Neural Regret-Matching for Distributed Constraint Optimization Problems

Yanchen Deng, **Runshen Yu**, **Xinrun Wang** and **Bo An**
School of Computer Science and Engineering, Nanyang Technological University, Singapore
ycdeng@ntu.edu.sg, runshengyu@gmail.com, {xinrun.wang,boan}@ntu.edu.sg

Abstract

Distributed constraint optimization problems (DCOPs) are a powerful model for multi-agent coordination and optimization, where information and controls are distributed among multiple agents by nature. However, most of incomplete algorithms for DCOPs are context-free, i.e., agents make a decision purely based on the state of their neighbors, which makes them prone to get trapped in poor local convergence. On the other hand, context-based algorithms use tables to exactly store all the information (e.g., costs, confidence bounds), which limits their scalability. This paper tackles the limitation by incorporating deep neural networks in solving DCOPs for the first time and presents a neural context-based sampling scheme built upon regret-matching. In the algorithm, each agent trains a neural network to approximate the regret related to its local problem under current context and performs sampling according to the estimated regret. Furthermore, to ensure exploration, we propose a regret rounding scheme that rounds small regret values to positive numbers. We theoretically show the regret bound of our algorithm and extensive evaluations indicate that our algorithm can scale up to large-scale DCOPs and significantly outperform the state-of-the-art methods.

1 Introduction

Distributed constraint optimization problems (DCOPs) [Fioretto *et al.*, 2018; Modi *et al.*, 2005] are a powerful model for multi-agent coordination and optimization, in which agents cooperatively find assignments that maximize a global objective. Due to their ability to model scenarios where information and controls are distributed among multiple agents, DCOPs have been successfully deployed into many real-world applications including radio channel allocation [Monteiro *et al.*, 2012], vessel navigation [Hirayama *et al.*, 2019] and resource management [Fioretto *et al.*, 2017].

Most algorithms for DCOPs generally follow search or inference strategy. The search-based algorithms either perform an exhaustive distributed backtrack search to guarantee optimality [Hirayama and Yokoo, 1997; Litov and Meisels, 2017; Modi *et al.*, 2005; Yeoh *et al.*, 2010] or iteratively optimize the solution by using hill-climbing [Maheswaran *et al.*, 2004; Okamoto *et al.*, 2016; Zhang *et al.*, 2005]. Additionally, some regret-based local search algorithms [Chapman *et al.*, 2011] cast a DCOP to a potential game and iteratively approximate equilibria via regret-matching [Hart and Mas-Colell, 2000]. In contrast, inference-based algorithms use dynamic programming [Petcu and Faltings, 2005; Vinyals *et al.*, 2011] or belief propagation [Farinelli *et al.*, 2008] to indirectly explore the solution space. Finally, sampling-based techniques [Nguyen *et al.*, 2019; Ottens *et al.*, 2017] are the emerging state-of-the-art incomplete methods for medium-scale DCOPs, which perform sequential sampling on a pseudo tree.

Most of incomplete algorithms for DCOPs are context-free, i.e., agents make a decision purely based on the state of their neighbors, which makes them prone to get trapped in poor local convergence since agents communicate their preferred decision based on the preferred decision of their neighbors [Farinelli *et al.*, 2008]. While DUCT [Ottens *et al.*, 2017] tries to remedy the problem by exactly storing an upper confidence bound (i.e., an optimistic estimation of the optimal utility value for the subproblem of an agent) for each context and assignment, it may need considerable rounds to make the bounds informative. Besides, exactly storing all confidence bounds requires exponential memory in the worst case, which severely limits its scalability.

Recent advances in deep reinforcement learning (deep RL) [François-Lavet *et al.*, 2018] have demonstrated the great potential of neural network for function approximation in handling large state space. Instead of storing information exactly in a table, deep RL uses deep neural networks to represent state or policy compactly and has led to tremendous success in various domains [Mnih *et al.*, 2015; OpenAI, 2018; Silver *et al.*, 2017]. Unfortunately, there is no previous work on employing function approximation to address the scalability limitation in the DCOP literature.

In this paper, we aim to develop the first scalable and efficient neural-based sampling algorithm for DCOPs. We make the following key contributions: (1) We first introduce a new context-based sampling algorithm for DCOPs built upon regret-matching. Different from existing regret-based local search algorithms, 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. Besides, compared to the existing sampling-based techniques, our method has a higher sample efficiency since it accumulates regret values according to the local problem rather than the subproblem of a variable. (2) We present a neural-based scheme to improve the scalability of the proposed sampling algorithm by using deep neural networks to approximate the regret values. To the best of our knowledge, we are the first to leverage deep neural networks into solving DCOPs. To incentivize exploration, we propose a regret rounding scheme that rounds small regret values to positive numbers. (3) We prove the regret bounds of our algorithms. (4) Extensive empirical evaluations indicate that our neural-based scheme can scale up to large-scale DCOPs and significantly outperform the state-of-the-art methods. Technical proofs, pseudo codes and additional results are provided in the appendix, which can be found at https://personal.ntu.edu.sg/boan/papers/IJCAI21_Deep_DCOP_Appendix.pdf.

2 Backgrounds

In this section, we briefly introduce DCOP, pseudo tree and regret-matching.

