Probabilistic Inference Based Message-Passing For Resource Constrained DCOPs

Supriyo Ghosh[†], Akshat Kumar[‡], and Pradeep Varakantham[‡]

School of Information Systems, Singapore Management University
[†]supriyog.2013@phdis.smu.edu.sg [†]{akshatkumar,pradeepv}@smu.edu.sg

Abstract. Distributed constraint optimization (DCOP) is an important framework for coordinated multiagent decision making. We address a practically useful variant of DCOP, called resource-constrained DCOP (RC-DCOP), which takes into account agents' consumption of shared limited resources. We present a promising new class of algorithm for RC-DCOPs by translating the underlying coordination problem to probabilistic inference. Using inference techniques such as expectation-maximization and convex optimization machinery, we develop a novel convergent message-passing algorithm for RC-DCOPs. Experiments on standard benchmarks show that our approach provides better quality than previous best DCOP algorithms and has much lower failure rate. Comparisons against an efficient centralized solver show that our approach provides near-optimal solutions, and is significantly faster on larger instances.

1 Introduction

Distributed constraint optimization (DCOP) is a general framework for coordinated decision making by a team of agents [25, 15, 19, 20]. DCOPs have been used to model several multiagent coordination problems [14, 8, 12, 26]. In DCOPs, agents control a set of variables with constraint or utility functions defined over subsets of variables. The task for agents is to assign values to variables to maximize the global utility using only local coordination among them.

In several real world applications, agents consume multiple shared resources with limited capacity. For e.g., in distributed meeting scheduling, agents' schedule is constrained by their travel budget; in sensor networks, sensors may have limited battery. The coordination problem is now to optimize the global objective, while also respecting the resource limit for each resource. To address such settings, the framework of resource-constrained DCOPs (RC-DCOPs) has been developed [3, 17, 4, 18], and has been utilized in applications such as distributed management of smart grids [8, 16].

RC-DCOPs have been solved by extending complete and optimal DCOP search algorithms such as ADOPT [19, 3, 17], and by adding support for resources to optimal dynamic programming based DPOP algorithm [20, 8, 18]. However, progress remains slow for developing *approximate* solvers for RC-DCOP that can provide scalable and good quality solutions in the presence of resource constraints. We show empirically that adding resources as a generic n-ary constraint to be solved using state-of-the-art approximate solvers such as max-sum (MS) [22] makes the algorithm unstable leading to high failure rate; implying that no resource-feasible solution was returned by

the algorithm. Therefore, our work develops a message-passing algorithm that explicitly addresses resource constraints, is guaranteed to converge, has low failure rate and provides high quality solutions over a range of benchmarks when compared against an efficient centralized constraint solver [5, 1].

Our work is motivated by the recently developed connections between decision making and probabilistic inference. Such planning-as-inference paradigm allows adoption of well known inference techniques such as expectation-maximization (EM) [6] for single agent planning [24, 23] and also multiagent planning [11]. Such inference based approach has also been applied to the problem of MAP estimation in graphical models [10], which is (almost) equivalent to the DCOP problem [9]. We extend the approach of [10] to RC-DCOPs. However, addressing resource constraints within the EM framework proves challenging as unlike the setting in [10], EM for RC-DCOP *does not* admit closed form solutions. Therefore, we combine several tools from convex optimization machinery (such as dual optimization, block coordinate descent) and algebra (polynomial root finding) in a novel way to derive the EM algorithm for RC-DCOPs. EM is easily implementable using local message-passing among agents, and is highly scalable. Unlike approaches such as MS, EM is guaranteed to converge. Empirically, we show that EM provides similar or better quality than MS, has low failure rate even under tight resource constraints and also proves highly competitive to an efficient centralized constraint solver.

2 The DCOP Framework

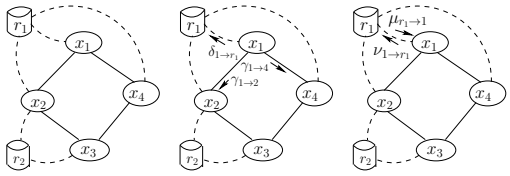
This section introduces the DCOP and the resource constrained DCOP (RC-DCOP) [3, 17] frameworks. A DCOP is defined using the tuple $\langle \mathcal{X}, \mathcal{D}, \Theta \rangle$. The set $\mathcal{X} = \{x_1, \ldots, x_n\}$ is a set of n variables; $\mathcal{D} = \{D_1, \ldots, D_n\}$ is the (finite) domain of possible values a variable can take. The set $\Theta = \{\ldots, \theta_{ij}, \ldots\}$ is the set of utility or constraint functions. A constraint function between variables x_i and x_j is defined as $\theta_{ij} : D_i \times D_j \rightarrow \Re$. We assumed w.l.o.g. that each function involves two variables.

The DCOP framework can be represented using a *constraint network* $G = (V, E)$ as follows. There is a node i for each variables x_i. For each constraint θ_{ij}, we create an edge between the nodes i and j in the graph. The objective is to find the joint variable assignment to solve:

$$\max_{x_1,\ldots,x_n} \sum_{(i,j)\in E} \theta_{ij}(x_i, x_j) \tag{1}$$

A key property of DCOPs is that there is an agent associated with each node of the constraint graph. An agent is only aware of its shared constraints with neighbor agents. Thus, there is no centralized view of the whole problem, requiring local message-passing based coordination.

