Appendix for Neural Regret-Matching for Distributed Constraint Optimization Problems

A Technical Proofs

In this section, we present the proofs of the proposed theorems.

A.1 Proof of Theorem 1

Proof. Adapted from [Tammelin *et al.*, 2015]. Since $\bar{R}_i^T(X_i, d_i) > 0$,

$$\left(\max_{d_i} \bar{R}_i^T(X_i, d_i)\right)^2 = \max_{d_i} \bar{R}_i^T(X_i, d_i)^2 \leq \sum_{d_i} \bar{R}_i^T(X_i, d_i)^2$$

If $X_i = X_i^T$, then

$$\begin{aligned}
\sum_{d_i} \bar{R}_i^T(X_i, d_i)^2 &= \sum_{d_i} \max\left(\bar{R}_i^{T-1}(X_i, d_i) + s_i^T - S_i^T(d_i), \delta_T\right)^2 \\
&\leq \sum_{d_i} \left(\bar{R}_i^{T-1}(X_i, d_i) + s_i^T - S_i^T(d_i)\right)^2 + \delta_T^2 \\
&= |D_i|\delta_T^2 + \sum_{d_i} \bar{R}_i^{T-1}(X_i, d_i)^2 + 2\bar{R}_i^{T-1}(X_i, d_i)\left(\sum_{d_i'} \pi_i^T(d_i') S_i^T(d_i') - S_i^T(d_i)\right) \\
&\quad + \left(\sum_{d_i'} \pi_i^T(d_i') S_i^T(d_i') - S_i^T(d_i)\right)^2
\end{aligned}$$

Since $L_i \geq |S_i^t(d_i) - S_i^t(d_i')|$ for all $d_i \neq d_i'$, we have

$$\begin{aligned}
\sum_{d_i} \bar{R}_i^T(X_i, d_i)^2 &\leq |D_i|\delta_T^2 + |D_i|L_i^2 + \sum_{d_i} \bar{R}_i^{T-1}(X_i, d_i)^2 + 2\bar{R}_i^{T-1}(X_i, d_i) \left(\sum_{d_i'} \frac{\bar{R}_i^{T-1}(X_i, d_i')}{\sum_{d_i''} \bar{R}_i^{T-1}(X_i, d_i'')} S_i^T(d_i') - S_i^T(d_i) \right) \\
&= |D_i|\delta_T^2 + |D_i|L_i^2 + \sum_{d_i} \bar{R}_i^{T-1}(X_i, d_i)^2 + 2 \left(\sum_{d_i} \bar{R}_i^{T-1}(X_i, d_i) \sum_{d_i'} \frac{\bar{R}_i^{T-1}(X_i, d_i')}{\sum_{d_i''} \bar{R}_i^{T-1}(X_i, d_i'')} S_i^T(d_i') \right. \\
&\quad \left. - \sum_{d_i} \bar{R}_i^{T-1}(X_i, d_i) S_i^T(d_i) \right) \\
&= |D_i|\delta_T^2 + |D_i|L_i^2 + \sum_{d_i} \bar{R}_i^{T-1}(X_i, d_i)^2 + 2 \left(\frac{\sum_{d_i} \bar{R}_i^{T-1}(X_i, d_i)}{\sum_{d_i''} \bar{R}_i^{T-1}(X_i, d_i'')} \sum_{d_i'} \bar{R}_i^{T-1}(X_i, d_i') S_i^T(d_i') \right. \\
&\quad \left. - \sum_{d_i} \bar{R}_i^{T-1}(X_i, d_i) S_i^T(d_i) \right) \\
&= |D_i|\delta_T^2 + |D_i|L_i^2 + \sum_{d_i} \bar{R}_i^{T-1}(X_i, d_i)^2 + 2 \left(\sum_{d_i'} \bar{R}_i^{T-1}(X_i, d_i') S_i^T(d_i') - \sum_{d_i} \bar{R}_i^{T-1}(X_i, d_i) S_i^T(d_i) \right) \\
&= |D_i|\delta_T^2 + |D_i|L_i^2 + \sum_{d_i} \bar{R}_i^{T-1}(X_i, d_i)^2
\end{aligned}$$

If $X_i \neq X_i^T$, according to Eq.(6), we have

$$\sum_{d_i} \bar{R}_i^T(X_i, d_i)^2 = \sum_{d_i} \bar{R}_i^{T-1}(X_i, d_i)^2$$

Therefore,

$$\left(\max_{d_i} \bar{R}_i^T(X_i, d_i) \right)^2 \leq |D_i|\delta_T^2 + |D_i|L_i^2 + \sum_{d_i} \bar{R}_i^{T-1}(X_i, d_i)^2$$

By iterating over the above formula, we get $\left(\max_{d_i} \bar{R}_i^T(X_i, d_i)\right)^2 \leq |D_i| \left(TL_i^2 + \sum_{t=1}^T \delta_t^2 \right)$ and therefore $\max_{d_i} \bar{R}_i^T(X_i, d_i) \leq \sqrt{|D_i| \left(TL_i^2 + \sum_{t=1}^T \delta_t^2 \right)}, \forall X_i$. □

A.2 Proof of Theorem 2

We first show that the total regret is bounded by the maximal regret of an individual variable. Then we show the total regret grows sublinearly.

Let $d^T = (d_1^T, \ldots, d_n^T)$ be the joint assignment of round T, $d_{-i}^T = d^T \backslash \{d_i^T\}$ be the set of assignments selected by variables other than x_i and $F(d^T) = \sum_{f_{ij} \in F} f_{ij}(d_i^T, d_j^T)$ be the sum of all constraints given joint assignment d^T.

