss, Publish and Subscribe, Multiagent System.

Abstract: Distributed constraint optimization problems have been studied as the bases of decentralized resource allocation and decision making on multiagent systems. The studies address constraint optimization problems distributed on multiple agents and decentralized solution methods. A number of types of solution methods based on different optimization techniques have been proposed, and various applications have been investigated, including resource allocation for sensor networks and power grids, and collaboration for meeting scheduling and disaster responses. One issue with implementation is the low communication quality in some cases of actual devices. A recent study addressed the delay and loss of messages in local search methods. On the other hand, opportunities exist for considering further practical implementation techniques. For example, in real-time systems, a solution process might be required to complete an optimization cycle even if several agents do not respond temporally. In this paper, we investigate fundamental implementation techniques toward distributed constraint optimization methods on a message-oriented communication platform based on the publish/subscribe model, which is generally employed for IoT devices and robot systems. In our first study, we address the deterministic local search method on the low QoS settings of communication environments with message loss, where the solution method must continue with temporally missing agents. We experimentally evaluate the influence of several approaches to handle temporally non-responding agents in the executing solution processes.

1 INTRODUCTION

Optimization problems and solution methods on multiagent systems have been studied as the bases for decentralized resource allocation and decision making. Research on distributed constraint optimization problems (Yeoh and Yokoo, 2012; Fioretto et al., 2018) address the constraint optimization problem that is distributed among multiple agents and decentralized solution methods. Various applications, including resource allocation for sensor networks (Zhang et al., 2005; Zivan et al., 2009; Matsui, 2020) and power grids (Kumar et al., 2009; Matsui and Matsuo, 2012) as well as collaboration for meeting scheduling (Maheswaran et al., 2004b) and disaster response (Ramchurn et al., 2010), require such a general representation of problems and distributed solvers as foundations of substantial cooperative problem-solving methods.

[a] https://orcid.org/0000-0001-8557-8167

Various types of solution methods based on different optimization techniques have been proposed. Such solution methods are categorized into complete methods to find optimal solutions (Modi et al., 2005; Petcu and Faltings, 2005; Yeoh et al., 2008) and incomplete methods for quasi-optimal solutions (Maheswaran et al., 2004a; Pearce et al., 2008; Zhang et al., 2005; Ottens et al., 2012; Nguyen et al., 2019; Mahmud et al., 2020; Choudhury et al., 2020). In particular, relatively simple local search methods have been employed for prototyping several applications, since they are relatively easily adjusted for practical implementation (Zivan et al., 2009; Fioretto et al., 2017; Matsui, 2020).

One of implementation issues is the low communication quality of some actual devices. A recent study (Rachmut et al., 2022), which addressed the delay and loss of messages in local search methods, described frameworks to handle delayed messages and investigated the influence of delay and loss of messages in fundamental local search methods.

Matsui, T.
An Investigation of Distributed Constraint Optimization with Non-Responding Agents Toward Real-Time Solution Method on Practical Messaging Platforms.
DOI: 10.5220/0012471000003636
Paper published under CC license (CC BY-NC-ND 4.0)
In *Proceedings of the 16th International Conference on Agents and Artificial Intelligence (ICAART 2024) - Volume 1*, pages 428-437
ISBN: 978-989-758-680-4; ISSN: 2184-433X
Proceedings Copyright © 2024 by SCITEPRESS – Science and Technology Publications, Lda.

On the other hand, opportunities also exist for considering further practical implementation techniques. For example, real-time systems might require a solution process to complete optimization cycles even if several agents are not responding temporally. Such agents resemble temporally uncontrolled agents in the solution process. In different studies of the class of problems containing uncontrolled agents, such agents are considered with boundaries of benefits or risks, and such boundaries are optimized (Rogers et al., 2011; Okimoto et al., 2011; Rogers et al., 2011; Matsui et al., 2010; Matsui, 2023). This approach can be applied to a solution process with temporally missing agents.

In this paper, we investigate fundamental implementation techniques toward distributed constraint optimization methods on a message-oriented communication platform based on the publish/subscribe model that is generally employed for IoT devices and robot systems. We address a deterministic local search method on the low QoS settings of a communication environment with message loss, where the solution method must continue with temporally missing agents. We employ a minimal environment that simulates such communication settings and investigate several approaches to handle temporally non-responding agents for executing solution processes. We also experimentally evaluate the influence of our proposed approach.

The rest of this paper is organized as follows. Next, we present the background of our study, including distributed constraint optimization problems, a publish/subscribe communication model with low QoS settings, a related work of local search methods with incomplete messaging, related works on uncontrollable agents, and the aim of this study. In Section 3, we present our proposed methods and consider the basic implementation of solution methods as well as several approaches to handle temporally non-responding agents in executing solution methods. We experimentally investigate these approaches in Section 4 and conclude in Section 5.