Resource Constrained DCOPs (RC-DCOPs) add support for resources to DCOPs. They include a set R of resources and a set U of requirements. The set $R = \{r_1, \ldots, r_m\}$ is the set of m resources. Each resource r_i has a fixed capacity $C(r_i)$. The set U is a collection of resource utilization functions $u_i(\cdot)$ for each agent i defined as $u_i : R \times D_i \rightarrow$



(a) (b) (c)

Fig. 1. a) Resource-augmented graph for an RC-DCOP instance; b) Message passing in outer loop c) Message passing in inner loop

$\Re^+$. The RC-DCOP framework solves the same problem as (1) with the added resource constraints as below:

$$\forall r \in R : \sum_{i \in V} u_i(r, x_i) \leq C(r) \tag{2}$$

We define a *resource-augmented* constraint network (RACN) for a RC-DCOP by creating one node for each of the m resources. For e.g., in figure 1(a), we have four agents x_1 to x_4, and two resources r_1 and r_2. We say that an agent i is involved in resource constraint r if $\exists x_i \in D_i : u_i(r, x_i) > 0$. Using this notion, we create an edge between a resource r and all the agents that are involved in r. In figure 1(a), resource r_1 involves three agents x_1, x_2 and x_4 denoted using dotted edges. Notationally, i and j denote agents and their corresponding variables x_i and x_j in RACN (see fig. 1); symbol r is used to index resources. Let $\text{Nb}(i)$ denote the agents that are immediate neighbors of agent i. For e.g., $\text{Nb}(1) = \{2, 4\}$ in fig. 1. Let $\text{Nr}(i)$ denote resources that are connected to the agent i (e.g., $\text{Nr}(2) = \{r_1, r_2\}$ in fig. 1). For a resource r, $\text{Nb}(r)$ denotes agents that are connected to it (e.g., $\text{Nb}(r_2) = \{2, 3\}$).

3 Continuous Relaxation of RC-DCOP

In this section, we first present a continuous relaxation of the DCOP and RC-DCOP problems. The continuous relaxation is essentially a quadratic program (QP) for solving the DCOP problem [21]. The basis for this QP lies in the near-equivalence of the DCOP problem and MAP estimation in graphical models [21, 9]. We associate a probability distribution $p_i(x_i)$ with each variable x_i in the DCOP. We can then write the following QP:

$$\begin{aligned} \max_{\boldsymbol{p}=\{p_1,\ldots,p_n\}} & \sum_{(i,j)\in E} \sum_{x_i, x_j} p_i(x_i) p_j(x_j) \theta_{ij}(x_i, x_j) \\ \text{s.t.} & \sum_{x_i} p_i(x_i) = 1 \; \forall i \in V \end{aligned} \tag{3}$$

The following result proved in [21] shows that such a QP relaxation is tight.

Theorem 1 *Let $f^\star_{dcop}$ denote the optimal objective for the DCOP problem* (1) *and $f^\star_{qp}$ denote the optimal objective for the QP in* (3)*. Then we have $f^\star_{dcop} = f^\star_{qp}$.*

$$\max_{\boldsymbol{p}=\{p_1,\ldots,p_n\}} \sum_{(i,j)\in E} \sum_{x_i,x_j} p_i(x_i)p_j(x_j)\theta_{ij}(x_i,x_j) \quad (4)$$

$$\text{s.t.} \quad \sum_{x_i} p_i(x_i) = 1 \qquad \forall i \in V \quad (5)$$

$$\sum_{i\in Nb(r)} \sum_{x_i} p_i(x_i)u_i(r,x_i) \leq C(r) \qquad \forall r \in R \quad (6)$$

Table 1. Quadratic programming based relaxation of RC-DCOP

Therefore, optimally solving the QP (3) in a distributed manner will also solve the DCOP problem. However, this QP is non-convex, therefore, convergence to the global optimum is not guaranteed.

RC-DCOP Relaxation Based on the QP formulation of the DCOP problem, we now present a QP formulation of the RC-DCOP problem in table 1. The key addition in the QP for RC-DCOP are constraints (6) for each resource r. This constraint says that for each resource r, the *expected* consumption of this resource by all the agents (=Nb(r)) must be less than the resource's capacity $C(r)$. Notice also that all these constraints are linear in QP parameters p_i, which would be advantageous later.

Theorem 2 *Let $f^{\star}_{rcdcop}$ denote the optimal objective for the RC-DCOP problem* (1) *subject to constraints* (2) *and $f^{\star}_{qprc}$ denote the optimal objective for the QP in table 1. Then we have $f^{\star}_{qprc} \geq f^{\star}_{rcdcop}$.*

We omit the formal proof for space reasons. It is easy to show the connection of the QP in table 1 with the LP relaxation of the knapsack problem.

Our goal in this work is to solve the QP relaxation of RC-DCOP in table 1 in a distributed manner. We achieve this goal via the following:

- We first transform the RC-DCOP problem to that of likelihood maximization (LM) in a mixture of Bayesian networks. The likelihood maximization problem is exactly equivalent to solving the QP in table 1.
- We then use the well known Expectation-Maximization (EM) framework [6] to maximize the likelihood in the Bayes net mixture. However, the M-step in this EM formulation does not admit a closed form solution. Therefore, we use tools from convex optimization such as block coordinate descent, and tools from algebra such as polynomial root finding to develop a message-passing algorithm to efficiently perform the M-step.

Once the EM algorithm has converged, we use a similar message-passing based rounding technique proposed in [21] to extract an integral variable assignment from the QP solution.

4 Expectation-Maximization For RC-DCOPs

In this section, we follow the similar strategy as in [10] to recast the RC-DCOP as a likelihood maximization problem. The key idea is to decompose the constraint network

into a mixture model of simpler Bayes nets with many hidden variables – all the variables x_i of the RC-DCOP. To incorporate the constraint functions θ's of RC-DCOP and achieve equivalence between the likelihood and the RC-DCOP objective, a special *binary* reward variable $\hat{\theta}$ is introduced with its conditional distribution proportional to potentials θ.