Definition 1. *The 1-opt regret of a DCOP is*

$$R_{1-opt}^T = \max_{i \in I} \max_{d_i \in D_i} \sum_{t=1}^T \left(\sum_{d_i' \in D_i} F(d_{-i}^t, d_i') \pi_i^t(d_i') - F(d_{-i}^t, d_i) \right) \tag{1}$$

Lemma 1. *$R_{1-opt}^T \leq K \max_i R_i^T$, where $R_i^T = \max_{X_i} \max_{d_i} \bar{R}_i^T(X_i, d_i)$ is the maximal local regret of variable x_i at round T and K is a constant.*

Proof. Given joint assignment d, denote the sum of the constraints which are related to x_i as $F_i(d_i, d_{N_i}) = \sum_{x_j \in N_i} f_{ij}(d_i, d_j)$ where $d_{N_i} \subseteq d$ is the set of assignments to x_i's neighbors, and the sum of those which are not related to x_i as $F_{-i}(d_{-i}) =$

$F(d) - F_i(d_i, d_{N_i})$. Note that

$$\begin{aligned}
\max_{d_i} \sum_{t=1}^{T} \left(\sum_{d_i'} F(d_{-i}^t, d_i') \pi_i^t(d_i') - F(d_{-i}^t, d_i) \right) &= \max_{d_i} \sum_{t=1}^{T} \left(\sum_{d_i'} \left(F_{-i}(d_{-i}^t) + F_i(d_i', d_{N_i}^t) \right) \pi_i^t(d_i') \right. \\
&\quad \left. - \left(F_{-i}(d_{-i}^t) + F_i(d_i, d_{N_i}^t) \right) \right) \\
&= \max_{d_i} \sum_{t=1}^{T} \left(\sum_{d_i'} F_{-i}(d_{-i}^t) \pi_i^t(d_i') + \sum_{d_i'} F_i(d_i', d_{N_i}^t) \pi_i^t(d_i') \right. \\
&\quad \left. - \left(F_{-i}(d_{-i}^t) + F_i(d_i, d_{N_i}^t) \right) \right) \\
&= \max_{d_i} \sum_{t=1}^{T} \left(\sum_{d_i'} F_i(d_i', d_{N_i}^t) \pi_i^t(d_i') - F_i(d_i, d_{N_i}^t) \right) \\
&= \max_{d_i} \sum_{t=1}^{T} \left(\sum_{d_i'} \left(\sum_{x_j \in AP(x_i)} f_{ij}(d_i', d_j^t) + \sum_{x_k \in AC(x_i)} f_{ij}(d_i', d_k^t) \right) \pi_i^t(d_i') \right. \\
&\quad \left. - \left(\sum_{x_j \in AP(x_i)} f_{ij}(d_i, d_j^t) + \sum_{x_k \in AC(x_i)} f_{ij}(d_i, d_k^t) \right) \right) \\
&= \max_{d_i} \sum_{t=1}^{T} s_i^t - S_i^t(d_i)
\end{aligned}$$

Assume that x_i has received $k_i^T \leq K_i$ different contexts $X_{i,1}, \ldots, X_{i,k_i^T}$ in first T rounds, where K_i is the total number of possible contexts of x_i.

$$\begin{aligned}
\max_{d_i} \sum_{t=1}^{T} s_i^t - S_i^t(d_i) &= \max_{d_i} \sum_{j=1}^{k_i^T} R_i^T(X_{i,j}, d_i) \\
&\leq \sum_{j=1}^{k_i^T} \max_{d_i} \max_{X_i} R_i^T(X_i, d_i) = k_i^T R_i^T \leq K_i R_i^T
\end{aligned}$$

Therefore,

$$R_{1-opt}^T \leq \max_i K_i R_i^T = K \max_i R_i^T$$

where $K = \max_i K_i$. □

Lemma 2. *Rounded regret are the upper bound on regret, i.e.,* $\bar{R}_i^t(X_i, d_i) \geq R_i^t(X_i, d_i), \forall t, X_i, d_i$.

Proof. For any $t > 0$, if $X_i = X_i^t$, then

$$\begin{aligned}
\bar{R}_i^t(X_i, d_i) - \bar{R}_i^{t-1}(X_i, d_i) &= \max\left(\bar{R}_i^{t-1}(X_i, d_i) + s_i^t - S_i^t(d_i), \delta_t \right) - \bar{R}_i^{t-1}(X_i, d_i) \\
&\geq \bar{R}_i^{t-1}(X_i, d_i) + s_i^t - S_i^t(d_i) - \bar{R}_i^{t-1}(X_i, d_i) \\
&= s_i^t - S_i^t(d_i) = R_i^t(X_i, d_i) - R_i^{t-1}(X_i, d_i)
\end{aligned}$$

Otherwise,

$$\bar{R}_i^t(X_i, d_i) - \bar{R}_i^{t-1}(X_i, d_i) = R_i^t(X_i, d_i) - R_i^{t-1}(X_i, d_i) = 0$$

Note that $\bar{R}_i^0(X_i, d_i) = R_i^0(X_i, d_i) = 0$. By induction, $\bar{R}_i^t(X_i, d_i) \geq R_i^t(X_i, d_i)$ holds for any t, X_i and d_i, which concludes the lemma. □

We now are ready to prove Theorem 2.

Proof of Theorem 2. According to Lemma 2, we have $\bar{R}_i^T(X_i, d_i) \geq R_i^T(X_i, d_i), \forall T, X_i, d_i$. Therefore, to show the the

algorithm is no-regret, we only needs to show the maximal rounded regret $\bar{R}_i^T = \max_{X_i} \max_{d_i} \bar{R}_i^T(X_i, d_i)$ grows sublinearly.

$$\begin{aligned}
\lim_{T\to\infty} \frac{\bar{R}_i^T}{T} &\leq \lim_{T\to\infty} \sqrt{\frac{|D_i|\left(TL_i^2 + \sum_{t=1}^T \delta_t^2\right)}{T^2}} \\
&= \lim_{T\to\infty} \sqrt{\frac{|D_i|L_i^2}{T^2} + \frac{|D_i|\sum_{t=1}^t \delta_t^2}{T^2}} \\
&\leq \lim_{T\to\infty} \sqrt{\frac{|D_i|L_i^2}{T^2} + \frac{|D_i|\delta_T^2}{T}} \\
&= \lim_{T\to\infty} \sqrt{\frac{|D_i|L_i^2}{T^2} + \frac{|D_i|(\sqrt{T})^2}{T}\left(\frac{\delta_T}{\sqrt{T}}\right)^2}
\end{aligned}$$

Since $\lim_{T\to\infty} \delta_T/\sqrt{T} = 0$, we have

$$\begin{aligned}
\lim_{T\to\infty} \frac{\bar{R}_i^T}{T} &\leq \lim_{T\to\infty} \sqrt{\frac{|D_i|L_i^2}{T^2} + |D_i|\left(\frac{\delta_T}{\sqrt{T}}\right)^2} \\
&= 0
\end{aligned}$$

Therefore, R_i^T grows sublinearly and according to Lemma 1 R_{1-opt}^T also grows sublinearly, which concludes the theorem. □

A.3 Proof of Theorem 3

We begin with showing the gap between the rounded regret and the Q-regret in RM$^+$ [Tammelin *et al.*, 2015] is no greater than δ_t. Then we show the regret bound by proving the upper bound of Q-regret.