2.1 Distributed Constraint Optimization Problems

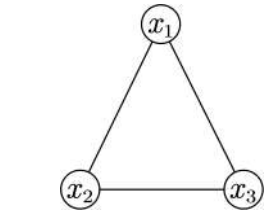
A distributed constraint optimization problem (DCOP) [Modi *et al.*, 2005] can be defined by a tuple $\langle I, X, D, F\rangle$ where $I = \{1, \dots, n\}$ is the set of agents, $X = \{x_1, \dots, x_m\}$ is the set of variables, $D = \{D_1, \dots, D_m\}$ is the set of discrete domains and $F = \{f_1, \dots, f_q\}$ is the set of constraint functions. Each variable x_i takes a value from domain D_i and each function $f_i : D_{i_1} \times \dots, \times D_{i_k} \to \mathbb{R}_{\geq 0}$ defines the cost for each possible combination of $x_{i_1}, \dots, x_{i_k}$. Finally, the objective is to find a joint assignment X^* that minimizes the total cost. That is,

$$X^* = \arg\min_X \sum_{f_i \in F} f_i. \quad (1)$$

For the sake of simplicity, we follow the common assumptions that each agent only controls a variable (i.e., $m = n$) and all constraints are binary (i.e., $f_{ij} : D_i \times D_j \to \mathbb{R}_{\geq 0}, \forall f_{ij} \in F$). Therefore, the term "agent" and "variable" can be used interchangeably and a DCOP can be visualized by a constraint graph. Fig.1(a) presents a constraint graph of a DCOP instance in which vertices and edges represent the variables and constraints of the DCOP, respectively.

2.2 Pseudo Tree

A pseudo tree [Freuder and Quinn, 1985] defines a partial ordering among variables, which is used to organize the search space or establish a communication structure. A pseudo tree can be generated by a depth-first traversal to the constraint graph which classifies the edges into tree edges and pseudo edges. Consequently, the neighbors of a variable x_i are classified into parent $P(x_i)$ (the direct ancestor which connects to x_i via a tree edge), pseudo parents $PP(x_i)$ (the direct ancestors which connect to x_i via pseudo edges), children $C(x_i)$ (the direct descendants which connect to x_i via tree edges), and pseudo children $PC(x_i)$ (the direct descendants which connect to x_i via pseudo edges). For succinctness, we also use $AP(x_i) = \{P(x_i)\} \cup PP(x_i)$ and $AC(x_i) = C(x_i) \cup PC(x_i)$ to denote all direct ancestors and descendants of x_i, respectively. Finally, the separators $Sep(x_i)$ of x_i are the set of ancestors which connect to x_i or its descendants.



(a) The constraint graph

(b) A possible pseudo tree

x_2 \ x_1	T	F
T	0	10
F	7	15

x_3 \ x_1	T	F
T	0	10
F	3	1

x_3 \ x_2	T	F
T	0	10
F	8	1

(c) Constraint functions

Figure 1: The constraint graph and a derived pseudo tree of a DCOP.

Fig.1(b) presents a possible pseudo tree corresponding to the constraint graph in Fig.1(a). The solid edges and the dotted edge are the tree edges and the pseudo edge, respectively. In the pseudo tree, x_2' neighbors $N_2 = \{x_1, x_3\}$ are classified into $P(x_2) = x_1$, $PP(x_2) = \emptyset$, $C(x_2) = \{x_3\}$ and $PC(x_2) = \emptyset$. Accordingly, $AP(x_2) = \{x_1\}$ and $AC(x_2) = \{x_3\}$. Since x_1 is constrained with x_2 and x_3, we have $Sep(x_2) = \{x_1\}$.

2.3 Regret-matching

Consider a scenario in which an agent chooses a mixed strategy $\pi^t \in \Delta_{|A|}$[1] and observes a reward vector $f^t \in \mathbb{R}^{|A|}$ in each round t, where A is the set of actions. Then the instantaneous regret of action $a \in A$ is $r^t(a) = f^t(a) - \sum_{a'} \pi^t(a') f^t(a')$ and the accumulated regret is $R^t(a) = \sum_{t'=1}^{t} r^{t'}(a)$. To improve the total reward, the agent needs to minimize the accumulated regret. Regret-matching [Hart and Mas-Colell, 2000] is an efficient algorithm for regret minimization, which computes a mixed strategy according to the positive part of accumulated regret. That is,

$$\pi^{t+1}(a) = \begin{cases} \frac{R^{t,+}(a)}{\sum_{a' \in A} R^{t,+}(a')} & \sum_{a' \in A} R^{t,+}(a') > 0 \\ \frac{1}{|A|} & \text{otherwise} \end{cases}, \quad \forall a \in A, \quad (2)$$

where $R^{t,+}(a) = \max(0, R^t(a))$ is the positive part of $R^t(a)$. Regret-matching has the regret bound of $L\sqrt{T|A|}$, where L is the largest gap in reward vectors. Because the accumulated regret grows sublinearly w.r.t. the number of rounds, regret-matching is a so-called no-regret algorithm [Blackwell, 1956].

3 Context-based Regret-matching for DCOPs

In this section, we present context-based regret-matching schemes for DCOPs. We begin with formally introducing our

[1] $\Delta_{|A|}$ is the set of probability distributions over set A

proposed sampling algorithm in Sect. 3.1. Then we show that the vanilla regret-matching could perform poorly and present the regret rounding scheme in Sect. 3.2.