2 PRELIMINARIES

2.1 Distributed Constraint Optimization Problem

A distributed constraint optimization problem (DCOP) (Yeoh and Yokoo, 2012; Fioretto et al., 2018) is a fundamental combinatorial optimization problem in multiagent systems. It is defined by $\langle A, X, D, F \rangle$, where

- A is a set of agents,
- X is a set of discrete variables,
- D is a set of domains of the variables, and
- F is a set of functions that represent the evaluation values of the constraints[1].

In general settings, variable $x_k \in X$ takes a value from corresponding domain $D_k \in D$, and function $f_k \in F$ defines non-negative evaluation values $f_k(X_k)$ s.t. $X_k \in X$ for assignments to the variables in X_k. The globally optimal assignment to all the variables maximizes the total evaluation value aggregated for all functions. Agent $a_i \in A$ has its own set of variables $X_i \in X$ that represents the decision or state of the agent, and a_i determines the assignment to its own variables in X_i. Each function represents the relationship among neighborhood agents who have variables related to the function. Each agent a_i knows the information of its variables X_i, the functions in $F_i \in F$ related to the variables, and the neighborhood agents' variables related to functions in F_i. N_i denotes the set of a_i's neighborhood agents.

As a standard setting without loss of generality, we focus on a simple case where agent a_i has a single variable x_i, and all the functions are binary functions $f_{j,k} : D_j \times D_k \to \mathbb{N}_0$. $f_{j,k}$ is denoted by $f_{j,k}(d_j, d_k)$. This problem can be depicted using a constraint graph where the nodes and edges represent agents/variables and functions. Here globally optimal solution $\mathcal{A}^*$ is represented as $\mathcal{A}^* = \arg\max_{\mathcal{A}} \sum_{f_{i,j} \in F} f_{i,j}(\mathcal{A}_{\downarrow x_i}, \mathcal{A}_{\downarrow x_j})$.

Agents search for the optimal assignment to their own variables in a decentralized manner with their related agents.

The solution methods for DCOPs (Fioretto et al., 2018) are categorized into complete methods (Modi et al., 2005; Petcu and Faltings, 2005; Yeoh et al., 2008) for optimal solutions and incomplete methods (Maheswaran et al., 2004a; Pearce et al., 2008; Zhang et al., 2005; Ottens et al., 2012; Nguyen et al., 2019; Mahmud et al., 2020; Choudhury et al., 2020) for quasi-optimal solutions. Since complete methods cannot be applied to densely constrained large-scale problems without approximation, incomplete solution methods have been developed. Although various sophisticated solution methods can be found, simple local search methods, where each agent exchanges its information with neighborhood agents, is practical in

[1]We note that DCOP has been derived from the distributed constraint satisfaction problem (Yokoo et al., 1998) to handle soft constraints, and in a fundamental definition, hard constraints are implicitly represented with special objective values such as $-\infty/\infty$ for maximization/minimization problem.

the first step modeling prototypes of application systems (Zivan et al., 2009; Fioretto et al., 2017; Matsui, 2020). We focus on a fundamental local search algorithm (Maheswaran et al., 2004a; Pearce et al., 2008) for a case study toward an implementation of solution methods in a practical communication platform with message loss.

2.2 Publish/Subscribe Communication Model and Message Loss

Publish/subscribe (pub/sub) models are employed in message oriented communication platforms, including MQTT[2] for IoT devices and DDS[3] in ROS2 for robots. Communication systems consist of the following elements.

- Publisher: distributes messages to a communication system without an awareness of the receivers.
- Subscriber: receives related messages from a communication system.
- Broker: transfers messages to subscribers. Several systems have no explicit brokers and deploy multi-cast communication.

We consider a pub/sub model with topics, where a publisher sends messages on specific topics, and subscribers sign up for messages related to required topics. The quality-of-service (QoS) of communication levels is generally defined in practical pub/sub models. Communication without message loss is assured with high QoS levels, although it requires relatively high communication overhead. On the other hand, a low QoS might cause message loss in exchange for reduced communication overhead. We assume a simple system without brokers and a low QoS for real-time situations.

2.3 Local Search Methods and Communication Quality Settings

We focus on a fundamental local search algorithm: maximum gain messages (MGM) (Maheswaran et al., 2004a; Pearce et al., 2008). Here neighborhood agents repeat synchronous communication steps with each other after sharing their initial solutions (lines 1 and 2 in Figure 1). First, each agent a_i locally searches for the best assignment to its own variable x_i under its view of the partial assignments received from its neighborhood agents in N_i, computes the gain

```
1 SendValueMessage($N_i$, $x_i$)
2 currentContext = GetValueMessages($N_i$)
3 [gain,newValue] = BestUnilateralGain(currentContext)
4 SendGainMessage($N_i$, gain)
5 neighborGains = ReceiveGainMessages($N_i$)
6 **if** gain $>$ max(neighborGains) **then**
7     $x_i$ = newValue
```

Figure 1: MGM (Pearce et al., 2008) in each agent a_i.

of the locally aggregated function values for the view (line 3), and multi-casts the gain to the neighborhood agents (line 4). Each agent receives the gain values from all of the neighborhood agents (line 5). If its gain is largest among its neighborhood agents (line 6), the agent commits to its new assignment (line 7), and multi-casts the update to its neighborhood agents (line 1). A tie of gains is broken using the names of agents (line 6). The process monotonically updates global evaluation values and corresponding solutions, and it converges into a one-optimal solution. This process is also an anytime algorithm. Although this is the simplest solution method for DCOPs, various studies have employed it for fundamental analyses and prototypes of application systems. We also use this method for our first case study toward the implementation of solution methods in a practical communication platform with message loss.