Lemma 3. *For variable x_i, $\bar{R}_i^t(X_i, d_i) - Q_i^t(X_i, d_i) \leq \delta_t$ for all X_i, d_i, t, where $Q_i^t(X_i, d_i) = (Q_i^{t-1}(X_i, d_i) + \mathbb{I}(X_i = X_i^t)(s_i^t - S_i^t(d_i))^+$ and $Q_i^0(X_i, d_i) = 0$.*

Proof. The lemma is trivial when $t = 1$ and $X_i \neq X_i^1$, as $\bar{R}_i^1(X_i, d_i) = Q_i^1(X_i, d_i) = 0$. Consider the case that $t = 1$ and $X_i = X_i^1$

$$\bar{R}_i^1(X_i, d_i) - Q_i^1(X_i, d_i) = \max(\delta_1, s_i^1 - S_i^1(d_i)) - \max(0, s_i^1 - S_i^1(d_i)) \leq \delta_1$$

Thus, the lemma holds for the basis. Assume the lemma holds for the $t-1$-th round, we are going to show the lemma holds for the t-th round as well.

If $X_i \neq X_i^t$, then $\bar{R}_i^t(X_i, d_i) - Q_i^t(X_i, d_i) = \bar{R}_i^{t-1}(X_i, d_i) - Q_i^{t-1}(X_i, d_i) \leq \delta_{t-1} \leq \delta_t$. Otherwise,

$$\bar{R}_i^t(X_i, d_i) - Q_i^t(X_i, d_i) = \max(\delta_t, \bar{R}_i^{t-1}(X_i, d_i) + s_i^t - S_i^t(d_i)) - \max(0, Q_i^{t-1}(X_i, d_i) + s_i^t - S_i^t(d_i))$$

If $s_i^t - S_i^t(d_i) \leq -Q_i^{t-1}(X_i, d_i)$, then

$$\begin{aligned}
\bar{R}_i^t(X_i, d_i) - Q_i^t(X_i, d_i) &= \max(\delta_t, \bar{R}_i^{t-1}(X_i, d_i) + s_i^t - S_i^t(d_i)) - 0 \\
&\leq \max(\delta_t, \delta_{t-1} + Q_i^{t-1}(X_i, d_i) + s_i^t - S_i^t(d_i)) \\
&\leq \max(\delta_t, \delta_{t-1}) = \delta_t
\end{aligned}$$

If $s_i^t - S_i^t(d_i) > -Q_i^{t-1}(X_i, d_i)$ and $\delta_t \leq \bar{R}_i^{t-1}(X_i, d_i) + s_i^t - S_i^t(d_i)$, then

$$\begin{aligned}
\bar{R}_i^t(X_i, d_i) - Q_i^t(X_i, d_i) &= \bar{R}_i^{t-1}(X_i, d_i) + s_i^t - S_i^t(d_i) - \left(Q_i^{t-1}(X_i, d_i) + s_i^t - S_i^t(d_i)\right) \\
&= \bar{R}_i^{t-1}(X_i, d_i) - Q_i^{t-1}(X_i, d_i) \leq \delta_{t-1} \leq \delta_t
\end{aligned}$$

If $s_i^t - S_i^t(d_i) > -Q_i^{t-1}(X_i, d_i)$ and $\delta_t > \bar{R}_i^{t-1}(X_i, d_i) + s_i^t - S_i^t(d_i)$, then

$$\bar{R}_i^t(X_i, d_i) - Q_i^t(X_i, d_i) = \delta_t - \left(Q_i^{t-1}(X_i, d_i) + s_i^t - S_i^t(d_i)\right)$$

Since $Q_i^{t-1}(X_i, d_i) + s_i^t - S_i^t(d_i) > 0$, $\bar{R}_i^t(X_i, d_i) - Q_i^t(X_i, d_i) < \delta_t$. Therefore, the lemma holds for all X_i, d_i, t. □

We now are ready to prove Theorem 3.

Proof of Theorem 3. According to Morrill's [Morrill, 2016] Theorem 3.0.10 and Theorem 3.0.12, the bound of Q-regrets if we use $\hat{R}_i^t(X_i, \cdot)$ to generate strategy in each round t is

$$\sqrt{L_i\sqrt{|D_i|}\left(L_iT\sqrt{|D_i|} + 4\sum_{t=1}^{T}\sum_{d_i'}|Q_i^t(X_i, d_i') - \hat{R}_i^t(X_i, d_i')|\right)}$$

Note that

$$\begin{aligned}
\sum_{t=1}^{T}\sum_{d_i'}|Q_i^t(X_i, d_i') - \hat{R}_i^t(X_i, d_i')| &= \sum_{t=1}^{T}\sum_{d_i'}|\hat{R}_i^t(X_i, d_i') - \bar{R}_i^t(X_i, d_i') + \bar{R}_i^t(X_i, d_i') - Q_i^t(X_i, d_i')| \\
&\leq \sum_{t=1}^{T}\sum_{d_i'}|\hat{R}_i^t(X_i, d_i') - \bar{R}_i^t(X_i, d_i')| + |\bar{R}_i^t(X_i, d_i') - Q_i^t(X_i, d_i')| \\
&\leq \sum_{t=1}^{T}\sum_{d_i'}|\hat{R}_i^t(X_i, d_i') - \bar{R}_i^t(X_i, d_i')| + \delta_t \\
&= \sum_{t=1}^{T}|D_i|\delta_t + \sum_{d_i'}|\hat{R}_i^t(X_i, d_i') - \bar{R}_i^t(X_i, d_i')| \\
&\leq T|D_i|\delta_T + \sum_{t=1}^{T}\sum_{d_i'}|\hat{R}_i^t(X_i, d_i') - \bar{R}_i^t(X_i, d_i')|
\end{aligned}$$

Since $\sum_{d_i'}|\hat{R}_i^t(X_i, d_i') - \bar{R}_i^t(X_i, d_i')| \leq \sqrt{|D_i|}\sum_{d_i'}\sqrt{(\hat{R}_i^t(X_i, d_i') - \bar{R}_i^t(X_i, d_i'))^2}$ and $R_i^t(X_i, d_i) < Q_i^t(X_i, d_i)$ (Lemma 1 of [Tammelin *et al.*, 2015]), the regret bound of neural-based sampling scheme is

$$\sqrt{L_i|D_i|\left((4\sqrt{|D_i|}\delta_T + L_i)T + 4\sum_{t=1}^{T}\sum_{d_i'}(\hat{R}_i^t(X_i, d_i') - \bar{R}_i^t(X_i, d_i'))^2\right)}$$