3.1 Context-based Regret-matching Scheme

The proposed context-based regret-matching scheme alternatively executes a sampling phase and a backtracking phase on a pseudo tree. Starting from the root agent, each agent in sampling phase sequentially selects an assignment according to the regret vector associated with the received context, while backtracking phase is a bottom-up procedure to refine the solution and update regret values for each agent.

Let us begin with introducing notations that will be used in the algorithm. For a variable x_i in a pseudo tree, we denote an assignment to the variables in $Sep(x_i)$ (i.e., a context) as X_i and an assignment to the variables in $AC(x_i)$ as Y_i, respectively. In each round t, after collecting X_i^t and Y_i^t, the hindsight local cost $S_i^t(d_i)$ of each assignment $d_i \in D_i$ is then computed by

$$S_i^t(d_i) = \sum_{x_j \in AP(x_i)} f_{ij}(d_i, X_i^t(x_j)) + \sum_{x_k \in AC(x_i)} f_{ik}(d_i, Y_i^t(x_k)), \quad (3)$$

where $X_i^t(x_j)$ denotes the assignment of x_j in X_i^t and $Y_i^t(x_k)$ denotes the assignment of x_k in Y_i^t, respectively. Given the mixed strategy π_i^t of x_i, the expected local cost $s_i^t = \sum_{d_i \in D_i} \pi_i^t(d_i) S_i^t(d_i)$. Finally, the accumulated regret of x_i in round t is defined as

$$R_i^t(X_i, d_i) = \begin{cases} R_i^{t-1}(X_i, d_i) + s_i^t - S_i^t(d_i) & X_i = X_i^t \\ R_i^{t-1}(X_i, d_i) & \text{otherwise} \end{cases}. \quad (4)$$

In each round t, the sampling phase is initiated by the root agent. After receiving the context X_i^t from its parent, a non-leaf variable x_i first computes the mixed strategy π_i^t by

$$\pi_i^t(d_i) = \begin{cases} \frac{R_i^{t-1,+}(X_i^t, d_i)}{\sum_{d' \in D_i} R_i^{t-1,+}(X_i^t, d')} & \sum_{d' \in D_i} R_i^{t-1,+}(X_i^t, d') > 0 \\ \frac{1}{|D_i|} & \text{otherwise} \end{cases}, \ \forall d_i \in D_i. \quad (5)$$

Then it samples an assignment $d_i^t \sim \pi_i^t$ and sends the augmented context $X_i^t \cup \{x_i = d_i^t\}$ to its children $C(x_i)$. Leaf variables, however, do not have any subproblem and thus directly select the assignment that minimizes their local cost. The phase ends when all leaf agents finish sampling.

The backtracking phase is a bottom-up procedure initiated by leaf agents sending the selected assignment to their direct ancestors. After collecting all the assignments of its direct descendants Y_i^t, a variable x_i computes the hindsight local cost vector S_i^t and updates the regret R_i^t according to Eq.(3) and Eq.(4), respectively. Finally, x_i refines the solution by changing its assignment to the one with the lowest hindsight local cost and sends the assignment to its direct ancestors. The procedure ends after the root agent updates its regret and then the next round of sampling starts.

To look deeper into how context-based sampling avoids poor local convergence, consider the instance shown in Fig.1. In the first round of sampling, all non-leaf agents have no prior knowledge and sample randomly. Assume that x_1 assigns F, x_2 assigns T and thus leaf agent x_3 plays the best response F. If it does not differentiate contexts, x_2 would update its regret values by

$$\begin{cases} S_2^1 = [18, 16] \\ s_2^1 = 18 \times 0.5 + 16 \times 0.5 = 17 \end{cases} \quad \begin{cases} R_2^1(\mathrm{T}) = 17 - 18 = -1 \\ R_2^1(\mathrm{F}) = 17 - 16 = 1 \end{cases}$$

which means that x_2 cannot assign T in the next round. In fact, one can easily verify that no matter what value x_1 assigns, if x_2 assigns F and x_3 plays the best response, then the instantaneous regret for T is always negative and x_2 cannot deviate to T, which precludes the possibility of reaching the optimal solution $\{\mathrm{T}, \mathrm{T}, \mathrm{T}\}$ in subsequent rounds.

The reason behind such mis-coordination is that in context-free method each agent fails to adapt the complex behavior of its lower priority neighbors since it uses the same regret values to make decision for different contexts. In the above example, given the fact that x_1 assigns F and x_3 plays the best response, x_2 would conclude that assigning F is better than assigning T, which is not true if x_1 switches to T. In contrast, our context-based sampling scheme explicitly maintains regret values for each context, which allows agents to adapt the multi-modal behavior of its descendants. In more detail, x_2 in our method would associate the fact "F is superior over T" with context $\{x_1 = \mathrm{F}\}$, which does not bias the decision-making procedure under the context $\{x_1 = \mathrm{T}\}$.

Since it contains all assignments of separators, the size of a context message is proportional to the number of agents, i.e., $O(|I|)$. Note that in sampling phase, each non-leaf agent sends a context message to each of its children via tree edges. Therefore, there are $|I| - 1$ context messages in the sampling phase, and the total information exchanged is in $O(|I|^2)$. Backtracking phase, on the other hand, exchanges the assignments among neighbors, which induces $|F|$ messages of size $O(1)$. Therefore, the total size of messages exchanged in each round is in $O(|I|^2 + |F|) = O(|I|^2)$.