There are several studies of solution methods for DCOPs with incomplete communication environments (Rachmut et al., 2022). A recent study (Rachmut et al., 2022) investigated the influence of message delay and loss for local search algorithms. It presented frameworks for local search methods with logical time stamps or contexts to handle message delays, while it is assumed that message loss can be mitigated by resending messages. Another investigation adjusted local search methods, including MGM and distributed stochastic algorithm (DSA), to handle message loss with an asynchronous process and experimentally concluded that the perturbation caused by message loss might improve the search path and the resulting solution quality. This study mainly addresses an asynchronous approach to handle message delay and considers message loss that is handled with asynchrony and resending. On the other hand, in real-time situations, there might be opportunities to temporally ignore non-responding agents, a situation that forces solution methods to be continued.

2.4 DCOPs with Uncontrollable Agents

There are several studies of DCOPs with uncontrollable agents. A basic case is the approximation of densely constrained problems, where several

[2]MQTT: The Standard for IoT Messaging (https://mqtt.org/)

[3]ROS on DDS (https://design.ros2.org/articles/ros_on_dds.html)

functions are replaced by low arity ones (Rogers et al., 2011; Okimoto et al., 2011). In a previous work (Rogers et al., 2011), the functions are approximated so that the graph structure representing the original problem is modified to a spanning tree by eliminating several edges. Here the impact of the edges is considered, and functions can be modified by aggregating the boundaries of the evaluation values for the eliminated edges.

For an extended class of problems with adversarial agents that might decrease the global solution quality, non-adversarial agents cooperatively optimize solutions assuming that the adversarial agents choose the worst case solution (Matsui et al., 2010; Matsui, 2023). The worst cases can also be approximated with the boundaries of the evaluation values for adversarial agents to optimize the problems based on worst case values.

Although the risks or expectation of uncontrolled agents are generally considered in the problem settings themselves with robustness or resilience, there might be opportunities to apply them to solution processes for real-time systems with non-responding agents.

2.5 Aim of Study

We are investigating the implementation techniques of solution methods on practical communication platforms based on pub/sub models. That includes several detailed tunings of message communication timings and synchronization logics. Particularly in low QoS settings that can be employed with mobile robots with wireless multi-cast communication, the situations where some publisher agents might be temporally missing due to collisions or obstacles on the communication systems. In real-time systems, a solution process might be required to continue its execution by ignoring such missing agents.

In this situation, a possible approach is to evaluate such missing agents as uncontrolled ones. We investigate the effect of this approach. As our first case study, we address the simplest deterministic local search method; we will address complicated cases with additional communication steps in a future study.

3 LOCAL SEARCH WITH MISSING AGENTS

3.1 Modeling Communication Platform

We are motivated to develop frameworks and implementation techniques for solution methods for DCOPs on practical communication platforms based on pub/sub models. Although a practical implementation must contain the input of sensing data, the initialization and execution of a solution method, and the output to an environment, we first study a solution method that is adjusted to a communication model. As preparation, we build a minimal simulation environment and investigate the solution methods. We assume that agents have the information about their neighborhood agents related to their common evaluation functions. Such information might be provided by employing additional services to register it. Each agent publishes messages and subscribes to them from its neighborhood agents. Although a typical implementation of IoT devices and cloud servers employs centralized processing that collects information from edge devices and gives commands to such devices, we focus on cooperative problem solving by edge devices.

Several solution methods for DCOPs are based on a communication system with message queuing. To adjust such methods to the pub/sub model, some management of communication that resembles distributed shared memories is necessary. To reduce the processing and communication loads, relatively light and incomplete solution methods are practical. Particularly in the case of low QoS communication systems, robustness for message loss is required. We focus on these points and investigate related issues. We address a simple setting where each published multi-cast message in each time step is either transferred or lost; delayed or corrupted messages are ignored.

3.2 Implementation of Fundamental Local Search Method

We implement a deterministic local search that is based on MGM. After a first step to share the initial solution, the solution process repeats its phases. Each phase of an agent consists of the computation of information to be published, publishing that information, and collecting subscribing information from all of the neighborhood agents. The phases are integrated using a barrier synchronization by received messages, while no agent knows whether its published messages have been transferred.

Each agent has a view, which is a cache of the information from each neighborhood agent. When a subscribing message is received, its corresponding information is updated. A view for a neighborhood agent consists of the following kinds of information.