For each edge (i, j) in the constraint network, we create a depth-1 Bayes net (BN). Notice that we do not consider edges between resources and agents during this process. Each Bayes net consists of a binary reward variable $\hat{\theta}$ with its parents being the variables x_i and x_j. Fig. 2(a) shows the RACN for a RC-DCOP instance over four variables. Fig. 2(b) shows the equivalent mixture of Bayes nets for each of the four agent-to-agent edges in this network. The mixture random variable l (with domain being agent-agent edge set E), is used to identify the Bayes nets for the corresponding edge. It has a uniform distribution (= $1/|E|$).

The parameters to estimate in this mixture are the probabilities $p_i(x_i)$ for each node x_i. Intuitively, these are the same as the variables in the QP of table 1. Furthermore, different Bayes nets share the same parameter p_i for any common variable x_i. E.g., variable x_2 in figure 2(b) is involved in two Bayes nets for $l = (1, 2)$ and $l = (2, 3)$. Therefore, $p_2(x_2)$ is the same for these two Bayes nets. The space Θ of all the valid parameters is specified by the following linear constraints:

$$\Theta : \sum_{x_i \in D_i} p_i(x_i) = 1 \ \forall i \ ; \ \sum_{i \in Nb(r)} \sum_{x_i} p_i(x_i) u_i(r, x_i) \leq C(r) \forall r \in R$$

Non-negativity of each p_i is also included in Θ. Therefore, the constraint on valid parameters in this BN mixture replicate those of in table 1. Next we set the conditional probability distribution of the variable $\hat{\theta}$ for each of the Bayes nets. For a BN l involving variables x_i and x_j, it is set as follows:

$$P\big(\hat{\theta} = 1 | x_i, x_j, l = (i, j)\big) = \hat{\theta}_{x_i, x_j} = \frac{\theta_{ij}(x_i, x_j) - \theta_{min}}{\theta_{max} - \theta_{min}} \tag{7}$$

where θ_{max} and θ_{min} are the maximum and minimum value over all constraint functions. The probabilities $\hat{\theta}_{x_i, x_j}$ are essentially normalized constraint functions θ_{ij} for the RC-DCOP instance.

Theorem 3 *For each BN l, let the CPT of binary reward variable $\hat{\theta}$ be set as per* (7)*. Then maximizing the likelihood $L^{\boldsymbol{p}} = P(\hat{\theta} = 1; \boldsymbol{p})$ of observing the reward variable in the mixture of Bayes nets is equivalent to solving the QP relaxation of RC-DCOP in table 1.*

The proof is similar to the one in [10] that shows the equivalence of likelihood maximization and the QP formulation (3) for DCOP. The only difference in our case is that the space of possible parameters Θ includes resource constraints, which makes the likelihood maximization approach applicable to RC-DCOPs.

4.1 Expected Log-Likelihood

To derive the EM algorithm for BN mixture of figure 2(b), we assume that only the reward variable $\hat{\theta} = 1$; rest of the variables are hidden. The full-joint for a BN l is given as:

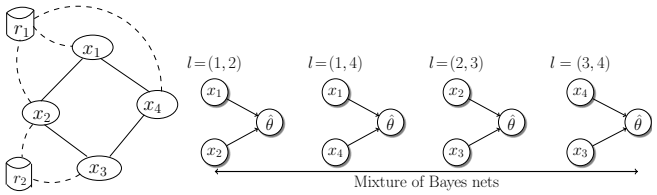


Fig. 2. a) A RC-DCOP instance; b) Equivalent mixture representation

$$P(\hat{\theta}=1, x_i, x_j, l=(i,j); \boldsymbol{p}) = \frac{1}{|E|}\hat{\theta}_{x_i,x_j} p_i(x_i) p_j(x_j) \tag{8}$$

The EM algorithm maximizes the following expected log-likelihood, $Q(\boldsymbol{p}, \boldsymbol{p}^\star)$, w.r.t. $\boldsymbol{p}^\star$ iteratively [10, 6]:

$$\sum_{l \in E} \sum_{x_{l_1}, x_{l_2}} P(\hat{\theta}=1, x_{l_1}, x_{l_2}, l; \boldsymbol{p}) \log P(\hat{\theta}=1, x_{l_1}, x_{l_2}, l; \boldsymbol{p}^\star) \tag{9}$$

where x_{l_1} and x_{l_2} denote the two variables that are involved in the BN l, $\boldsymbol{p}$ denotes the previous iteration's parameters and $\boldsymbol{p}^\star$ denote the new parameters to be optimized. We take the log of (8), and simplify $Q(\boldsymbol{p}, \boldsymbol{p}^\star)$ as follows:

$$\sum_{l \in E} \sum_{x_{l_1}, x_{l_2}} \hat{\theta}_{x_{l_1}, x_{l_2}} p_{l_1}(x_{l_1}) p_{l_2}(x_{l_2}) \big\{ \log p^\star_{l_1}(x_{l_1}) + \log p^\star_{l_2}(x_{l_2}) \big\}$$

In the above expression, we have ignored terms independent of $\boldsymbol{p}^\star$. We simplify the above expression by grouping together terms for each variable x_i:

$$Q(\boldsymbol{p}, \boldsymbol{p}^\star) \propto \sum_{i \in V} \sum_{x_i} p_i(x_i) \log p^\star_i(x_i) \underbrace{\sum_{j \in \text{Nb}(i)} \sum_{x_j} \hat{\theta}_{x_i x_j} p_j(x_j)}_{f_i(x_i)} \tag{10}$$

Using the above expression, the M-step involves solving the following convex optimization problem (minus sign used to make it a minimization problem):

$$\min_{\boldsymbol{p}^\star} - \sum_{i \in V} \sum_{x_i} p_i(x_i) f_i(x_i) \log p^\star_i(x_i) \tag{11}$$

$$\text{s.t.} \sum_{x_i} p^\star_i(x_i) = 1 \ \forall i \in V \tag{12}$$

$$\sum_{i \in Nb(r)} \sum_{x_i} p^\star_i(x_i) u_i(r, x_i) \leq C(r) \ \forall r \in R \tag{13}$$

4.2 Maximizing Expected Log-Likelihood

In this section, we detail how to solve the convex optimization problem (11). Because of the complicating resource constraints (13), this problem does not admit a closed form solution unlike the case in [10]. Therefore, we use several tools from convex optimization and algebra to develop a message-passing algorithm for this problem. Our high level approach is as follows:

- We write the dual of problem (11). The dual has simpler structure than the original problem, making optimization easier. Furthermore, as (11) is a convex optimization problem, there is no duality gap implying optimal dual quality equals optimal of (11) [2].
- To optimize the dual, we use results from convex optimization [2] that guarantee that a block coordinate descent (BCD) strategy wherein we fix all the dual variables except one, and then optimize the dual over the one variable is guaranteed to converge to the optimal dual solution. The BCD strategy gets translated into message-passing over the RACN.