□

B Related Work

Algorithms for DCOPs. According to their ability to guarantee optimality, DCOP algorithms are classified as either complete or incomplete. While search-based complete algorithms [Gershman *et al.*, 2009; Hirayama and Yokoo, 1997; Litov and Meisels, 2017; Modi *et al.*, 2005; Netzer *et al.*, 2012; Yeoh *et al.*, 2010] explicitly exhaust the solution space by distributed backtrack search, inference-based complete algorithms [Petcu and Faltings, 2005; Petcu and Faltings, 2007; Vinyals *et al.*, 2011] perform dynamic programming to backup utility tables. Some hybrid schemes [Atlas *et al.*, 2008; Chen *et al.*, 2019; Deng *et al.*, 2019; Kim and Lesser, 2014] attempt to combine the advantages of both search and inference. However, due to the NP-hardness of solving a DCOP, complete algorithms incur exponential overheads and can only solve the problems with handful variables. In contrast, incomplete algorithms trade the solution quality for smaller computational overhead. Local search algorithms [Grinshpoun *et al.*, 2019; Hoang *et al.*, 2018; Maheswaran *et al.*, 2004; Okamoto *et al.*, 2016; Zhang *et al.*, 2005; Zivan *et al.*, 2014] iteratively optimize the solution via local moves, but they suffer from poor local convergence. Belief propagation [Chen *et al.*, 2018; Cohen *et al.*, 2020; Farinelli *et al.*, 2008; Rogers *et al.*, 2011; Zivan *et al.*, 2017] is another class of incomplete methods, where agents accumulate the global utility for each assignment according to the belief of their neighbors. Finally, the state-of-the-art sampling-based techniques including DUCT [Ottens *et al.*, 2017] and D-Gibbs [Nguyen *et al.*, 2019] perform sequential sampling on a pseudo tree, but cannot scale up due to poor sample efficiency and high memory consumption. In contrast, our proposed neural-based sampling scheme can overcome the above limitations. The regret in our method is accumulated according to the local problem of a variable, which avoids the poor sample efficiency incurred by exploring the whole subproblem. Moreover, we compactly represent the regret tables by deep neural networks, which significantly improves the scalability of the algorithm.

Regret-based methods for DCOPs. In [Chapman *et al.*, 2011], Chapman et al. established the connection between DCOPs and potential games and presented a local search adaptation of regret-matching for DCOPs. In the algorithm, each agent accumulates regret based on the assignment of its neighbors and changes its assignment according to the regret values with a fixed probability p in each round. WRM-I [Arslan *et al.*, 2007; Chapman *et al.*, 2011] is another regret-based local search scheme that converges to a pure-strategy Nash equilibrium. Different from the previous context-free methods, our methods perform sampling on a pseudo tree and store regret values for each context separately, which allows an agent to devise different strategies for different contexts. Additionally, we show the negative regret values in vanilla regret-matching could result in poor performance and propose a novel regret rounding scheme.

Regret values approximation. Regression Regret-Matching (RRM) [Morrill, 2016; Waugh *et al.*, 2015] uses regression trees to estimate the accumulated regret. However, since a regression tree cannot be trained incrementally, RRM requires to re-train the model on *all* data as long as regret values are updated, which makes it very inefficient and scales up poorly. Deep CFR [Brown *et al.*, 2019] estimate the regret by training a deep neural network on instantaneous regret values, which is not applicable to our case where instantaneous regret is not proportional to the total regret due to the non-linear regret rounding scheme.

C Pesudo Codes

Algorithm 1 presents the sketch of our proposed neural-based sampling scheme.

Algorithm 1: Neural-based sampling scheme for variable x_i

Require: number of training steps h, rounding term δ_t and discounted factor γ

When `Initialization`**:**

1 call Algorithm 2 to determine the group G_i where x_i belongs to

2 **if** x_i *is not a leaf* **then**

3 initialize neural network V_{i,θ_i} such that $V_{i,\theta_i}(X_i, d_i) = 0, \forall X_i, d_i$

4 $M_i \leftarrow \emptyset, e_i^0 \leftarrow 0, \bar{e}_i^0 \leftarrow 0, \tilde{e}_i^0 \leftarrow 0$

5 **if** x_i *is the root* **then** $t \leftarrow 1, X_i^t \leftarrow \emptyset$, `Sample()`

When *received CONTEXT*$(\{X^t, G, \bar{e}\})$ *from* $P(x_i)$**:**

6 **if** $G_i = G$ **then** $\bar{e}_i^{t-1} \leftarrow \bar{e}$

7 $X_i^t \leftarrow X^t[Sep(x_i)]$, `Sample()`

When *received BACKTRACK*$(\{x_k = d_k^t\})$ *from* $x_k \in AC(x_i)$**:**

8 $Y_i^t \leftarrow Y_i^t \cup \{x_k = d_k^t\}$

9 **if** x_i *has received all BACKTRACK from* $AC(x_i)$ **then**

10 compute hindsight local costs S_i^t and the expected local cost s_i^t

11 **foreach** $d_i \in D_i$ **do**

12 **if** $\langle X_i^t, d_i \rangle \notin M_i$ **then** $M_i(X_i^t, d_i) \leftarrow V_{i,\theta_i}(X_i^t, d_i)$

13 $M_i(X_i^t, d_i) \leftarrow \max\{\delta_t, M_i(X_i^t, d_i) + s_i^t - S_i^t(d_i)\}$

14 $e_i^t \leftarrow \gamma e_i^{t-1} + (1-\gamma) \sum_{d_i \in D_i} |M_i(X_i^t, d_i) - V_{i,\theta_i}(X_i^t, d_i)|$

15 **if** x_i *is not the root* **then** send BACKTRACK$(\{x_i = \arg\min_{d_i} S_i^t(d_i)\})$ to $\forall x_j \in AP(x_i)$

When *received ACC_ERROR*$(\tilde{e})$ *from* $x_k \in C(x_i)$**:**

16 **if** $|C(x_i)| > 1$ **then** $\tilde{e}_i^t \leftarrow e_i^t$, block until all ACC_ERROR messages from $C(x_i)$ arrive