- The current iteration of the solution process in the neighborhood agent.

- The current phase in the current iteration of the solution process in the neighborhood agent.
- The current assignment to the neighborhood agent's variable.
- The current gain of the local evaluation in the neighborhood agent.

Each message also consists of similar information, and the assignment or gain value is exclusively carried depending on the corresponding phase value.

When a message is received, a pair of logical time stamps for iterations and phases is checked with those of the subscriber agents. If a pair of iteration and phase matches the information in a view, the information is updated with the message. Otherwise the message is ignored.

The followings are the detailed phases in each iteration.

1. Initialization of solution: Each agent a_i randomly selects an assignment to its own variable x_i and publishes the assignment. After the subscribing messages for the current iteration and phase are received from all the neighborhood agents in N_i, a_i moves to the next phase.
2. Evaluation of local gain: Each agent a_i locally searches for the best solution under the partial solution in its current view, and evaluates the gain of the evaluation value aggregated for its related evaluation function. The gain is published. After the subscribing messages for the current iteration and phase are received from all the neighborhood agents, a_i moves on to the next phase.
3. Commitment of the best assignments: Each agent checks whether it is the winning agent whose gain exceeds that of any other agents in the current view received from the neighborhood agents. The winning agent updates its best assignment to its own variable x_i. Then the current assignment to x_i is published. Although the information of the unchanged assignments is redundant, we prefer that all agents publish their current assignment for a simple handshake among agents and for room to adjust the solution process. After the subscribing messages for the current iteration and the current phase are received from all the neighborhood agents, a_i moves to the next iteration and the second gain evaluation phase, skipping the first initial solution phase.

We separately implement the computation, publishing, and subscribing steps in each phase as callback procedures that are repeatedly executed by the whole system. The computation, publishing and subscribing steps can be repeatedly called in different frequencies, and barrier synchronization, which is waiting for all the messages from the neighborhood agents, is applied to the end of each phase.

3.3 Resending Messages

In an environment with message loss, resending the same message is the basic approach. To simulate this situation, the callback for publishing and subscribing can be frequently called than that for the computation of the solution process. However, since we address a real-time system, re-publishing messages are limited up to the maximum times of retries. When some subscribing messages are not received, a barrier synchronization at each phase cannot be achieved, and the solution process in such an agent gets stuck. On the other hand, an agent that received messages from all the neighborhood agents might move to the next phase. Due to such situations the solution process of the entire system does not consistently progress. To avoid such situations, sufficient re-publishing attempts are required.

3.4 Default Values Aggregated from Agent Views

The information in the view of each agent should represent inconsistent situations due to missing messages. The information of each agent's view is initially empty and default values are set. We set the default values of iteration and phase to common initial values, which are incrementally updated with received messages. For the default assignment to each variable, we use an empty value, which also represents the situation of non-responding agents. We also use an empty value for the default gain value. When the default gain value is used to represent non-responding agent, it can be used to exclude missing agents from the comparison of gain values.

If the solution process only considers the current information, the old information in the views with the old pairs of iteration and phase values should be ignored. In an exception case for MGM, the assignment of the agent that does not commit its new best assignment is still valid, and some peers can observe this situation without any subsequent communication. For such cases, old assignments can be inherited as a backup for a missing (but substantially redundant) subsequent message.

3.5 Excluding Agents with Message Loss

When a message from a neighboring agent is lost, other neighborhood agents ignore the agent and continue their solution process to avoid a lock of system. Each agent a_i ignores a neighborhood agent after a threshold number of subscription-retries, and the ignored agent is marked and excluded from the cooperation in a_i. Here the ignored agent itself is not aware this situation due to the communication system's settings.

Each agent, who is ignoring the missing neighborhood agents, continues the process by collecting available information from the other neighborhood agents. In this case, part of the information is missing in an agent's view for the current iteration and phase. In the phase of the local gain evaluation, each agent evaluates the partial solution, including the missing assignment. We investigate the following approaches.

- Inherit: Each agent keeps its view for the last received assignment and employs it by ignoring the consistency of its related iteration and phase. This can be considered a natural adaption of MGM to the situation.
- LB: For an assignment of missing neighborhood agent a_j and the own assignment of a_i, the lower bound values of the related evaluation function $f_{i,j}(d_i,d_j)$ are employed by approximating the binary functions to the unary function $f_i(d_i) = \min_{d_j} f_{i,j}(d_i,d_j)$. Such agent a_i performs a maximin optimization, which is a relatively pessimistic approach.
- UB: It resembles an LB, although the upper bound values of the evaluation functions are employed. This is a relatively greedy approach.

In a basic setting that emphasizes the situation of the missing agents, once an agent ignores a neighborhood agent a_j, it does not trust the communication quality of a_j and continues to ignore a_j.

3.6 Accepting Committed Solutions from Ignored Agents

The previous section introduced ignored agents to continue the solution process by admitting the missing agents. When a new message is received from an ignored agent, such a message might be based on different information.