Dual of (11): We first write the Lagrangian function for problem (11) by introducing dual variables $\boldsymbol{\lambda}, \boldsymbol{\mu}$ for constraints (12), (13) respectively:

$$L(\boldsymbol{p}^\star, \boldsymbol{\lambda}, \boldsymbol{\mu}) = -\sum_{i \in V} \sum_{x_i} p_i(x_i) f_i(x_i) \log p_i^*(x_i) + \sum_i \lambda_i \cdot$$
$$\big(\sum_{x_i} p^*(x_i) - 1\big) + \sum_{r \in R} \mu_r \big(\sum_{i, x_i} p_i^*(x_i) \cdot u_i(r, x_i) - C(r)\big) \tag{14}$$

The dual function is $q(\boldsymbol{\lambda}, \boldsymbol{\mu}) = \min_{\boldsymbol{p}^\star} L(\boldsymbol{p}^\star, \boldsymbol{\lambda}, \boldsymbol{\mu})$. It is found by setting derivatives of $L(\boldsymbol{p}^\star, \boldsymbol{\lambda}, \boldsymbol{\mu})$ w.r.t. each $p_i^\star$ to zero. Upon simplification, the dual is given as:

$$q(\boldsymbol{\lambda}, \boldsymbol{\mu}) = -\sum_i \sum_{x_i} p_i(x_i) f_i(x_i) \Big[\log p_i(x_i) + \log f_i(x_i)$$
$$-1 - \log\big(\lambda_i + \sum_r \mu_r u_i(r, x_i)\big)\Big] - \sum_i \lambda_i - \sum_r \mu_r C(r) \tag{15}$$

Optimizing the dual (15): We now detail the problem of maximizing dual: $\max_{\boldsymbol{\lambda}, \boldsymbol{\mu}} q(\boldsymbol{\lambda}, \boldsymbol{\mu})$. Notice also the fact that the domain of function $q(\cdot)$ includes only those $\boldsymbol{\lambda}, \boldsymbol{\mu}$ where $q(\cdot)$ is defined and is greater than $-\infty$ [2]. We also have each variable $\mu \geq 0$ [2]. These facts will be exploited later.

Block Coordinate Descent (BCD) Consider the following iterative strategy to maximize (15). We choose an arbitrary dual variable, say λ_i, fix all other variables, and optimize the function $q(\cdot)$ w.r.t. the chosen variable (λ_i). In general, this strategy is not guaranteed to converge to the optimal solution. However, the function $q(\boldsymbol{\lambda}, \boldsymbol{\mu})$ satisfies additional properties which guarantee that the BCD approach will converge to the optimal. These conditions are a) $q(\cdot)$ is continuously differentiable over its domain; b) $q(\cdot)$ is strictly concave w.r.t. each dual variable λ_i and μ_r due to the presence of $\log$ terms in (15), resulting in a unique solution for each BCD iteration [2, Proposition 2.7.1].

We now detail optimization over a single λ_i variable, *fixing all other variables*. This can be done by setting partial derivative w.r.t. λ_i of $q(\cdot)$ to zero, resulting in:

$$\sum_{x_i} \frac{p_i(x_i) \cdot f_i(x_i)}{\lambda_i + \sum_r \mu_r \cdot u_i(r, x_i)} - 1 = 0 \tag{16}$$

Roots of rational functions: Notice that (16) is a rational function in λ_i, and solving for λ_i will result in multiple values of λ_i. This complicates the BCD approach which requires a unique value for λ_i. Fortunately, we show that despite multiple λ_is satisfying (16), there is *one and only one* λ_i that is applicable in our approach. Essentially, we show that there is just one λ_i for which the dual function $q(\cdot)$ in (15) is defined, for every other possible λ_i, $q(\cdot)$ becomes undefined as it involves taking the $\log$ of a negative quantity. We start by considering a rational function of the form as:

$$g(x) = \sum_{t=1}^{T} \frac{a_t}{x + b_t} - c \tag{17}$$

where index T denotes total number of terms, $a_t > 0$, $b_t \geq 0\ \forall t$, and $c > 0$. Eq. (16) fits such a rational function categorization as numerator is positive and all variables μ are also positive. We now analyze the roots of $g(x) = 0$. Let us consider an ascending order over the terms b_t such that b_1 is smallest and b_T is the largest. For simplicity, assume that each of b_t is different and positive (> 0).

Theorem 4 *The rational function $g(x)$ of the form* (17) *has exactly one root in each interval $(-b_{t+1}, -b_t)\ \forall t = 1$ to $T-1$ and exactly one root in the interval $(-b_1, \infty)$.*

Proof. Notice that discontinuity in the function $g(x)$ occurs only at points $-b_t\ \forall t$. Consider the interval $[-b_{t+1} + \epsilon, -b_t - \epsilon]$ for any $\epsilon > 0$ such that $-b_{t+1} + \epsilon < -b_t$. We have:

$$g(-b_{t+1} + \epsilon) = \frac{a_{t+1}}{\epsilon} + \ldots \text{ and } g(-b_t - \epsilon) = \frac{a_t}{-\epsilon} + \ldots \tag{18}$$

As $\epsilon \to 0$, then we have $g(-b_{t+1} + \epsilon) \to +\infty$ and $g(-b_t - \epsilon) \to -\infty$. We also know that $g(x)$ is continuous and monotonically decreasing (using the first derivative test) in the interval $[-b_{t+1} + \epsilon, -b_t - \epsilon]$. Therefore, we can deduce that $g(x)$ crosses the horizontal axis $y = 0$ exactly once in this interval. This proves the first part of the theorem.

