Byzantine Fault Tolerance in Distributed Constraint Optimization Problems (Supplementary Material)

Koji Noshiro[1] and Koji Hasebe[1]

[1]University of Tsukuba

Abstract

This document is the supplementary material for the paper "Fault Tolerance Against Byzantine Faults in Distributed Constraint Optimization Problems".

A Proof of impossibility result

This section presents the proof of the impossibility result of FT-DCOPs.

For any g and k such that $g \leq 2k$, no (g,k)-complete FT-DCOP algorithm exists.

Proof. We prove this theorem by contradiction. We assume that there exists a (g,k)-complete FT-DCOP algorithm π such that $g \leq 2k$. Without loss of generality, we assume that π is deterministic.

Consider the following two FT-DCOPs P and P':

$$P = (A, X, D, F, \alpha, \gamma)\,,$$
$$P' = (A, X, D, F', \alpha, \gamma)\,.$$

These FT-DCOPs differ only in their sets of utility functions. The sets are defined as $F = F_0 \cup \{f_i\}$ and $F' = F_0 \cup \{f'_i\}$, where F_0 is a common set of utility functions. The utility functions f_i and f'_i differ only in their values, having the same scopes and satisfying $\gamma(f_i) = \gamma(f'_i)$. We assume that $|\gamma(f_i)| = |\gamma(f'_i)| = g$.

In the two FT-DCOPs, we consider two distinct sets of faulty agents, B and B'. Since $|\gamma(f_i)| = g \leq 2k$, there exist B and B' such that $|\gamma(f_i)| \leq 2|\gamma(f_i) \cap B|$, $|\gamma(f_i)| \leq 2|\gamma(f_i) \cap B'|$, and $B \cup B' = \gamma(f_i) = \gamma(f'_i)$. Additionally, let $\beta(f_i) = \gamma(f_i) \cap B$ and $\beta'(f'_i) = \gamma(f'_i) \cap B'$. We also define $\tilde{S} = B \cap B'$, $S = B \setminus \tilde{S}$, and $S' = B' \setminus \tilde{S}$.

We further assume that there exists a variable x_h such that $\alpha(x_h) \cap (B \cup B') = \emptyset$ and the sets of x_h's assignments in the optimal solutions of P and P', denoted by Σ_h and Σ'_h respectively, are disjoint, i.e.,

$$\begin{aligned}
&\Sigma_h \cap \Sigma'_h \\
=&\operatorname*{argmax}_{d \in D_h} \max_{\sigma \in \prod_{x_{j'} \in X} D_{j'}} \sum_{f_j \in F} f_j\left((d, \sigma)_{\mathbf{x}^j}\right) \\
\cap&\operatorname*{argmax}_{d \in D_h} \max_{\sigma \in \prod_{x_{j'} \in X} D_{j'}} \sum_{f_j \in F'} f_j\left((d, \sigma)_{\mathbf{x}^j}\right) \\
=&\emptyset,
\end{aligned}$$

where $(d, \sigma)_{\mathbf{x}^j}$ denotes the projection of the combined assignment of d and σ onto the scope of f_j.

First, when solving P with π, we assume the faulty agents $B = S \cup \tilde{S}$ behave as follows. Agents in S execute π using utility function f'_i instead of f_i. Except for this replacement of the utility function, the agents correctly follow the algorithm π. On the other hand, agents in $\tilde{S}$ do not execute any operation. Let e be an execution of the entire system, i.e., the sequence of message transmissions. By our assumption regarding π, the correct agents obtain an optimal solution to P upon termination. Specifically, the correct agents assigned x_h obtain an assignment $d_h \in \Sigma_h$.