On the other hand, the committed/current assignment to the neighborhood agents' variables improves the accuracy of the evaluation values in the local search even if the agents are still ignored and excluded from cooperation. We investigate the robustness in this partially corrupted solution process.

3.7 Rejoin to Cooperation

Ignored agents can return to the solution process at an appropriate timing. For simplicity, we assume a recovery is tried at a commitment phase that can be a reset timing of an agents' view. When an agent a_i receives a commitment message from its ignored agent, it checks whether the iteration and phase pair in the message is consistent with that in a_i. If that is consistent, the subscriber agent accepts the message, updates its related view, and removes the publisher agent from its list of ignored agents. The timestamps of the iteration and phase are not related to the context of the assignment of variables, and a publisher agent simply rejoins without other conditions.

When an agent observes that its current time stamp is older than those of the other agents, the delaying agent can update its time stamp to the newest one by locally adjusting its solution process, and it can catch up with the other agents. Actually in our experiments, since the agents separately update their phases almost at the same interval, we mainly focus on the influence of the message loss. Our future work will investigate such delays.

4 EVALUATION

4.1 Settings

We experimentally evaluated our proposed approach. The following typical settings were selected from our various preliminary experiments. The benchmark problems consist of 50 agents/variables and randomly generated c binary constraints/functions. Each variable takes a value from its domain whose size is three.

We employed the following types of evaluation functions.

- MATCHING: A function value is set by random integer values in $[1,10]$ based on uniform distribution for the same assignment to a pair of variables, while it takes zero for different assignments.
- RANDOM: The function values are set by random integer values in $[1,10]$ based on uniform distribution.

Intuitively, MATCHING resembles a weighted vertex coloring problem for minimization problems, but it is solved as a maximization problem. We employed this problem setting to show a result different from other cases. We confirmed that the result of vertex coloring problems with random weight values for maximization problems resembled that of RANDOM.

Table 1: Result without message loss (BASE).

prb.	c	utility	last update	
			#iter.	#sys. cyc.
MATCHING	49	272.5	2.1	12.0
	100	497.4	5.0	23.6
	150	795.5	8.1	35.8
RANDOM	49	404.2	2.7	14.8
	100	743.4	4.4	21.6
	150	1072.7	6.1	28.3

We evaluated the following settings of the solution methods.

- BASE: a baseline method that does not consider message loss.
- IGNORE: a method that continues to ignore the neighborhood agents with message loss, even if the agents can communicate after that.
- RCVCMT: In addition to IGNORE, commitment messages from ignored agents are accepted.
- REJOIN: In addition to RCVCMT, commitment messages are treated as the rejoin to cooperation.

We compared the evaluation approaches for missing assignments (shown in Section 3.5), and combined them with the default assignment values in the agent's views (discussed in Section 3.4) as follows.

- Inherit: The initial assignments in the agents' views were shared by a simulator by assuming an additional mechanism, and the initialization phase was skipped.
- LB, UB: Each agent a_i resets the assignments in its view for the neighborhood agents who have larger gain values than a_i, because such neighborhood agents must report whether they update their assignment by commitment messages.

We investigated several settings of communication environments that emphasize specific situations:

- The frequency of callbacks for computation, publishing, and subscribing steps in each phase.
- The number of agents with low communication quality and their message missing rates.
- The threshold number of subscribing retries before ignoring the missing agents.

The simulation was terminated at 100 system cycle to focus on the convergence of solution quality in early steps. We evaluated the quality of the final solution and the last iteration of the solution updates in the entire system. Each result was averaged over ten problem instances and ten trials with different random initial solutions.

Table 2: Convergence of solution for ratio of publishing steps per computation step (BASE, RANDOM, one low-communication-quality agent with message loss rate 0.5).

c	#pub. / #iter.	utility	last update	
			#iter.	#sys. cyc.
49	1	374.3	1.9	11.5
	5	**400.6**	2.6	71.8
	10	392.5	1	80
100	1	609.8	0.2	4.9
	5	**719.0**	3.0	79.4
	10	693.8	1	80
150	1	855.9	0.0	3.6
	5	**1021.1**	3.0	79.8
	10	977.1	1	80

4.2 Results

Table 1 shows the baseline result without any message loss. Here #iter is the number of iterations in the solution methods, and "#sys. cyc." is the number of system cycles in the simulator that executes the callback procedures of a solution method. The presented values are those in the last solution update. We set parameters so that each callback procedure of the solution methods is called once in each system cycle.

Table 2 shows the convergence of the solution for each ratio of publishing callbacks per computation callback in the baseline method. Here the message loss rate was set to 0.5 for one agent to emphasize the message loss. With a larger number of publishing retries, the iteration increased in average, and it reveals that the stuck situations decreased. In addition, there is a trade-off between the retry and the progress of solution process. However, such retry settings are experimental and agents became got stuck cannot continue their solution process.