Using similar argument as above, we can show that there is no root in the interval $(-\infty, -b_T)$ (proof omitted). Given that the total number of roots of $g(x)$ is T and total number of intervals $(-b_{t+1}, -b_t)$ are $T - 1$, it implies that there is exactly one root in the remaining interval $(-b_1, \infty)$.

We now relate the above theorem to our problem of determining a unique solution of (16). From analyzing $\log$ terms in (15), we have:

$$\lambda_i + \sum_r \mu_r u_i(r, x_i) \geq 0\ \forall x_i \Rightarrow \lambda_i \geq \max_{x_i} -\sum_r \mu_r u_i(r, x_i) \tag{19}$$

The term $\max_{x_i} -\sum_r \mu_r u_i(r, x_i)$ is equivalent to the term $-b_1$ used in the theorem 4. Therefore, we require that a feasible λ_i must lie in the interval $(-b_1, \infty)$. Using theorem 4 we already know that there is just one root for any rational function $g(x)$ in this

Algorithm 1: solveRCDCOP($\boldsymbol{\theta}, \boldsymbol{u}$)

1 **Initialize:** $p_i^*(x_i) \leftarrow \frac{1}{|D_i|}$ $\quad \forall x_i \in D_i,\ \forall i \in \mathcal{A}$

2 **repeat**

3 $\quad p_i(x_i) \leftarrow p_i^*(x_i)$ $\quad \forall x_i \in D_i,\ \forall i \in \mathcal{A}$

4 $\quad$ *send* $\gamma_{i\to j}(x_j) \leftarrow \sum_{x_i} p_i(x_i)\hat{\theta}_{x_i x_j}$ $\quad \forall j \in \text{Nb}(i), \forall i \in \mathcal{A}$

5 $\quad$ *send* $\delta_{i\to r}(x_i) \leftarrow p_i(x_i) \sum_{k\in\text{Nb}(i)} \gamma_{k\to i}(x_i)$ $\quad \forall r \in \text{Nr}(i), \forall i \in \mathcal{A}_R$

6 $\quad \{\boldsymbol{\lambda}, \boldsymbol{\mu}\} \leftarrow$ solveBCD($\boldsymbol{p}, \boldsymbol{\gamma}, \boldsymbol{\delta}$)

7 $\quad p_i^*(x_i) \leftarrow \frac{p_i(x_i)\sum_{k\in\text{Nb}(i)}\gamma_{k\to i}(x_i)}{\lambda_i + \sum_{r\in\text{Nr}(i)}\mu_{r\to i}\cdot u_i(r,x_i)}$ $\quad \forall x_i \in D_i,\ \forall i \in \mathcal{A}_R$

8 $\quad p_i^*(x_i) \leftarrow \frac{p_i(x_i)\cdot\sum_{k\in\text{Nb}(i)}\gamma_{k\to i}(x_i)}{\sum_{x_i} p_i(x_i)\cdot\sum_{k\in\text{Nb}(i)}\gamma_{k\to i}(x_i)}$ $\quad \forall x_i \in D_i\ \forall i \in \mathcal{A}_{-R}$

9 **until** $p \neq p^*$

10 $\boldsymbol{p}^{\text{int}} \leftarrow$ **Primal_Extraction**($\boldsymbol{p}^*$)

11 **return** $\boldsymbol{p}^{\text{int}}$

range. And this unique root in the interval $(-b_1, \infty)$ is the valid solution of (16) used by BCD approach.

Maximizing (15) w.r.t. μ_r requires finding the unique value of μ_r from the following equation:

$$\sum_{i\in\text{Nb}(r),x_i} \frac{p_i(x_i)\cdot f_i(x_i)\cdot u_i(r,x_i)}{\mu_r\cdot u_i(r,x_i) + \lambda_i + \sum_{r'\in\text{Nr}(i)\setminus r}\mu_{r'}\cdot u_i(r',x_i)} = C(r)$$

As noted before, dual variables μ must be positive. Let r denotes the largest root for the above rational function, we can show that $\max(0, r)$ is the unique solution for the BCD approach.

Message-passing implementation: All the steps of the EM and the BCD approach can be implemented via message-passing over the RACN for a RC-DCOP as shown in Alg. 1 and 2. Let $\mathcal{A} = \mathcal{A}_R \cup \mathcal{A}_{-R}$ denote the set of all agents, $\mathcal{A}_R$ denotes agents that are directly connected to at least one resource and $\mathcal{A}_{-R}$ denotes agents that do not participate in any resource constraint. The EM algorithm for RC-DCOPs is a double loop message-passing algorithm as shown in alg. 1.

In the outer loop (lines 2-9 of algo. 1), two types of messages are passed among agents and resources, as shown in fig. 1(b) and lines 4 and 5 in algo. 1. Message $\gamma_{i\to j}$ (size=$|D_j|$) is passed from agent i to its neighbor agent j, $\delta_{i\to r}$ (size=$|D_i|$) is passed from agent i to connected resource r. In the inner loop (lines 4 to 14 in algo. 2), the steps of the BCD approach are implemented. In the inner loop, message $\nu_{i\to r}$ (size =$|D_i|$) is passed from agent i to connected resource r, $\mu_{r\to i}$ (size=1) is passed from resource r to agent i. Both these message types are shown in fig. 1(c).