Next, when solving P' with π, we similarly assume that agents in S' execute π using utility function f_i instead of f'_i, while agents in $\tilde{S}$ do not execute any operation. Let e' be this system execution. Since S, S', and $\tilde{S}$ partition $\gamma(f_i) = \gamma(f'_i)$, all agents in

$\gamma(f_i) = \gamma(f_i')$, including all faulty agents, behave identically in both e and e': agents in S execute π with f_i', agents in S' execute π with f_i, and agents in $\tilde{S}$ do not execute any operation. Consequently, all other agents without knowledge of f_i and f_i' also execute π identically in both e and e', as the problems P and P' differ only in the utility functions and sets of faulty agents.

Therefore, the executions e and e' are identical when agents exchange messages in the same order. In this case, the correct agent assigned x_h obtains the assignment $d_h \in \Sigma_h$ in execution e' as well. However, since d_h is not in the optimal assignment set of P', i.e., $d_h \notin \Sigma_h'$, this contradicts our assumption about π. □

B Additional Results in Experimental Evaluation

This section presents additional results that complement Section 6. First, we compare the solution quality of Max-Sum and Repl-Max-Sum on identical graph coloring problem (GCP) instances. Then, we report a preliminary evaluation of the communication overhead of Repl-Max-Sum on a small testbed.

B.1 Comparison of Solution Quality on Graph Coloring Problems

To provide a more precise comparison between algorithms, we analyze their outcomes on identical instances in GCPs. For each experimental condition, both algorithms solved the same 50 instances. We quantify the difference in solution quality (measured by the number of constraint violations) using the following metric:

$$\Delta = \mathrm{CV}_{\mathrm{MS}} - \mathrm{CV}_{\mathrm{Repl}},$$

where $\mathrm{CV}_{\mathrm{MS}}$ and $\mathrm{CV}_{\mathrm{Repl}}$ denote the number of constraint violations obtained by Max-Sum and Repl-Max-Sum, respectively. A positive Δ indicates that Max-Sum produced a solution with more constraint violations.

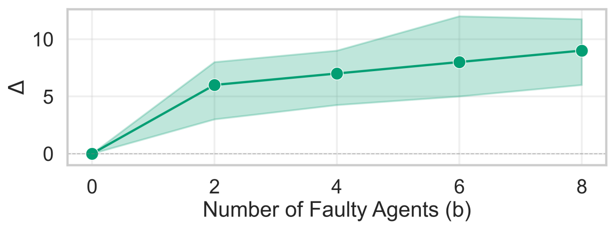

Figure 1: Difference of Constraint Violations on GCPs ($n = 24$)

1 Medians and IQRs of differences of constraint violations between Max-Sum and Repl-Max-Sum on GCPs when $n = 24$.

Table 1: Distribution of Δ ($n = 24$)

	$b = 0$	$b = 2$	$b = 4$	$b = 6$	$b = 8$
$\Delta < 0$	0	3	1	0	3
$\Delta = 0$	50	5	3	1	1
$\Delta > 0$	0	42	46	49	46

We first fixed $n = 24$ and varied the number of faulty agents $b \in \{0, 2, 4, 6, 8\}$. We evaluated the distribution of Δ across the 50 instances in terms of medians and interquartile ranges (IQRs) as shown in Figure 1. As described in Section 6, when no faults are present ($b = 0$), Repl-Max-Sum returns identical solutions to Max-Sum, and thus Δ is 0 for all instances. In contrast, when faults are present ($b > 0$), the IQR of Δ never falls below 3 and the median is at least 6. In other words, compared to Repl-Max-Sum, Max-Sum incurs at least 3 additional violations on at least 75% of instances and at least 6 additional violations on at least 50% of instances. Moreover, both the median and IQR of Δ increase approximately monotonically with b, indicating that the degradation of Max-Sum intensifies as the number of faulty agents grows, while Repl-Max-Sum continues to mask their impact.