17 **else** $\tilde{e}_i^t \leftarrow \tilde{e} + e_i^t$

18 **if** *random()* $< e_i^{t-1} / \bar{e}_i^{t-1}$ **then**

19 **for** *training step* $k = 1, \ldots, h$ **do**

20 uniformly sample a batch B from M_i and train V_{i,θ_i} to minimize $\mathcal{L}_i(\theta_i) = \frac{1}{|B|} \sum_{\langle X_i, d_i \rangle \in B} \left(M_i(X_i, d_i) - V_{i,\theta_i}(X_i, d_i)\right)^2$

21 **if** x_i *is the first variable of* G_i **then**

22 $\bar{e}_i^t \leftarrow \tilde{e}_i^t$

23 **if** x_i *is not the root* **then** send ACC_ERROR(0) to $P(x_i)$

24 **else** $t \xleftarrow{+} 1$, `Sample()`

25 **else** send ACC_ERROR$(\tilde{e}_i^t)$ to $P(x_i)$

Function `Sample()`**:**

26 **if** x_i *is not a leaf* **then**

27 $\tilde{e}_i \leftarrow 0, Y_i^t \leftarrow \emptyset, \hat{R}_i^{t-1} \leftarrow []$

28 **foreach** $d_i \in D_i$ **do**

29 **if** $\langle X_i^t, d_i \rangle \in M_i$ **then** $\hat{R}_i^{t-1}(d_i) \leftarrow M_i(X_i^t, d_i)$

30 **else** $\hat{R}_i^{t-1}(d_i) \leftarrow V_{i,\theta_i}(X_i^t, d_i)$

31 compute the mixed strategy π_i^t according to $\hat{R}_i^{t-1}$ and perform sampling $d_i^t \sim \pi_i^t$

32 send CONTEXT$(X_i^t \cup \{x_i = d_i^t, G_i, \bar{e}_i^{t-1}\})$ to $\forall x_k \in C(x_i)$

33 **else**

34 send BACKTRACK$(\{x_i = \arg\min_{d_i} \sum_{x_j \in AP(x_i)} f_{ij}(d_i, X_i^t(x_j))\})$ to $\forall x_j \in AP(x_i)$

35 send ACC_ERROR(0) to $P(x_i)$

The algorithm begins with all non-leaf agents creating both neural networks and memory (line 2-4) and the root agent start the message-driven process by sampling and propagating context (line 5). When perform sampling, a non-leaf agent retrieves the estimated regret values from either the memory or bootstrapping from the neural network, depending on whether the current context-assignment pair presents in the memory (line 27-30). Then it samples an assignment according to the derived mixed

strategy and propagates the augmented context to its children (line 30-31). Leaf agents, on the other hand, do not have any subproblem and just communicate the best response via BACKTRACK messages (line 33-34).

When received all BACKTRACK messages from its children and pseudo children, an agent computes the hindsight local costs and updates the regret values (line 9-13). A bootstrap happens if the regret value corresponding to an assignment under current context does not appear in the memory (line 12). Finally, the agent communicates its best response to its all parents if it is not the root (line 15).

To enforce prioritized training scheme in a fully decentralized manner, each agent i calls a grouping procedure (detailed in Algorithm 2) to determine the group G_i it belongs to, and maintains current discounted error e_i^t of agent i, partial error sum $\tilde{e}_i^t$ and error sum $\bar{e}_i^t$ of group G_i (line 1, 4). Here, $\tilde{e}_i^t$ is the sum of discounted error of each agent ordered behind i in the group. Therefore, if i is the first agent in the group, then $\bar{e}_i^t = \tilde{e}_i^t$. The error sum of a group is communicated via CONTEXT messages from the first agent in the group to the last agent in the group (line 32, 6), which is accumulated progressively via ACC_ERROR messages. In more detail, a ACC_ERROR message carries the partial error sum of a group. When receiving a ACC_ERROR message from a child, agent i computes $\tilde{e}_i$ by adding the partial error sum from child to its own discounted error (line 17). Then the agent continues to propagate the partial error sum if it is not the first agent of the group (line 25). Otherwise, it assigns the error sum of the group and perform the next round of sampling if it is the root (line 22-24). It is worth noting that since our prioritized training scheme is confined to chains, an agent with more than one child must be the last one in a group. Therefore, it does not need to consider the partial error sum from its children if it receives multiple ACC_ERROR messages (line 16). Finally, an agent trains its neural network with a probability proportional to its prediction error (line 18-20, 14).

Algorithm 2 presents the sketch of the grouping procedure.

Algorithm 2: Grouping procedure for variable x_i

Require: difference budget b
When `Initialization`**:**
1 **if** x_i *is the root* **then**
2 $G_i \leftarrow$ create a new group ID, mark x_i as the first variable of G_i
3 `SendGrpInfo` $(\{G_i, \{x_i\}, 1\})$
When *received GROUP*$(\{G, DIM, n^*_{dim}\})$ *from* $P(x_i)$**:**
4 **if** $G = NIL$ **then**
5 $G_i \leftarrow$ create a new group ID
6 mark x_i as the first variable of G_i
7 `SendGrpInfo` $(\{G_i, Sep(x_i) \cup \{x_i\}, |Sep(x_i)| + 1\})$
8 **else**
9 $DIM' \leftarrow DIM \cup \{x_i\} \cup Sep(x_i), {n^*_{dim}}' \leftarrow \min\{n^*_{dim}, |Sep(x_i)| + 1\}$
10 **if** $|DIM'| - {n^*_{dim}}' > b$ **then**
11 $G_i \leftarrow$ create a new group ID, mark x_i as the first variable of G_i
12 `SendGrpInfo` $(\{G_i, Sep(x_i) \cup \{x_i\}, |Sep(x_i)| + 1\})$
13 **else**
14 $G_i \leftarrow G$, `SendGrpInfo` $(\{G_i, DIM', {n^*_{dim}}'\})$
15 **Function** `SendGrpInfo` $(\{G, DIM, n^*_{dim}\})$**:**
16 **if** $|C(x_i)| > 1$ **then** send GROUP$(\{NIL, \emptyset, 0\})$ to $\forall x_k \in C(x_i)$.
17 **else if** $|C(x_i)| = 1$ **then** send GROUP$(\{G, DIM, n^*_{dim}\})$ to $x_k \in C(x_i)$.