The core of the BCD approach is implemented in lines 5 to 12 in algo. 2. Line 6 updates the μ_r variable by finding the largest root of the given rational function; similarly line 10 updates the λ_i variable. Notice that while root finding, agents and resources use the *latest* μ or ν message they have received. Main idea is that whenever a variable is updated, its updated value must be communicated to all other relevant

Algorithm 2: **solveBCD**($\boldsymbol{p}, \boldsymbol{\gamma}, \boldsymbol{\delta}$)

1 **Initialize:**
2 Set $\lambda_i \leftarrow 0$; *send* $\nu_{i\to r}(x_i) \leftarrow \lambda_i$ $\quad \forall r \in \text{Nr}(i), \forall i \in \mathcal{A}_R$
3 Set $\mu_r \leftarrow 0$; *send* $\mu_{r\to i} \leftarrow \mu_r$ $\quad \forall i \in \text{Nb}(r), \forall r \in R$
4 **repeat**
5 **for** *each resource* $r \in R$ **do**
6 Find *largest* root μ_r for $g(\mu_r)=0$:
7 $g(\mu_r) = \sum_{i\in\text{Nb}(r)} \sum_{x_i} \frac{\delta_{i\to r}(x_i) u_i(r,x_i)}{\mu_r u_i(r,x_i) + \nu_{i\to r}(x_i)} - C(r)$
8 *send* $\mu_{r\to i} \leftarrow \max(0, \mu_r)$ $\quad \forall i \in \text{Nb}(r)$
9 **for** *each agent* $i \in Nb(r)$ **do**
10 Find *largest* root λ_i for $g(\lambda_i)=0$:
11 $g(\lambda_i) = \sum_{x_i} \frac{p_i(x_i) \cdot \sum_{k\in\text{Nb}(i)} \gamma_{k\to i}(x_i)}{\lambda_i + \sum_{r\in\text{Nr}(i)} \mu_{r\to i} \cdot u_i(r,x_i)} - 1$
12 *send* $\nu_{i\to\hat{r}}(x_i) \leftarrow \lambda_i +$
13 $\sum_{r'\in\text{Nr}(i)\setminus\hat{r}} \mu_{r'\to i} u_i(r', x_i)$ $\quad \forall \hat{r} \in \text{Nr}(i)$
14 **until** ***convergence***
15 **return** $\{\boldsymbol{\lambda}, \boldsymbol{\mu}\}$

entities that depend upon the updated variable. For example, after the variable λ_i is updated in line 11, an updated message ν is sent to all the resources r connected to i in line 12. Convergence is detected in the inner loop if messages change by a value smaller than a given threshold.

Primal Extraction: Upon convergence, we extract the integral solution from $\boldsymbol{p}^\star$ using a similar rounding technique as proposed in [21, Theorem 3.2] by adding to it support for handling resources. This rounding technique also has a message-passing structure.

Complexity: Let m denote the maximum number of agents involved in any single resource; m' denote the maximum number of resources any single agent is connected to. If inner loop (lines 4 to 14 in algo. 2) is run for I iterations, then total μ messages are $O(mI|R|)$. Total number of ν messages is $O(mm'I|R|)$. In each outer loop iteration (lines 2 to 9 in algo. 1), the total number of messages exchanged (both γ and δ) equals to the number of edges in the RACN. The maximum size of any message (in inner or outer loop) is bounded by the maximum domain $|D|_{\text{max}}$ of any variable. Thus, our message-passing approach is highly efficient and scales gracefully with the number of edges or resources in the RACN.

5 Experiments

We compare our EM approach with the popular approximate DCOP algorithm max-sum (MS). Resource constraints were encoded as n-ary constraints for MS. As both the EM and MS lack quality guarantees, we also show results against an optimal efficient constraint solver 'toulbar2' [1]. We imposed a time limit of 1 hour for toulbar2. We used the MS implementation provided by the Frodo 2.0 software [13]. Our approach

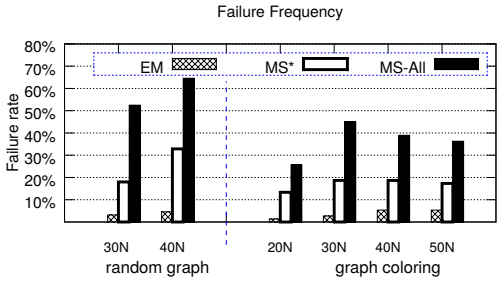


Fig. 3. Failure frequency of EM and MS on random graphs and graph coloring problems

was implemented in Python and used Bisection method for root finding. We test on two benchmarks, random graphs and graph coloring.

Random graphs: We tested with 30 and 40 node graphs with domain size $|D_i|$=5. We vary the edge density from 0.5 to 0.9 resulting in challenging problems. Each utility, $\theta_{ij}(\cdot,\cdot)$, is a random value between 1 and 10. Each resource constraint involved three agents. Total number of resources created were such that about 50% of all the agents were involved in at least one resource constraint. A 30 node problem had 5 resource constraints on an average, and a 40 node problem had about 7 resource constraints. The resource consumption of agents for each resource was also generated randomly between 1 to 5. We controlled the resource capacity $C(r)$ of each resource carefully. Let M_r, m_r denote the maximum and minimum amount of resource r respectively that can be consumed by all the involved agents. To control the tightness of capacity constraints, we use a parameter t_r varied from 0.2 to 0.6. The capacity $C(r)$ is set as $m_r + t_r(M_r - m_r)$. For each setting $\langle \#\text{Nodes}, \text{edge density}, \text{resource capacity} \rangle$, we generated 4 instances.