Table 3 shows the result of a case of ignored agents due to message loss. Here the message loss rate was set to 0.5 for 25 agents. The same frequency was set for the computation, publishing and, subscribing callback procedures of the solution method. The threshold of publishing retries before a missing agent was ignored was set to once. "#ignored nbr. at last" is the number of neighborhood agents that were ignored by each agent in the cutoff system cycle. Although the solution method continued, the quality of the solutions decreased more than the baseline result in Table 1.

Regarding the evaluation approaches for missing assignments to the variables in the agents' views, UB that maximizes the best case found slightly better solutions than others in the cases of (MATCHING, $c = 49$ and $c = 100$) and RANDOM in average. This can be considered that each evaluation function is symmetric, and agents' local gains are not so opposed. In the cases of (MATCHING, 150), the result of LB was better. We note that LB is also competi-

Table 3: Result with ignored agents(IGNORE, 25 low-communication-quality agents with message loss rate 0.5).

prb.	*c*	lcl. eval.	utility	last update		#ignored. nbr. at last		
				#iter.	#sys. cyc	min.	ave.	max
MATCHING	49	Inherit	231.3	1.5	7.6	0	1.0	3.5
		LB	243.2	1.8	11.2	”	”	”
		UB	**256.7**	1.9	11.4	”	”	”
	100	Inherit	414.6	2.9	13.5	0	2.0	5.5
		LB	424.2	2.8	15.3	”	”	”
		UB	**424.3**	3.0	15.9	”	”	”
	150	Inherit	614.4	3.6	16.2	0.3	3.0	7.1
		LB	**653.7**	3.5	18.0	”	”	”
		UB	612.2	3.7	18.9	”	”	”
RANDOM	49	Inherit	380.9	1.9	9.5	0	1.0	3.5
		LB	378.1	2.1	12.5	”	”	”
		UB	**396.7**	2.1	12.4	”	”	”
	100	Inherit	705.6	3.1	14.2	0	2.0	5.5
		LB	708.2	3.1	16.4	”	”	”
		UB	**721.9**	3.2	16.9	”	”	”
	150	Inherit	1022.8	3.8	17.4	0.3	3.0	7.1
		LB	1020.7	3.8	19.2	”	”	”
		UB	**1041.4**	4.1	20.4	”	”	”

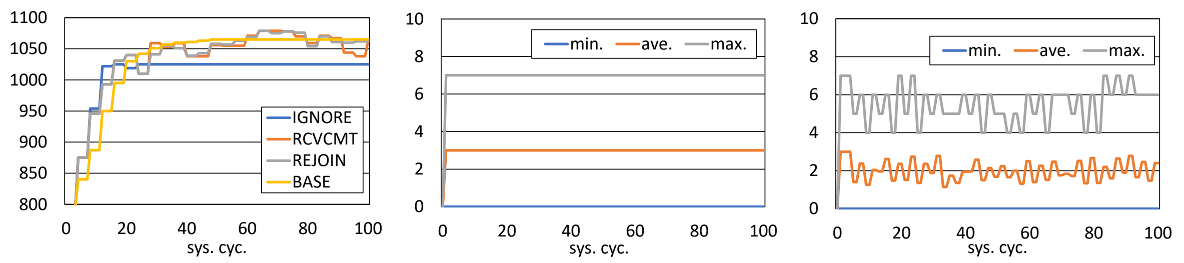
Table 4: Result with ignored agents(RCVCMT, 25 low-communication-quality agents with message loss rate 0.5).

prb.	*c*	lcl. eval.	utility	last update		#ignored. nbr. at last		
				#iter.	#sys. cyc	min.	ave.	max
MATCHING	49	Inherit	270.6	3.9	17.4	0	1.0	3.5
		LB	267.0	22.3	93.2	”	”	”
		UB	**278.5**	23.8	99.4	”	”	”
	100	Inherit	503.9	7.7	32.6	0	2.0	5.5
		LB	**528.0**	21.6	90.3	”	”	”
		UB	510.3	23.8	99.0	”	”	”
	150	Inherit	783.4	9.0	37.8	0.3	3.0	7.1
		LB	**818.4**	16.7	70.8	”	”	”
		UB	801.7	22.5	94	”	”	”
RANDOM	49	Inherit	403.7	4.2	18.9	0	1.0	3.5
		LB	399.2	23.9	99.7	”	”	”
		UB	**407.9**	23.8	99.3	”	”	”
	100	Inherit	**745.2**	6.1	26.2	0	2.0	5.5
		LB	737.9	24.0	99.96	”	”	”
		UB	744.4	23.9	99.7	”	”	”
	150	Inherit	1073.2	7.6	32.2	0.3	3.0	7.1
		LB	1066.8	24	100	”	”	”
		UB	**1075.4**	24.0	99.9	”	”	”

Table 5: Result with ignored agents(REJOIN, 25 low-communication-quality agents with message loss rate 0.5).