3.2 Regret Rounding Scheme

Regret-matching could perform poorly in the context of solving DCOPs due to insufficient exploration. Consider the simple instance in Fig.2. Assume that x_1 selects $d_1^1 = \mathrm{T}$ and x_2 selects $d_2^1 = \mathrm{R}$ in the first round of sampling. The following equations show the trace when x_1 updates its regret.

$$\begin{cases} S_1^1 = [1, 3] \\ s_1^1 = 1 \times 0.5 + 3 \times 0.5 = 2 \end{cases} \quad \begin{cases} R_1^1(\emptyset, \mathrm{T}) = 2 - 1 = 1 \\ R_1^1(\emptyset, \mathrm{F}) = 2 - 3 = -1 \end{cases}$$

According to Eq.(5) the assignment F cannot be selected by x_1 in the subsequent rounds since its regret is negative, even though $\{x_1 = \mathrm{F}, x_2 = \mathrm{L}\}$ is the optimal solution. In other words, negative regret would limit the exploration of the algorithm. Although regret-matching$^+$ [Tammelin *et al.*, 2015] mitigates the issue by resetting the negative regret to zero, selecting these assignments still depends on the positive instantaneous regret in subsequent rounds, which is highly coupled with the behavior of other agents.

We trigger the effective exploration by rounding small regret values to positive numbers. This way, all assignments have a non-zero probability in the mixed strategy and the exploration is independent of other agents' behavior. Specifically, instead of maintaining regret R_i^t for each variable x_i, we maintain the rounded regret $\bar{R}_i^t$:

$$\bar{R}_i^t(X_i, d_i) = \begin{cases} \max\left(\delta_t, \bar{R}_i^{t-1}(X_i, d_i) + s_i^t - S_i^t(d_i)\right) & X_i = X_i^t \\ \bar{R}_i^{t-1}(X_i, d_i) & \text{otherwise} \end{cases}, \quad (6)$$

where $\bar{R}_i^0(\cdot, \cdot) = 0$ and $\delta_t > 0$ is a non-decreasing term. In this example, if set $\delta_t = t^\alpha, \alpha > 0$, then $\bar{R}_1^1(\emptyset, \mathrm{T}) =$

(a) The pseudo tree

x_2 \ x_1	T	F
L	2	0
R	1	3

(b) The constraint function

Figure 2: A DCOP instance with structured constraint functions

$\bar{R}_1^1(\emptyset, \text{F}) = 1$ and thus x_1 still has chance to select assignment F in subsequent rounds, regardless of the strategy selected by x_2. Theorem 1 gives an upper bound of the rounded regret and Theorem 2 provides the no-regret guarantee.

Theorem 1. *For variable x_i, let $L_i = \sum_{x_j \in N_i}(f_{ij}^+ - f_{ij}^-)$ be the largest gap in its local cost, where $f_{ij}^+ = \max_{d_i}\max_{d_j} f_{ij}(d_i, d_j)$ and $f_{ij}^- = \min_{d_i}\min_{d_j} f_{ij}(d_i, d_j)$ are the upper bound and lower bound of constraint f_{ij}, respectively. After x_i finishes T rounds of sampling, the rounded regret $\bar{R}_i^T(X_i, d_i)$ is no higher than $\sqrt{|D_i|\left(TL_i^2 + \sum_{t=1}^T \delta_t^2\right)}$ for all X_i, d_i.*

Theorem 2. *When $\lim_{t\to\infty} \frac{\delta_t}{\sqrt{t}} = 0$, the regret rounding scheme is no-regret, i.e., the total regret grows sublinearly.*

4 Scaling Up to Large Problems

In this section, we first address the memory challenge by using deep neural networks to approximate high-dimensional regret tables. Then we present a prioritized training scheme to speed up the proposed algorithm by reducing the number of non-concurrent training processes in each round.

4.1 Neural-based Sampling Scheme

A prominent issue of the sampling algorithm proposed in Sect.3 is the high memory consumption incurred by exactly storing regret values for each context. A straightforward way would be approximating the regret tables by using linear regression. However, due to their limited capacity, linear models may not be able to capture the complex patterns when the problem is large. Alternatively, Regression Regret-Matching (RRM) [Morrill, 2016; Waugh *et al.*, 2015] tries to address the issue by using regression trees to estimate the accumulated regret, but it still needs to preserve all regret values during the solving process, which eliminates the most attractive advantage of reducing memory consumption. Also, since a regression tree cannot be trained incrementally, RRM needs to re-train the model on *all* data after updating regret values, which could be extremely inefficient. As confirmed in our experimental results, given a very generous runtime limit (e.g., 1000s), RRM can only finish about 500 rounds of sampling and ends up with poor solutions.

In contrast, deep neural networks have been demonstrated to be a powerful function approximator and lead to great successes in a wide variety of domains in AI. Therefore, we aim to address the scalability challenge by parameterizing the regret tables via neural networks. In our neural-based sampling scheme, for each non-leaf variable x_i we maintain an estimator $V_i : \Pi_{x_j \in Sep(x_i)\cup\{x_i\}} D_j \to \mathbb{R}$ and a FIFO capacitied memory M_i to estimate regret and store cached regret, respectively. The input of V_i is the concatenation of the one-hot encoding of the assignment to $Sep(x_i)$ and x_i and the output is the estimated regret.

When receiving a context from its parent, a non-leaf variable x_i first retrieves the estimated regret $\hat{R}_i^{t-1}$ related to the context from either memory M_i or estimator V_i, depending on whether the context-assignment pair $\langle X_i^t, d_i\rangle$ presents in the memory. Then x_i computes the mixed strategy π_i^t according to $\hat{R}_i^{t-1}$ and perform sampling.

