$\mathbb{E}$[DPOP]: Distributed Constraint Optimization under Stochastic Uncertainty using Collaborative Sampling

Thomas Léauté and Boi Faltings

École Polytechnique Fédérale de Lausanne (EPFL)
Artificial Intelligence Laboratory (LIA)
firstname.lastname@epfl.ch

Abstract. Many applications that require distributed optimization also include uncertainty about the problem and the optimization criteria themselves. However, current approaches to distributed optimization assume that the problem is entirely known before optimization is carried out, while approaches to optimization with uncertainty have been investigated for centralized algorithms. This paper introduces the framework of *Distributed Constraint Optimization under Stochastic Uncertainty* (*StochDCOP*), in which random variables with known probability distributions are used to model sources of uncertainty. Our main novel contribution is a distributed procedure called *collaborative sampling*, which we use to produce several new versions of the *DPOP* algorithm for StochDCOPs. We evaluate the benefits of collaborative sampling over the simple approach in which each agent samples the random variables independently. We also show that collaborative sampling can be used to implement a new, distributed version of the *consensus* algorithm, which is a well-known algorithm for centralized, online stochastic optimization in which the solution chosen is the one that is optimal in most cases, rather than the one that maximizes the expected utility.

1 Introduction

Many optimization problems in the real world today are intrinsically distributed, and involve multiple parties having each their own coupled subproblems, and willing to collaborate to produce a solution to an overall problem. Examples include monitoring moving objects using sensor networks, search-and-rescue scenarios performed by autonomous robots, or supply chain optimization.

In yet another domain of application, consider a *Truck Task Coordination Problem* [1], in which a nation-wide company receives pickup-and-delivery requests from customers. Because of the large geographical extent of its operations, the company uses the truck fleets of franchised subcontractors scattered around the country, and faces the problem of assigning its packages to these trucks. Each truck has a dedicated pickup area; it only picks up packages that are waiting in this area, but can deliver them beyond if necessary. This problem is essentially distributed and does not lend itself well to centralized problem solving, because

subcontractors have constraints and preferences that they consider trade secrets and do not want to share. On the other hand, because they work under the same franchise, they are willing to collaborate to maximize the total value of the satisfied requests, minus the cost of pickup and delivery.

Distributed Constraint Optimization (*DCOP*) is a powerful framework that has emerged as one that is able to model such multi-agent optimization problems. Distributed algorithms have been proposed to solve DCOPs, including SynchBB [2], ADOPT [3], OptAPO [4], DPOP [5], AFB [6], and NCBB [7]. The focus in the research so far has been mainly on designing fast, complete algorithms. Very little work however has been done on algorithms capable of handling *uncertainty* in the problem data, and none of the aforementioned algorithms addresses this issue.

The relevance to real-life problems of handling uncertainty can be illustrated by revisiting the previous package delivery problem: perfect information about all packages might not be known at the time an assignment of packages to trucks must be computed, as not all packages to be delivered in a given day might be known in advance. Fortunately, historical data and statistics about customer orders are available, and can be used to produce predictions of future packages in the form of probability distributions.

Previous work has tried to extend the DCOP framework to include some form of uncertainty modeled by probability distributions. [8] proposed an extension called *DCOP with Utility Distributions*, in which each constraint's utilities are no longer simple real values, but probabilistic distributions of real values. Unfortunately, this formalism assumes that all utility distributions are independent, and fails to model the fact that two agents' utilities can depend on the same source of uncertainty. Furthermore, to our knowledge and at the time of this writing, no algorithm has been proposed that solves such extended DCOPs.

We propose to extend the DCOP formalism to that of *DCOP under Stochastic Uncertainty* (*StochDCOP*), which is a generalization of the formalism in [8]. In this new framework, *random variables* with probability distributions are used to model sources of uncertainty. The goal of the problem becomes to compute an assignment to the *decision variables* that maximizes the *expected* utility.

One of the challenges in reasoning about uncertainty is that the probability distributions for the random variables might not be known directly, or their full forms might be known but too complex to use in the optimization process. In such cases, it is frequent to resort to *sampling* in order to obtain simple, approximate representations of the probability distributions in the form of sample realizations of the random variables. While sampling is a commonly used approach in centralized optimization, its implementation in the context of distributed optimization is however not necessarily trivial. For instance, it is intuitive to expect a lower solution quality if all agents independently sample shared random variables, resulting in the agents reasoning about inconsistent samples.

To address this, we introduce a distributed procedure called *collaborative sampling*, following which all agents concerned with a given random variable agree on a common sample set for its probability distribution. This procedure

partially centralizes the sampling, while taking advantage of the problem structure and the local, limited influence of each random variable on the overall problem to retain as much parallelism as possible. We use this approach to produce new versions of the DPOP algorithm that are able to solve StochDCOPs, and we evaluate the impact of collaborative sampling against the approach where all agents use inconsistent samples.