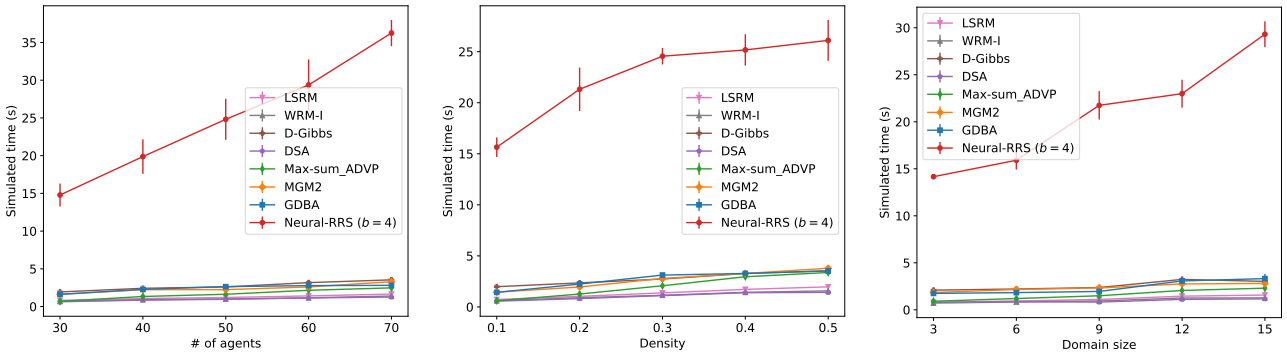
The procedure begins with the root agent creating group ID, and propagating the related information to its children (line 1-3). That is, if it has more than one child, then the group cannot be extended and a NIL group is communicated to its children (line 16). Otherwise it propagates the group ID G, running regret table dimensions DIM and the minimum number of dimensions in the group n^*_{dim} to its child (line 17). When an agent received a GROUP message, it creates a new group if (1) the received group ID is NIL (line 4-7), or (2) the difference budget is violated (line 10-12). Otherwise it extends the current group by augmenting the running regret table dimensions (line 14). The procedure ends if every non-root agent receives a GROUP message.

D Additional Experimental Results

In this section, we present the additional experimental results. In our experiments, we developed a Python framework for simulating message-passing algorithms and used it to implement all algorithms. For Neural-RRS, we use Pytorch [Paszke *et al.*, 2019] to implement deep neural networks.

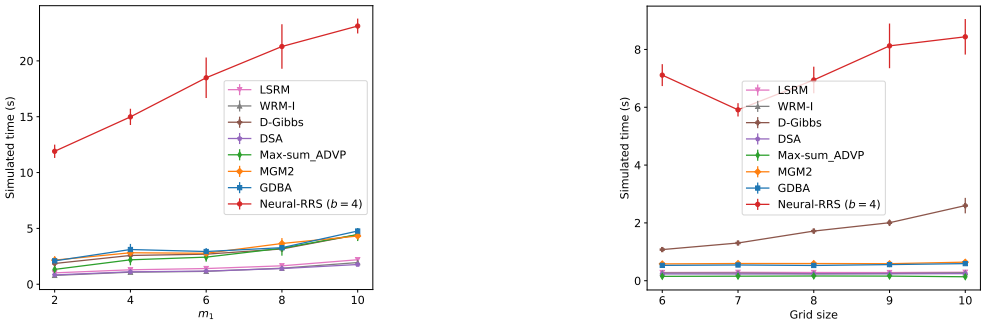
D.1 Results on Simulated Runtime

Fig.8-9 present the simulated runtime [Sultanik *et al.*, 2008] results of different algorithms on random DCOPs and structured problems, respectively. It can be seen that all traditional methods terminate very quickly. That is no surprise since they essentially perform table lookup, leading to lower overall simulated runtime. In contrast, our Neural-RRS incurs relatively



(a) $p_1 = 0.25$, $|D_i| = 10$ (b) 50 agents, $|D_i| = 10$ (c) 50 agents, $p_1 = 0.25$

Figure 8: Simulated runtime results on random DCOPs



(a) Scale-free network problems (b) Sensor network problems

Figure 9: Simulated runtime results on structured problems

higher simulated runtime due to the online training procedure. However, as we have shown in Fig.3-4, Neural-RRS can find significantly higher quality solutions than state-of-the-art methods, making it a powerful alternative for distributed constraint reasoning. Therefore, our method is suitable for the scenarios where high-quality solutions are desired while the tolerance for latency is relatively high. A possible scenario could be maritime navigation [Hirayama *et al.*, 2019] where multiple vessels coordinate their speed and direction to avoid collision. In such scenario, high-quality solutions are desired since we aim to avoid collision, while the computation latency has minor influence since both direction and speed of a vessel cannot change drastically.

D.2 Results on Total Information Exchanged

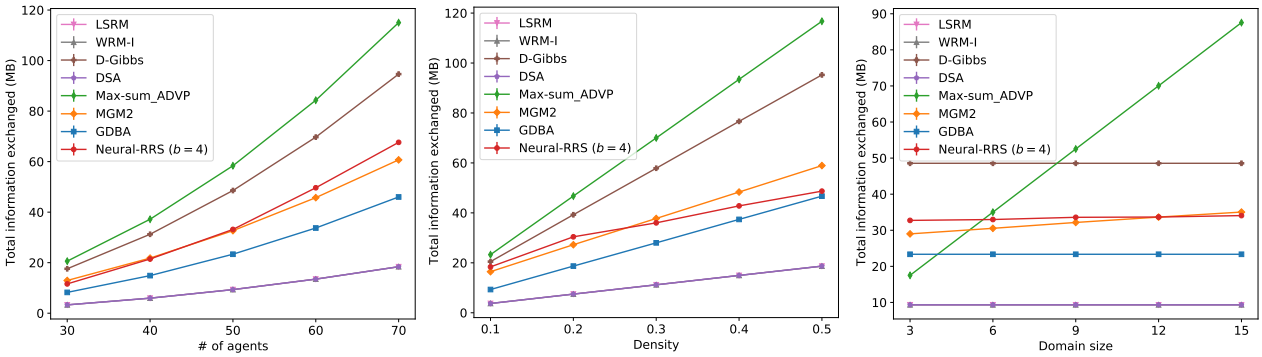
Fig.3-4 present the size of total information exchanged during the solving process of each algorithm on random and structured problem, respectively. Generally, DSA, LSRM and WRMI have small communication overheads since they just communicate values between neighbors. Max-sum_ADVP, on the other hand, needs to communicate the belief vector of length $|D_i|$ over the whole problem in each round. Therefore, it incurs the worst communication overhead among the compared algorithms in most of test cases. MGM2 and GBDA require multiple communication cycles per rounds, which requires more communication overheads than DSA. Finally, our algorithm incurs mild network traffic than Max-sum_ADVP and D-Gibbs excepts on sensor network problems.