In the backtracking phase, after collecting all the assignments of its direct descendants, x_i trains neural network V_{i,θ_i} to minimize the mean squared error (MSE) between the estimated regret and cached regret for h steps. For each step, x_i samples a mini-batch $B \subseteq M_i$ of regret values and updates the parameters θ_i to minimize the MSE:

$$\mathcal{L}_i(\theta_i) = \frac{1}{|B|}\sum\nolimits_{\langle X_i, d_i\rangle \in B} \left(M_i(X_i, d_i) - V_{i,\theta_i}(X_i, d_i)\right)^2, \tag{7}$$

where $|B|$ is the size of the batch, $M_i(X_i, d_i)$ and $V_i(X_i, d_i)$ are the cached regret value and the estimated regret value for the context-assignment pair $\langle X_i^t, d_i\rangle$, respectively. Then x_i updates the regret according to Eq.(6). Particularly, a bootstrapping step is performed if context-assignment pair $\langle X_i^t, d_i\rangle$ does not present in the memory. Theorem 3 presents the regret bound of the neural-based sampling scheme.

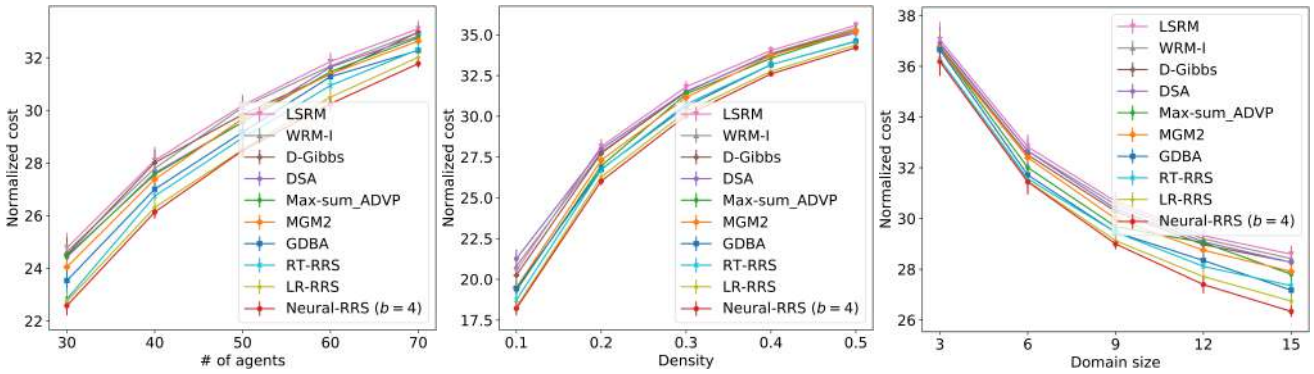
Theorem 3. *For variable x_i and round T, the regret $R^T(X_i, d_i)$ is no higher than $\sqrt{L_i|D_i|\left((4\sqrt{|D_i|}\delta_T + L_i)T + 4\sum_{t=1}^T\sum_{d_i'}\epsilon_i^t(X_i, d')^2\right)}$ for all X_i, d_i, where $\epsilon_i^t(X_i, d_i') = \hat{R}_i^t(X_i, d_i') - \bar{R}_i^t(X_i, d_i')$.*

4.2 Prioritized Training Scheme

One potential issue of the proposed neural-based sampling scheme could be the high latency incurred by non-concurrent training processes in the backtracking phase of each round. More specifically, agents in a chain structure sequentially train their neural network, which is not desirable when there are long chains in a pseudo tree. Unfortunately, since a pseudo tree is generated by depth-first traversal, it usually has few branches when the problem is large, which makes the training quite time-consuming.

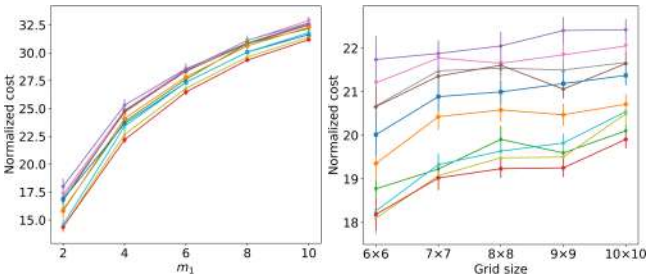
Therefore, we propose a prioritized training scheme to reduce the training latency by selecting several *key agents* in each chain structure to perform training procedure in each round. We confine our training scheme to chain structures because agents in different branches (1) can perform in parallel in real-world scenarios and (2) need additional efforts to coordinate training process.

In more detail, in preprocessing phase, we first cluster the agents into different groups according to their position in the pseudo tree and the similarity of dimensions of their regret table. That is, starting from an empty group and the first agent of the chain, we extend the group by adding the current agent until (1) the agent has more than one child, or (2) there is an agent whose regret table dimensions differ by b variables from the union of the ones of remaining agents in the group,



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

Figure 3: Performance comparison on random DCOPs



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

Figure 4: Performance comparison on structured problems

where b is a user-specified difference budget to control the size of each group. Then a new group is created and this procedure repeats until each agent belongs to a group.