Failure rate: We first report on the failure rate of MS and EM for random graphs and graph coloring for varying node sizes in Figure 3. If a particular run of the algorithm fails to find a resource feasible solution for a given feasible instance, it is classified as a failed run. For MS, we ran it 3 times for each instance. We set 1000 iterations for each run and used the best solution over all the iterations. MS gave variable results for each run. In contrast, we ran EM just once for each instance, and EM's final solution is deterministic for a given initialization of parameters. We report two statistics for MS, the *best failure rate* (MS* in figure 3) denotes percentage of instances for whom none of MS's three runs produced a feasible solution. The 'MS-All' in figure 3 denotes the total percentage of runs where MS failed to find a feasible solution.

Figure 3 clearly shows the instability of the MS algorithm in the presence of resource constraints. Even the best failure rate of MS ('MS*') was as high as 32% for 40 node random graphs. The overall failure rate of MS ('MS-All') was much higher, more than 50% for random graphs and about 40% for graph coloring problems. This shows that a significant fraction of MS's runs resulted in failure. In contrast, EM's failure rate is extremely low (less than 5%) across all the problems despite EM being run just once for each instance. These results further show that by accounting for resources explicitly, the EM algorithm is significantly more stable and better than MS across different settings.

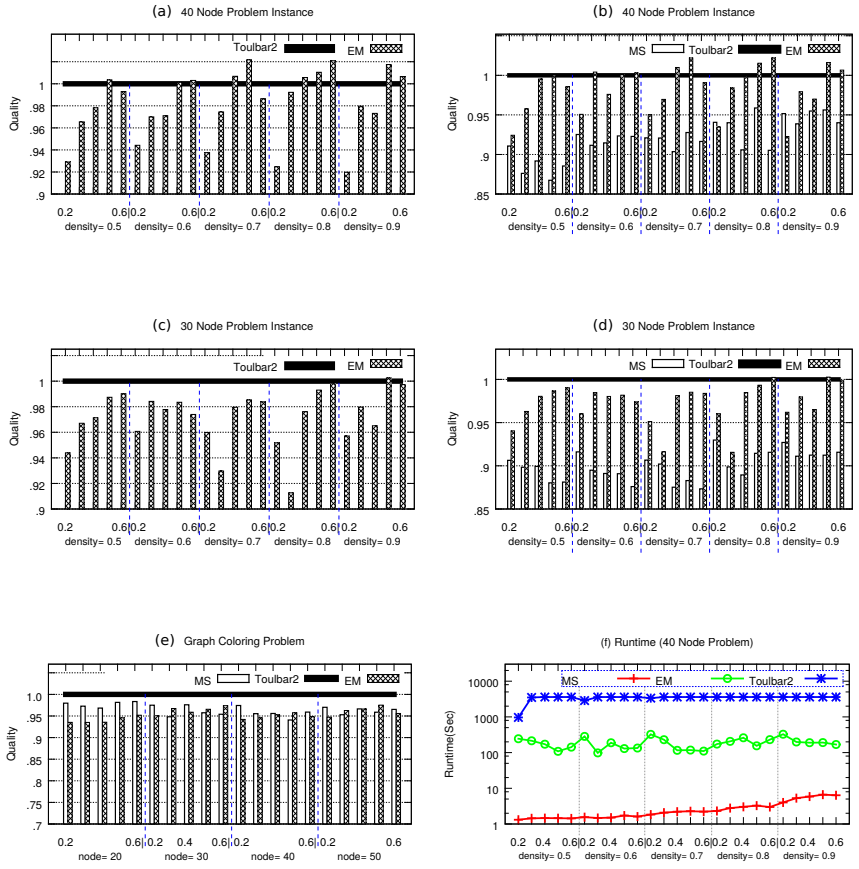


Fig. 4. Solution quality comparison a) EM and Toulbar2 on 40 node problem b) EM, MS and Toulbar2 on 40 node problem c) EM and Toulbar2 on 30 node problem d) EM, MS and Toulbar2 on 30 node problem e) EM, MS, and Toulbar2 on graph coloring problem f) Runtime comparisons for 40 node random graphs

Quality: Figure 4(a) shows (average) quality comparisons between EM and toulbar2 on common 40 node instances where both EM and toulbar2 return a resource feasible solution. The x-axis shows resource tightness varied from $t_r = 0.2$ to $t_r = 0.6$, and edge density varied from 0.5 to 0.9. We show normalized quality for EM by assigning 1 to the quality achieved by toulbar2. As we can clearly see, EM was always able to achieve a solution very close to toulbar2. Indeed, for harder instance with higher edge density, such as 'density=0.9', EM provided better quality than toulbar2, which did not find the optimal solution within 1 hour limit. The average time required by EM was 180 sec., showing that EM was significantly faster than the toulbar2 solver.

Figure 4(b) shows (average) quality comparisons between EM, MS and toulbar2 on common instances where each algorithm found a resource feasible solution. These results also show clearly that EM always provided similar or better quality than MS.

Figure 4(c) and (d) show the same previous two sets of comparisons for 30 node problems. For these problems, toulbar2 was able to achieve the optimal solution for most instances. EM achieved very close to the optimal solution. From figure 4(d) we further observe that EM provided better quality than MS.

Graph coloring: We also tested EM on graph coloring instances generated in the same fashion as [7]. Resource constraints are generated similarly as for random graphs. Figure 4(e) shows the results for 20 to 50 nodes and varying resource tightness t_r from 0.2 to 0.6 on the x-axis. These results further show that EM is highly competitive even with a centralized solver, while having a low failure rate (less than 5%) when compared against the existing MS algorithm.

Timing results: Figure 4(f) shows the (average) runtime (in log scale) for toulbar2, EM and MS for the most challenging 40 node random graphs. The toulbar2 had a time limit of 1 hour. While MS is the fastest of all, as shown before, its solution quality is worst and failure rate high. In contrast, EM almost always converged within 3 min for all the settings and also provided comparable quality solutions to toulbar2 for higher density graphs. Therefore, EM proved very effective for these challenging problems.