prb.	*c*	lcl. eval.	utility	last update		#ignored. nbr. at last		
				#iter.	#sys. cyc	min.	ave.	max
MATCHING	49	Inherit	271.1	4.0	17.6	0	0.5	2.5
		LB	267.1	21.9	91.8	”	0.8	3.0
		UB	**279.3**	23.9	99.4	”	”	”
	100	Inherit	503.3	8.0	33.6	0	1.0	3.7
		LB	**529.8**	21.2	88.8	”	1.6	4.6
		UB	513.8	23.8	99	”	”	”
	150	Inherit	783.8	9.1	38	0	1.5	4.6
		LB	**817.1**	17.0	72	0.05	2.3	5.9
		UB	803.7	22.5	94.0	”	”	”
RANDOM	49	Inherit	403.8	4.5	20.2	0	0.5	2.5
		LB	399.7	24.0	99.8	”	0.8	3.0
		UB	**408.7**	23.8	99.2	”	”	”
	100	Inherit	744.5	6.2	26.7	0	1.0	3.7
		LB	738.2	24	100	”	1.6	4.6
		UB	**745.5**	23.8	99.3	”	”	”
	150	Inherit	1073.7	8.0	33.8	0	1.5	4.6
		LB	1067.9	24.0	99.9	0.05	2.3	5.9
		UB	**1075.9**	24.0	99.96	”	”	”

tive with UB in the case of (MATCHING, 100). For MATCHING, LB substantially ignore the evaluation values as zero for missing agents and does not expect the values of them. If the local gains of agents are more asymmetric, LB might be better, while we tuned settings within symmetric DCOPs in this study.



(a) Solution quality (b) #ignored agt. (IGNORE/RCVCMT) (c) #ignored agt. (REJOIN)

Figure 2: Anytime curve of an instance (RANDOM, $c = 150$, UB, 25 low-communication-quality agents with message loss rate 0.5).

Table 4 shows the results of the cases that always accepted commit messages, and Table 5 shows the cases that allowed cooperation to be rejoined. With the (partial) recovery of coordination, the progress of solution processes was increased. From the comparison of the cases of RCVMT and REJOIN, the sharing of agents' states seemed to sufficiently propagate information particularly in the simple local search method. In comparison to Table 1, the perturbation due to incomplete messaging increased the search neighborhood in a solution space and improved the solution quality, as pointed out in the previous study (Rachmut et al., 2022).

Figure 2 shows a set of anytime curves for an instance. As shown in Figure 2 (a), the solution quality monotonically increased in IGNORE, while RCVCMT and REJOIN were unstable due to the additional communication. On the other hand, the latter two methods obtained relatively higher quality of solutions in later steps, and those were competitive with BASE in average. Regarding the comparison of Figures 2 (b) and (c), REJOIN reduced the number of ignored agents in average.

5 CONCLUSION

We investigated fundamental implementation techniques toward distributed constraint optimization methods on a message-oriented communication platform based on the publish/subscribe model. We addressed the deterministic local search method on low QoS settings for a communication environment with message loss, where the solution method must continue with temporally missing agents. We experimentally evaluated the influence of several approaches to handle temporally non-responding agents in an executing solution process, and our result revealed the potentiality of such approaches. On the other hand, the proposed methods compromise the monotonicity of anytime solution that is a property of the original MGM algorithm, and some additional investigation is necessary to ensure the stability of the solution.

Our future study will investigate different solution methods with more phases, mechanisms to improve the stability of anytime solutions, application to dynamically changing problems, and evaluation in actual environments.

ACKNOWLEDGEMENTS

This study was supported in part by The Telecommunications Advancement Foundation (2022 Research Grant) and JSPS KAKENHI Grant Number 22H03647.

REFERENCES

Choudhury, M., Mahmud, S., and Khan, M. M. (2020). A Particle Swarm Based Algorithm for Functional Distributed Constraint Optimization Problems. In *Proceedings of the 34th AAAI Conference on Artificial Intelligence*, volume 34, pages 7111–7118.

Fioretto, F., Pontelli, E., and Yeoh, W. (2018). Distributed Constraint Optimization Problems and Applications: A Survey. *Journal of Artificial Intelligence Research*, 61:623–698.

Fioretto, F., Yeoh, W., and Pontelli, E. (2017). A Multiagent System Approach to Scheduling Devices in Smart Homes. In *Proceedings of the 16th Conference on Autonomous Agents and MultiAgent Systems*, page 981–989.

Kumar, A., Faltings, B., and Petcu, A. (2009). Distributed constraint optimization with structured resource constraints. In *Proceedings of the 8th International Conference on Autonomous Agents and Multiagent Systems*, pages 923–930.

Maheswaran, R. T., Pearce, J. P., and Tambe, M. (2004a). Distributed Algorithms for DCOP: A Graphical Game-Based Approach. In *Proceedings of the 17th International Conference on Parallel and Distributed Computing Systems*, pages 432–439.