In backtracking phase, we schedule the training processes of each group G according to the prediction error of each agent. That is, each agent i maintains a discounted error e_i^t. Each time agent i backtracks, it performs training with the probability $e_i^t / \sum_{j \in G} e_j^t$. The discounted error is updated by the absolute error between the predicted regret and the cached regret under current context with discount factor γ. That is,

$$e_i^t = \gamma e_i^{t-1} + (1-\gamma) \sum\nolimits_{d_i \in D_i} |M_i(X_i^t, d_i) - V_{i,\theta_i}(X_i^t, d_i)|.$$

5 Empirical Evaluations

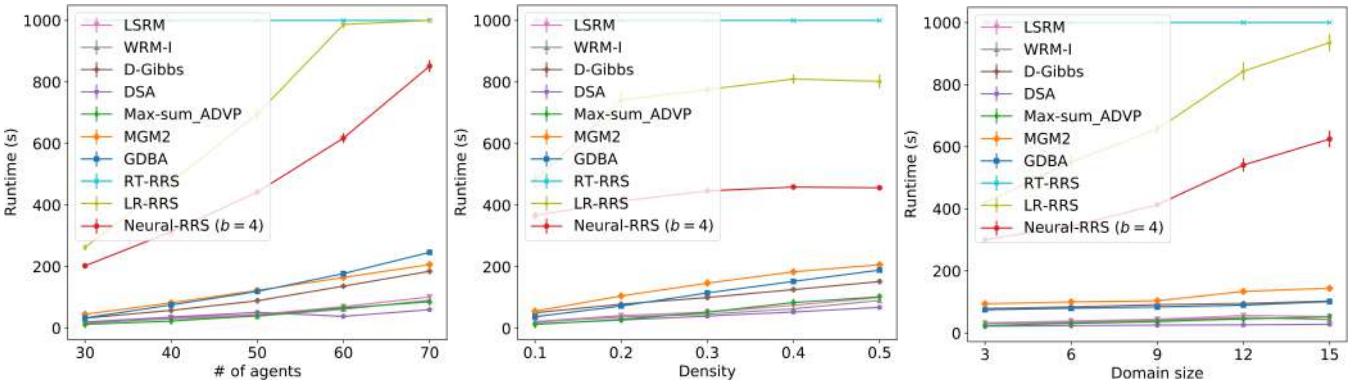
We empirically evaluate our proposed neural-based sampling scheme (Neural-RRS) on standard benchmark problems including random DCOPs, scale-free network problems and sensor network problems. We set $\delta_t = t^{0.45}$ and consider each estimator as a neural network with two hidden layers. Each hidden layer has 16 neurons and uses `relu` as the activation function. Each time the neural networks are trained by 2 steps of mini-batch stochastic gradient descent (SGD) with a batch size of 32. We use Adam optimizer [Kingma and Ba, 2014] with a learning rate of 2×10^{-3} to update parameters. Finally, we set difference budget $b = 4, \gamma = 0.9$ and the capacity of the memory to 5000 regret values.

The baselines we consider include DSA-C [Zhang *et al.*, 2005], GDBA [Okamoto *et al.*, 2016] as representative local search algorithms, MGM-2 [Maheswaran *et al.*, 2004] as a representative k-OPT algorithm, D-Gibbs [Nguyen *et al.*, 2019] as a representative sampling algorithm and Max-sum_ADVP [Zivan *et al.*, 2017] as a representative belief propagation algorithm. Besides, we also include regret-based local search schemes [Chapman *et al.*, 2011] (LSRM and WRM-I) since they are context-free regret-matching algorithms for DCOPs. Finally, we also consider the variants LR-RRS and RT-RRS that use linear regression and regression tree to approximate the regret values, respectively.

We set $p = 0.8$ for DSA-C and use $\langle M, NM, T \rangle$ variant for GDBA according to [Okamoto *et al.*, 2016]. Finally, all algorithms terminate after 5000 rounds with a timeout of 1000s and report the anytime normalized cost (i.e., the best solution cost divided by the number of constraints) as the result. All experiments are conducted on an i7 octa-core workstation with 32 GB memory. For each experiment, we average the results over 50 random instances.

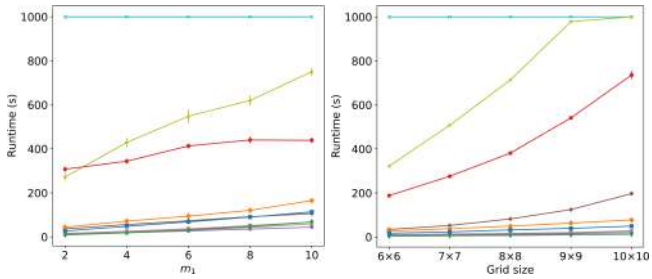
Results on random DCOPs. In a random DCOP, two agents randomly establish a constraint with a probability p_1, resulting a constraint graph with density p_1. For each constraint, we uniformly randomly select costs from $[0, 100]$. We vary the number of agents, density and domain size respectively and present the results in Fig.3. Regret-based local search algorithms explore low-quality convergences and are inferior to DSA. Similarly, D-Gibbs converges to local optima quickly and performs just like stochastic search. On the other hand, the merits of our proposed context-based regret-matching scheme are confirmed by the fact LR-RRS and Neural-RRS significantly outperform their context-free counterparts as well as MGM2 and GDBA. RT-RRS, however, fails to strictly dominate GDBA when solving the problems with large domain size. That is because regression tree cannot be trained incrementally. As a consequence, each agent in RT-RRS has to fit its model on *all* cached regret values, which is extremely inefficient and does not scale up well. In fact, RT-RRS can only finish about 500 rounds of sampling and ends up with poor solutions.