D.3 Results on Different Hyper-parameters

Fig.5-6 present the results when vary the difference budget b and the number of training step h, respectively. It can be seen that increasing the difference budget can drastically reduce the overall runtime at slightly loss of solution quality. We therefore select $b = 4$ as it can produce high-quality solutions but requires significantly less runtime. Similarly, increasing the number of training steps helps to improve solution quality, but the runtime also increases significantly. Hence, $h = 2$ could be a good choice since it achieves similar performance with much lower runtime.

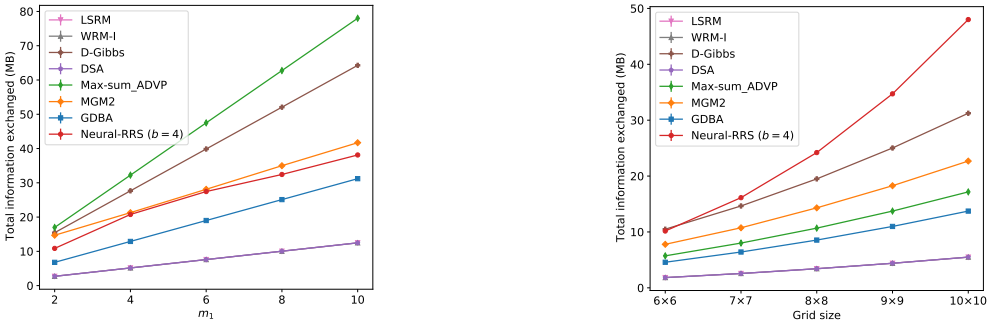
References

[Arslan *et al.*, 2007] Gürdal Arslan, Jason R Marden, and Jeff S Shamma. Autonomous vehicle-target assignment: A game-theoretical formulation. *Journal of Dynamic Systems, Measurement and Control*, 129(5):584–596, 2007.



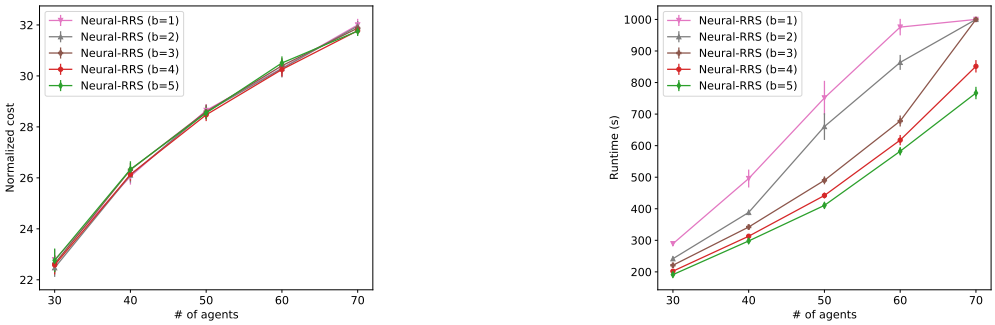
(a) $p_1 = 0.25$, $|D_i| = 10$ (b) 50 agents, $|D_i| = 10$ (c) 50 agents, $p_1 = 0.25$

Figure 10: Size of total information exchanged on random DCOPs



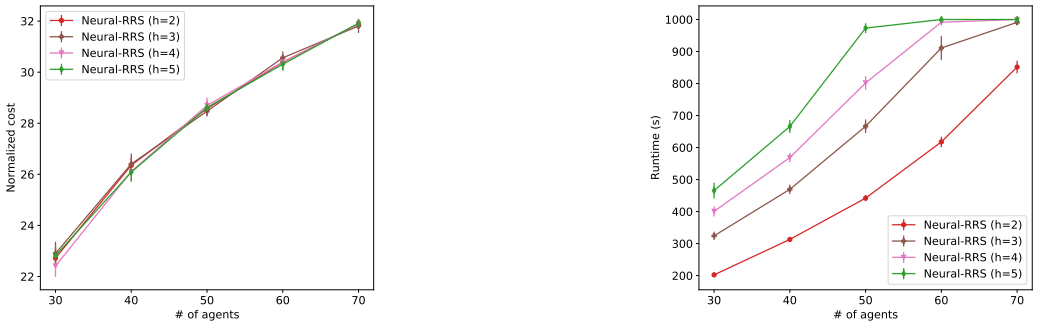
(a) Scale-free network problems (b) Sensor network problems

Figure 11: Size of total information exchanged on structured problems



(a) Solution quality when varying difference budget b (b) Runtime when varying difference budget b

Figure 12: Results on random DCOPs when varying difference budget b



(a) Performance when varying the number of training steps h (a) Runtime when varying the number of training steps h

Figure 13: Results on random DCOPs when varying the number of training steps h

[Atlas *et al.*, 2008] James Atlas, Matt Warner, and Keith Decker. A memory bounded hybrid approach to distributed constraint optimization. In *Proceedings of 10th International Workshop on Distributed Constraint Reasoning*, pages 37–51, 2008.

[Brown *et al.*, 2019] Noam Brown, Adam Lerer, Sam Gross, and Tuomas Sandholm. Deep counterfactual regret minimization. In *ICML*, pages 793–802, 2019.

[Chapman *et al.*, 2011] Archie C. Chapman, Alex Rogers, Nicholas R. Jennings, and David S. Leslie. A unifying framework for iterative approximate best-response algorithms for distributed constraint optimization problems. *Knowledge Eng. Review*, 26(4):411–444, 2011.

[Chen *et al.*, 2018] Ziyu Chen, Yanchen Deng, Tengfei Wu, and Zhongshi He. A class of iterative refined Max-sum algorithms via non-consecutive value propagation strategies. *Autonomous Agents and Multi-Agent Systems*, 32(6):822–860, 2018.

[Chen *et al.*, 2019] Dingding Chen, Yanchen Deng, Ziyu Chen, Wenxin Zhang, and Zhongshi He. HS-CAI: A hybrid DCOP algorithm via combining search with context-based inference. *CoRR*, abs/1911.12716, 2019.

[Cohen *et al.*, 2020] Liel Cohen, Rotem Galiki, and Roie Zivan. Governing convergence of Max-sum on DCOPs through damping and splitting. *Artificial Intelligence*, 279:103212, 2020.

[Deng *et al.*, 2019] Yanchen Deng, Ziyu Chen, Dingding Chen, Xingqiong Jiang, and Qiang Li. PT-ISABB: A hybrid tree-based complete algorithm to solve asymmetric distributed constraint optimization problems. In *AAMAS*, pages 1506–1514, 2019.