Maheswaran, R. T., Tambe, M., Bowring, E., Pearce, J. P., and Varakantham, P. (2004b). Taking DCOP to the Real World: Efficient Complete Solutions for Distributed Multi-Event Scheduling. In *Proceedings of the 3rd International Joint Conference on Autonomous Agents and Multiagent Systems*, pages 310–317.

Mahmud, S., Choudhury, M., Khan, M. M., Tran-Thanh, L., and Jennings, N. R. (2020). AED: An Anytime Evolutionary DCOP Algorithm. In *Proceedings of the 19th International Conference on Autonomous Agents and MultiAgent Systems*, pages 825–833.

Matsui, T. (2020). Decentralized Constraint Optimization in Composite Observation Task Allocation to Mobile Sensor Agents. In *Proceedings of the 18th International Conference on Practical Applications of Agents and Multi-Agent Systems*, pages 171–187.

Matsui, T. (2023). Study on Decentralized Anytime Evolutionary Algorithm for DCOPs Containing Adversarial Agents. In *In Proceedings of the 15th International Conference on Agents and Artificial Intelligence*, volume 1, pages 338–346.

Matsui, T. and Matsuo, H. (2012). Considering Equality on Distributed Constraint Optimization Problem for Resource Supply Network. In *2012 IEEE/WIC/ACM International Joint Conferences on Web Intelligence and Intelligent Agent Technology*, volume 2, pages 25–32.

Matsui, T., Matsuo, H., Silaghi, M. C., Hirayama, K., Yokoo, M., and Baba, S. (2010). A Quantified Distributed Constraint Optimization Problem. In *Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems*, volume 1, pages 1023–1030.

Modi, P. J., Shen, W., Tambe, M., and Yokoo, M. (2005). Adopt: Asynchronous distributed constraint optimization with quality guarantees. *Artificial Intelligence*, 161(1-2):149–180.

Nguyen, D. T., Yeoh, W., Lau, H. C., and Zivan, R. (2019). Distributed Gibbs: A Linear-Space Sampling-Based DCOP Algorithm. *Journal of Artificial Intelligence Research*, 64(1):705–748.

Okimoto, T., Joe, Y., Iwasaki, A., Yokoo, M., and Faltings, B. (2011). Pseudo-Tree-Based Incomplete Algorithm for Distributed Constraint Optimization with Quality Bounds. In *Proceedings of the 17th International Conference on Principles and Practice of. Constraint Programming*, volume 6876 of *Lecture Notes in Computer Science*, pages 660–674.

Ottens, B., Dimitrakakis, C., and Faltings, B. (2012). DUCT: An Upper Confidence Bound Approach to Distributed Constraint Optimization Problems. In *Proceedings of the 26th AAAI Conference on Artificial Intelligence*, pages 528–534.

Pearce, J. P., Tambe, M., and Maheswaran, R. (2008). Solving Multiagent Networks Using Distributed Constraint Optimization. *AI Magazine*, 29(3):47–62.

Petcu, A. and Faltings, B. (2005). A Scalable Method for Multiagent Constraint Optimization. In *Proceedings of the 19th International Joint Conference on Artificial Intelligence*, pages 266–271.

Rachmut, B., Zivan, R., and Yeoh, W. (2022). Communication-Aware Local Search for Distributed Constraint Optimization. *Journal of Artificial Intelligence Research*, 75:637–675.

Ramchurn, S. D., Farinelli, A., Macarthur, K. S., and Jennings, N. R. (2010). Decentralized Coordination in RoboCup Rescue. *Computer Journal*, 53(9):1447–1461.

Rogers, A., Farinelli, A., Stranders, R., and Jennings, N. R. (2011). Bounded Approximate Decentralised Coordination via the Max-Sum Algorithm. *Artificial Intelligence*, 175(2):730–759.

Yeoh, W., Felner, A., and Koenig, S. (2008). BnB-ADOPT: an asynchronous branch-and-bound DCOP algorithm. In *Proceedings of the 7th International Joint Conference on Autonomous Agents and Multiagent Systems*, pages 591–598.

Yeoh, W. and Yokoo, M. (2012). Distributed Problem Solving. *AI Magazine*, 33(3):53–65.

Yokoo, M., Durfee, E. H., Ishida, T., and Kuwabara, K. (1998). The Distributed Constraint Satisfaction Problem: Formalization and Algorithms. *IEEE Transactions on Knowledge and Data Engineering*, 10(5):673–685.

Zhang, W., Wang, G., Xing, Z., and Wittenburg, L. (2005). Distributed stochastic search and distributed breakout: properties, comparison and applications to constraint optimization problems in sensor networks. *Artificial Intelligence*, 161(1-2):55–87.

Zivan, R., Glinton, R., and Sycara, K. (2009). Distributed Constraint Optimization for Large Teams of Mobile Sensing Agents. In *2009 IEEE/WIC/ACM International Joint Conference on Web Intelligence and Intelligent Agent Technology*, volume 2, pages 347–354.