Results on scale-free network problems. In the experiment, we use Barabási-Albert model [Barabási and Albert, 1999] to generate scale-free networks. We consider the problems with $m_0 = 10$ initially connected vertices (i.e., variables). In



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

Figure 5: Runtime results on random DCOPs



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

Figure 6: Runtime results on structured problems

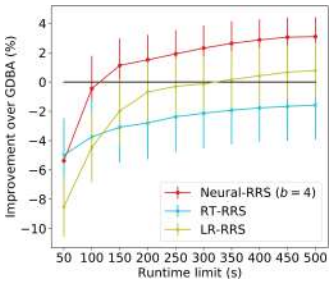


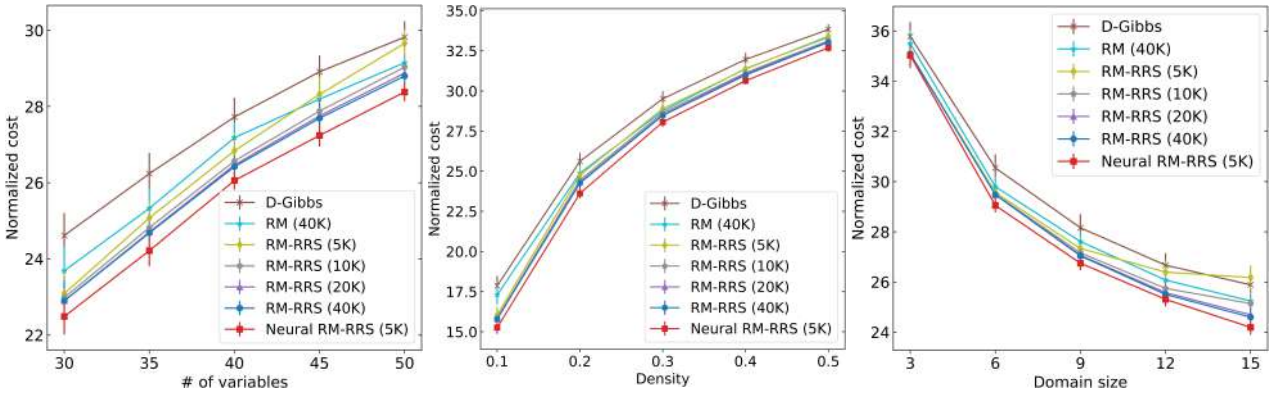
Figure 7: Convergence analysis on random DCOPs

each iteration, a new vertex is connected to m_1 vertices with a probability that is proportional to the degree of each existing vertex. We set the number of vertices to 50 and uniformly select costs from [0,100]. Fig.4(a) presents the results when varying m_1. D-Gibbs performs poorly due to the inability of exploiting different contexts and gets trapped in local optima quickly. GDBA and MGM2 perform better by using generalized breakout mechanism and coordinated moves between two agents to escape poor convergence, respectively. On the other hand, our Neural-RRS finds significantly better solutions than all the competitors on different m_1. In fact, the average improvement of Neural-RRS over GDBA is 5.6%. That is due to the fact that our algorithm stores regret for each context, which finds significantly better solutions by allowing an agent to devise different strategies for different contexts.

Results on sensor network problems. In a sensor network problem [Nguyen *et al.*, 2012; Nguyen *et al.*, 2019], sensors are placed in a 2D grid and each sensor can move along its four cardinal directions or stay stationary (i.e., each sensor has 5 possible actions). Besides, sensors are constrained with their neighboring sensors and the costs are uniformly selected from [0,100]. We vary the grid size from 6×6 (i.e., 36 variables) to 10×10 (i.e., 100 variables) and present the results in Fig.4(b). Interestingly, Max-sum_ADVP is quite competitive and finds the solutions with quality much better than the ones found by GDBA and MGM2 in this set of experiments. This might be due to the high-structured topology of sensor networks. Besides, it is worth noting that the performance of LR-RRS is similar to Neural-RRS when solving small problems (i.e., the ones with grid size of 6×6), but the performance substantially degenerates w.r.t. growing grid size. That might be due to the fact that the limited representational capability of linear models cannot capture the complex patterns in large problems. In contrast, Neural-RRS leverages powerful deep neural networks to represent regret tables, leading to the best performance on all configurations.

Runtime results. We compare the wall clock runtime of each algorithm and present the results when solving random problems and structured problems in Fig.5 and Fig.6, 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 runtime. In contrast, all functional versions of RRS require higher runtime as they need additional efforts to train estimators. In particular, RT-RRS is timed out on all test cases, and the runtime of LR-RRS grows significantly w.r.t. the growing problem scales. In contrast, combining with the prioritized training scheme, our Neural-RRS incurs relatively modest runtime requirement than LR-RRS. Notably, on the random problems with different density (i.e., Fig.5(b)) and the scale-free problems (i.e., Fig.6(a)), our Neural-RRS requires significantly less runtime than other RT-RRS and LR-RRS, which demonstrates the merits of prioritized training scheme.