Additionally, Table 1 presents the distribution of Δ when $n = 24$ and $b \in \{0, 2, 4, 6, 8\}$. As shown in the table, Δ is positive, i.e., Repl-Max-Sum achieved better solutions than Max-Sum, in over 80% instances with $b = 2$ and in over 90%instances with $b \in \{2, 4, 6, 8\}$. These results highlight the superior-

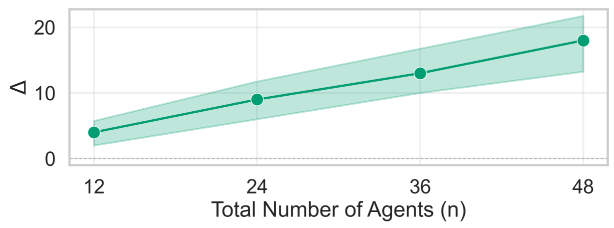

Figure 2: Difference of Constraint Violations on GCPs ($b = n/3$)

1 Medians and IQRs of differences of constraint violations between Max-Sum and Repl-Max-Sum on GCPs when $b = n/3$.

Table 2: Distribution of Δ ($b = n/3$)

	$n = 12$	$n = 24$	$n = 36$	$n = 48$
$\Delta < 0$	5	3	0	0
$\Delta = 0$	2	1	1	0
$\Delta > 0$	43	46	49	50

ity of Repl-Max-Sum.

We also observed rare exceptions: for each $b > 0$, at most three instances exhibited fewer violations with Max-Sum than with Repl-Max-Sum. This phenomenon can be attributed to the incompleteness of (Repl-)Max-Sum; random messages generated by faulty agents can occasionally and inadvertently steer the updates of the q and r messages in a favorable direction for Max-Sum. However, in our truck appointment scheduling (TAS) experiments, Repl-Max-Sum consistently outperformed Max-Sum on all instances when faults were present.

Next, we fixed the fault scale at $b = n/3$ and varied the number of agents $n \in \{12, 24, 36, 48\}$, showing the results in Figure 2 and Table 2. As presented in the figure, for all n, the IQR of Δ remains strictly positive, demonstrating that Repl-Max-Sum yields better solutions than Max-Sum on the majority of instances. Furthermore, Δ increases monotonically with n. In particular, for $n = 48$, the median of Δ reaches 18, underscoring the widening performance gap and the pronounced advantage of Repl-Max-Sum as problem size grows.

B.2 Preliminary Experiments Regarding Communication Overhead

Repl-Max-Sum incurs a higher communication cost than Max-Sum. Specifically, in the experimental configuration, Repl-Max-Sum assigns three replicas to each node, which multicast across replica sets of neighboring nodes. Consequently, the number of messages is theoretically amplified by approximately $3 \cdot 3 = 9$ compared to Max-Sum, a trend consistent with the results in Section 6. However, multiple replicas assigned to an agent can be processed in parallel, and messages are small (in our GCP implementation, each message is below 256 bytes). Hence, the wall-clock runtime need not scale proportionally to the increase in message count.

To examine this hypothesis, we conducted a small test on two physical servers (denoted S1 and S2) connected over the Internet. We repeated the following round-trip for 1000 rounds:

1. S1 sends N messages;
2. after receiving N messages, S2 replies with N messages;
3. after receiving N replies, S1 proceeds to the next round.

This procedure emulates the parallel per-replica processing and synchronous message passing used in (Repl-)Max-Sum. The experiment was implemented in Python with PyZMQ (Python bindings for ZeroMQ, a well-known messaging library). To mirror the difference in message multiplicity between Max-Sum and Repl-Max-Sum, we compared two conditions, $N = 1$ and $N = 9$.

Over five trials per condition, the average completion time for 1000 rounds was 13.0 seconds (standard deviation 0.264) for $N = 1$ and 23.2 seconds (standard deviation 0.635) for $N = 9$. Thus, even with nine times as many messages, runtime increased by less than a factor of two. These results suggest that although Repl-Max-Sum increases the number of messages quadratically with the number of replicas per node, practical runtime overhead can be substantially mitigated by parallelism and small message size.

Nevertheless, the actual overhead in deployment depends on multiple factors, including the cost of local optimization processes at nodes and network conditions. A rigorous evaluation should therefore be conducted on the target system itself, which we leave for future work.