6 Conclusion

We presented a promising new class of algorithms based on probabilistic inference for RC-DCOPs. We showed a close connection between likelihood maximization and RC-DCOPs, and using this connection developed the EM algorithm for RC-DCOPs. By using tools from convex optimization, we showed that EM algorithm takes a message-passing structure over the constraint network. Unlike previous approaches, such as MS, EM is guaranteed to converge. Empirically, EM had significantly lower failure rate than MS even in the presence of tight resource constraints, and proved highly competitive to a centralized constraint solver.

References

1. D. Allouche, S. de Givry, and T. Schiex. Toulbar2, an open source exact cost function network solver. Technical report, INRA, 2010.
2. D. P. Bertsekas. *Nonlinear Programming*. Athena Scientific, 2nd edition, 1999.
3. E. Bowring, M. Tambe, and M. Yokoo. Multiply-constrained distributed constraint optimization. In *International Conference on Autonomous Agents and Multiagent Systems*, pages 1413–1420, 2006.
4. E. Bowring, Z. Yin, R. Zinkov, and M. Tambe. Sensitivity analysis for distributed optimization with resource constraints. In *International Conference on Autonomous Agents and Multiagent Systems - Volume 1*, pages 633–640, 2009.
5. S. de Givry, F. Heras, M. Zytnicki, and J. Larrosa. Existential arc consistency: Getting closer to full arc consistency in weighted CSPs. In *International Joint Conference on Artificial Intelligence*, pages 84–89, 2005.
6. A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical society, Series B*, 39(1):1–38, 1977.

7. A. Farinelli, A. Rogers, A. Petcu, and N. R. Jennings. Decentralised coordination of low-power embedded devices using the max-sum algorithm. In *International Joint Conference on Autonomous Agents and Multiagent Systems*, pages 639–646, 2008.
8. A. Kumar, B. Faltings, and A. Petcu. Distributed constraint optimization with structured resource constraints. In *International Joint Conference on Autonomous Agents and Multiagent Systems*, pages 923–930, 2009.
9. A. Kumar, W. Yeoh, and S. Zilberstein. On message-passing, MAP estimation in graphical models and DCOPs. In *International Workshop on Distributed Constraint Reasoning (DCR)*, pages 57–70, 2011.
10. A. Kumar and S. Zilberstein. MAP estimation for graphical models by likelihood maximization. In *Neural Information Processing Systems*, pages 1180–1188, 2010.
11. A. Kumar, S. Zilberstein, and M. Toussaint. Scalable multiagent planning using probabilistic inference. In *Proceedings of the 22nd International Joint Conference on Artificial Intelligence*, pages 2140–2146, 2011.
12. T. Léauté and B. Faltings. Coordinating logistics operations with privacy guarantees. In *International Joint Conference on Artificial Intelligence*, pages 2482–2487, 2011.
13. T. Léauté, B. Ottens, and R. Szymanek. Frodo 2.0: An open-source framework for distributed constraint optimization. In *IJCAI-09 Distributed Constraint Reasoning Workshop (DCR" 09)*, pages 160–164, 2009.
14. R. Maheswaran, M. Tambe, E. Bowring, J. Pearce, and P. Varakantham. Taking DCOP to the real world: Efficient complete solutions for distributed event scheduling. In *International Joint Conference on Autonomous Agents and Multiagent Systems*, pages 310–317, 2004.
15. R. Mailler and V. Lesser. Solving distributed constraint optimization problems using cooperative mediation. In *International Joint Conference on Autonomous Agents and Multiagent Systems*, pages 438–445, 2004.
16. T. Matsui and H. Matsuo. Considering equality on distributed constraint optimization problem for resource supply network. In *IEEE/WIC/ACM International Joint Conferences on Web Intelligence and Intelligent Agent Technology*, pages 25–32, 2012.
17. T. Matsui, H. Matsuo, M. Silaghi, K. Hirayama, and M. Yokoo. Resource constrained distributed constraint optimization with virtual variables. In *In AAAI Conference on Artificial Intelligence*, pages 120–125, 2008.
18. T. Matsui, M. Silaghi, K. Hirayama, M. Yokoo, B. Faltings, and H. Matsuo. Reducing the search space of resource constrained DCOPs. In *Principles and Practice of Constraint Programming*, pages 576–590, 2011.
19. P. Modi, W.-M. Shen, M. Tambe, and M. Yokoo. ADOPT: Asynchronous distributed constraint optimization with quality guarantees. *Artificial Intelligence*, 161(1-2):149–180, 2005.
20. A. Petcu and B. Faltings. DPOP: A scalable method for multiagent constraint optimization. In *International Joint Conference on Artificial Intelligence*, pages 266–271, 2005.
21. P. Ravikumar and J. Lafferty. Quadratic programming relaxations for metric labeling and Markov random field MAP estimation. In *International Conference on Machine Learning*, pages 737–744, 2006.
22. R. Stranders, A. Farinelli, A. Rogers, and N. R. Jennings. Decentralised coordination of mobile sensors using the max-sum algorithm. In *International Joint Conference on Artificial Intelligence*, pages 299–304, 2009.
23. M. Toussaint, L. Charlin, and P. Poupart. Hierarchical POMDP controller optimization by likelihood maximization. In *International Conference on Uncertainty in Artificial Intelligence*, pages 562–570, 2008.
24. M. Toussaint and A. J. Storkey. Probabilistic inference for solving discrete and continuous state Markov decision processes. In *International Conference on Machine Learning*, pages 945–952, 2006.

25. M. Yokoo, E. H. Durfee, T. Ishida, and K. Kuwabara. The distributed constraint satisfaction problem: Formalization and algorithms. *IEEE Transactions on Knowledge and Data Engineering*, 10:673–685, 1998.
26. R. Zivan, S. Okamoto, and H. Peled. Explorative anytime local search for distributed constraint optimization. *Artificial Intelligence*, 212(0):1 – 26, 2014.