To investigate the efficiency of our algorithms, we conduct a convergence analysis on random DCOPs with 50 agents, density of 0.25 and domain size of 15, and present the results in Fig.7. RT-RRS improves slowly and fails to outperform



(a) $p_1 = 0.25$, $|D_i| = 10$ (b) 40 variables, $|D_i| = 10$ (c) 40 variables, $p_1 = 0.25$

Figure 8: Ablation study on random DCOPs

GDBA in the first 500s, while LR-RRS marginally surpasses GDBA after 350s. Finally, combining with the prioritized training scheme, our Neural-RRS improves relatively fast and steadily, dominating GDBA after 150s.

Ablation study. To explicitly demonstrate the necessity of regret rounding scheme and function approximation, we consider the tabular counterpart of Neural RM-RRS (namely RM-RRS) and the one uses vanilla regret-matching (namely RM). We set the memory capacity to 5000 for Neural RM-RRS and vary the memory capacity from 5000 to 40000 for tabular counterparts to simulate different memory budgets in real-world scenarios. Fig. 8 presents the results on random DCOPs. Here we omit the results of RM under different memory capacities due to their similarity.

It can be clearly seen that RM is inferior to RM-RRS under the same memory capacity. In fact, given smaller memory capacity (e.g., 10000), RM-RRS can still outperform RM in the most cases. That is because the negative regret values in RM would restrict exploration and eventually lead to poor results. In contrast, RM-RRS triggers effective exploration by actively rounding the negative regret to small positive values, ensuring that all assignments have a non-zero probability.

On the other side, exploration triggered by rounded regret could also lead to a large number of different contexts. Therefore, given limited budget (e.g., 5000), RM-RRS quickly runs out of memory before finding a good solution, leading to poor performance when solving large-scale problems, while Neural RM-RRS with the same memory constraint still achieves the best performance. Therefore, the scalability of RM-RRS is severely restricted by its tabular nature, which is successfully addressed by neural function approximation.

6 Conclusion

Most of incomplete algorithms for DCOPs are context-free, which usually leads to low-quality convergence. While context-based methods tries to remedy the problem by explicitly storing information for each context, they usually suffer from low sample efficiency and high memory consumption which prohibit them from scaling up to large problems. In this paper, we tackle the issues by proposing Neural-RRS, the first neural context-based algorithm for DCOPs, built upon regret-matching. The algorithm overcomes the pathology of low sample efficiency by accumulating regret according to the local problem of each variable. To address the scalability challenge incurred by exactly storing regret for each context, Neural-RRS approximates the regret tables by deep neural networks. Besides, we propose a prioritized training scheme to reduce the training latency. Finally, the algorithm uses a regret rounding scheme that rounds small regret values to positive numbers to ensure exploration. We theoretically show the regret bound and the extensive evaluations indicate that our algorithm can scale up to large problems and significantly outperform the state-of-the-art methods.

Acknowledgements

This research was supported by the National Research Foundation, Singapore under its AI Singapore Programme (AISG Award No: AISG-RP-2019-0013), National Satellite of Excellence in Trustworthy Software Systems (Award No: NSOE-TSS2019-01), and NTU.

References

[Barabási and Albert, 1999] Albert-László Barabási and Réka Albert. Emergence of scaling in random networks. *Science*, 286(5439):509–512, 1999.

[Blackwell, 1956] David Blackwell. An analog of the minimax theorem for vector payoffs. *Pacific Journal of Mathematics*, 6(1):1–8, 1956.

[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.

[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.

[Fioretto *et al.*, 2017] Ferdinando Fioretto, William Yeoh, Enrico Pontelli, Ye Ma, and Satishkumar J Ranade. A distributed constraint optimization (DCOP) approach to the

economic dispatch with demand response. In *AAMAS*, pages 999–1007, 2017.

[Fioretto *et al.*, 2018] Ferdinando Fioretto, Enrico Pontelli, and William Yeoh. Distributed constraint optimization problems and applications: A survey. *Journal of Artificial Intelligence Research*, 61:623–698, 2018.

[François-Lavet *et al.*, 2018] Vincent François-Lavet, Peter Henderson, Riashat Islam, Marc G. Bellemare, and Joelle Pineau. An introduction to deep reinforcement learning. *Foundations and Trends in Machine Learning*, 11(3-4):219–354, 2018.

[Freuder and Quinn, 1985] Eugene C Freuder and Michael J Quinn. Taking advantage of stable sets of variables in constraint satisfaction problems. In *IJCAI*, volume 85, pages 1076–1078, 1985.

[Hart and Mas-Colell, 2000] Sergiu Hart and Andreu Mas-Colell. A simple adaptive procedure leading to correlated equilibrium. *Econometrica*, 68(5):1127–1150, 2000.

[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.

[Kingma and Ba, 2014] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 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.

[Mnih *et al.*, 2015] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015.

[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.

[Monteiro *et al.*, 2012] Tânia L Monteiro, Guy Pujolle, Marcelo E Pellenz, Manoel C Penna, and Richard Demo Souza. A multi-agent approach to optimal channel assignment in WLANs. In *WCNC*, pages 2637–2642, 2012.

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

[Nguyen *et al.*, 2012] Duc Thien Nguyen, William Yeoh, and Hoong Chuin Lau. Stochastic dominance in stochastic dcops for risk-sensitive applications. In *AAMAS*, pages 257–264, 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.

[OpenAI, 2018] OpenAI. OpenAI Five. https://blog.openai.com/openai-five/, 2018.

[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.

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

[Silver *et al.*, 2017] David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, et al. Mastering the game of go without human knowledge. *Nature*, 550(7676):354–359, 2017.

[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.*, 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.