[Farinelli *et al.*, 2008] Alessandro Farinelli, Alex Rogers, Adrian Petcu, and Nicholas R Jennings. Decentralised coordination of low-power embedded devices using the Max-sum algorithm. In *AAMAS*, pages 639–646, 2008.

[Gershman *et al.*, 2009] Amir Gershman, Amnon Meisels, and Roie Zivan. Asynchronous forward bounding for distributed COPs. *Journal of Artificial Intelligence Research*, 34:61–88, 2009.

[Grinshpoun *et al.*, 2019] Tal Grinshpoun, Tamir Tassa, Vadim Levit, and Roie Zivan. Privacy preserving region optimal algorithms for symmetric and asymmetric DCOPs. *Artificial Intelligence*, 266:27–50, 2019.

[Hirayama and Yokoo, 1997] Katsutoshi Hirayama and Makoto Yokoo. Distributed partial constraint satisfaction problem. In *CP*, pages 222–236, 1997.

[Hirayama *et al.*, 2019] K Hirayama, K Miyake, T Shiotani, and T Okimoto. DSSA+: Distributed collision avoidance algorithm in an environment where both course and speed changes are allowed. *International Journal on Marine Navigation and Safety of Sea Transportation*, 13(1):117–123, 2019.

[Hoang *et al.*, 2018] Khoi D Hoang, Ferdinando Fioretto, William Yeoh, Enrico Pontelli, and Roie Zivan. A large neighboring search schema for multi-agent optimization. In *CP*, pages 688–706, 2018.

[Kim and Lesser, 2014] Yoonheui Kim and Victor Lesser. DJAO: A communication-constrained DCOP algorithm that combines features of ADOPT and Action-GDL. In *AAAI*, pages 2680–2687, 2014.

[Litov and Meisels, 2017] Omer Litov and Amnon Meisels. Forward bounding on pseudo-trees for DCOPs and ADCOPs. *Artificial Intelligence*, 252:83–99, 2017.

[Maheswaran *et al.*, 2004] Rajiv T Maheswaran, Jonathan P Pearce, and Milind Tambe. Distributed algorithms for DCOP: A graphical-game-based approach. In *ISCA PDCS*, pages 432–439, 2004.

[Modi *et al.*, 2005] Pragnesh Jay Modi, Wei-Min Shen, Milind Tambe, and Makoto Yokoo. Adopt: Asynchronous distributed constraint optimization with quality guarantees. *Artificial Intelligence*, 161(1-2):149–180, 2005.

[Morrill, 2016] Dustin Morrill. Using regret estimation to solve games compactly. Master's thesis, University of Alberta, 2016.

[Netzer *et al.*, 2012] Arnon Netzer, Alon Grubshtein, and Amnon Meisels. Concurrent forward bounding for distributed constraint optimization problems. *Artificial Intelligence*, 193:186–216, 2012.

[Nguyen *et al.*, 2019] Duc Thien Nguyen, William Yeoh, Hoong Chuin Lau, and Roie Zivan. Distributed Gibbs: A linear-space sampling-based DCOP algorithm. *Journal of Artificial Intelligence Research*, 64:705–748, 2019.

[Okamoto *et al.*, 2016] Steven Okamoto, Roie Zivan, and Aviv Nahon. Distributed breakout: Beyond satisfaction. In *IJCAI*, pages 447–453, 2016.

[Ottens *et al.*, 2017] Brammert Ottens, Christos Dimitrakakis, and Boi Faltings. DUCT: An upper confidence bound approach to distributed constraint optimization problems. *ACM Transactions on Intelligent Systems and Technology*, 8(5):69:1–69:27, 2017.

[Paszke *et al.*, 2019] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In *NeurIPS*, pages 8024–8035, 2019.

[Petcu and Faltings, 2005] Adrian Petcu and Boi Faltings. A scalable method for multiagent constraint optimization. In *IJCAI*, pages 266–271, 2005.

[Petcu and Faltings, 2007] Adrian Petcu and Boi Faltings. MB-DPOP: A new memory-bounded algorithm for distributed optimization. In *IJCAI*, pages 1452–1457, 2007.

[Rogers *et al.*, 2011] Alex Rogers, Alessandro Farinelli, Ruben Stranders, and Nicholas R Jennings. Bounded approximate decentralised coordination via the max-sum algorithm. *Artificial Intelligence*, 175(2):730–759, 2011.

[Sultanik *et al.*, 2008] Evan A Sultanik, Robert N Lass, and William C Regli. DCOPolis: A framework for simulating and deploying distributed constraint reasoning algorithms. In *AAMAS*, pages 1667–1668, 2008.

[Tammelin *et al.*, 2015] Oskari Tammelin, Neil Burch, Michael Johanson, and Michael Bowling. Solving heads-up limit Texas hold'em. In *IJCAI*, pages 645–652, 2015.

[Vinyals *et al.*, 2011] Meritxell Vinyals, Juan A Rodriguez-Aguilar, and Jesús Cerquides. Constructing a unifying theory of dynamic programming DCOP algorithms via the generalized distributive law. *Autonomous Agents and Multi-Agent Systems*, 22(3):439–464, 2011.

[Waugh *et al.*, 2015] Kevin Waugh, Dustin Morrill, James Andrew Bagnell, and Michael H. Bowling. Solving games with functional regret estimation. In *AAAI*, pages 2138–2145, 2015.

[Yeoh *et al.*, 2010] William Yeoh, Ariel Felner, and Sven Koenig. BnB-ADOPT: An asynchronous branch-and-bound DCOP algorithm. *Journal of Artificial Intelligence Research*, 38:85–133, 2010.

[Zhang *et al.*, 2005] Weixiong Zhang, Guandong Wang, Zhao Xing, and Lars Wittenburg. Distributed stochastic search and distributed breakout: Properties, comparison and applications to constraint optimization problems in sensor networks. *Artificial Intelligence*, 161(1-2):55–87, 2005.

[Zivan *et al.*, 2014] Roie Zivan, Steven Okamoto, and Hilla Peled. Explorative anytime local search for distributed constraint optimization. *Artificial Intelligence*, 212:1–26, 2014.

[Zivan *et al.*, 2017] Roie Zivan, Tomer Parash, Liel Cohen, Hilla Peled, and Steven Okamoto. Balancing exploration and exploitation in incomplete min/max-sum inference for distributed constraint optimization. *Autonomous Agents and Multi-Agent Systems*, 31(5):1165–1207, 2017.