We also show how collaborative sampling can be used to implement a new, distributed version of the *consensus* algorithm for (centralized) Online Stochastic Optimization [9]. In contrast to the traditional *expectation* approach, this algorithm chooses the solution that is optimal for the largest number of samples, rather than the one that optimizes the sampled expected utility. The *consensus* approach has been shown to perform better on online, centralized problems when the number of samples is limited by time constraints.

The rest of the paper is organized as follows. Section 2.1 recalls the DCOP formalism and the DPOP algorithm, and illustrates them on the class of Truck Task Coordination Problems. Section 3 then introduces the new StochDCOP formalism. Section 4 presents our novel collaborative sampling approach, and a set of new StochDCOP algorithms based on it, which are evaluated in Section 5. Section 6 then concludes.

2 DCOP and the Original DPOP Algorithm

This section first recalls the DCOP formalism, and then briefly presents the original DPOP algorithm.

2.1 DCOP Formalism

A discrete *Distributed Constraint Optimization Problem* (*DCOP*) is formally defined as a tuple $< \mathcal{A}, \mathcal{X}, \mathcal{D}, \mathcal{C} >$:

- $\mathcal{A} = \{a_1, \ldots, a_{|\mathcal{A}|}\}$ is a set of *agents*;
- $\mathcal{X} = \{x_1, \ldots, x_n\}$ is a set of *decision variables*, each decision variable x_i being owned by an agent $a(x_i)$;
- $\mathcal{D} = \{D_1, \ldots, D_n\}$ is a set of finite *domains* for the decision variables such that x_i takes values in D_i;
- $\mathcal{C} = \{c_1, \ldots, c_m\}$ is a set of soft *constraints*, where each constraint is a function $c_i : D_{i_1} \times \ldots \times D_{i_l} \rightarrow \mathbb{R}$ that assigns a *utility* $c_i(x_{i_1}, \ldots, x_{i_l})$ to combinations of assignments to a subset of decision variables.

A *solution* to the DCOP is an assignment to all decision variables that maximizes the sum of all utilities $\sum_i c_i$. Since all constraints are soft constraints defined as utility functions, they will interchangeably be denoted by the letter u. Hard constraints can be modeled as soft constraints using virtually infinite negative utilities. In the following, it is assumed that any constraint/utility function is known to all variables in its scope. It is also assumed for simplicity and without loss of generality that each agent only owns a single variable, and Agent $a(x)$ will sometimes be referred to as "Agent x."

2.2 The Truck Task Coordination Problem

For the sake of simplicity, assume that each company owns only one truck. A simple instance of this problem is illustrated in Fig. 1. Notice that this problem consists of two subproblems: 1) given a set of packages to be picked up and delivered by a specific truck, compute the cost of the optimal pickup and delivery plan; and 2) given these costs, collaboratively compute an optimal allocation of packages to trucks. The first subproblem is internal to each truck and independent of any other truck's subproblem; it hence does not require a collaborative solution process. In the rest of this paper, we abstract away this subproblem by assuming that all costs are known beforehand, and we focus on solving the second optimal allocation problem, using a distributed optimization algorithm.

Following the DCOP formalism from Section 2.1, we model the problem as follows. Each truck is assigned a decision variable, whose domain is the set of all subsets of packages that are located in its pickup area. For instance, in Fig. 1, the domain of Truck w's decision variable (hereafter simply denoted w) is the powerset $\mathcal{P}(P_D \cup P_E)$.

The constraints in the DCOP are twofold:

- One unary (soft) constraint for each truck variable, which defines the value of each assignment of packages to that truck, as the sum of the values of all packages in the assigment minus the cost of the corresponding optimal pickup and delivery plan.
- One binary (hard) constraint for each pair of trucks whose areas overlap, which imposes that packages located in the intersection of the two pickup areas can only be assigned to at most one of the two trucks. For instance, in Fig. 1, there exists such a constraint between decision variables w and y, imposing that none of the packages in P_E can be delivered by both trucks.

2.3 Original DPOP Algorithm

The DPOP algorithm [5] is an instance of the general bucket elimination scheme from [10], which is adapted for the distributed case. DPOP has 3 phases:

Phase 1 – DFS Pseudotree Generation: Variables are first arranged following a pseudotree by running a leader election algorithm in order to choose a root variable, followed by a DFS traversal of the constraint graph. As a result, each node consistently labels its neighbours as parent/child or pseudoparent/pseudochild. The DFS pseudotree serves as a communication structure for the other two phases of the algorithm: UTIL messages (Phase 2) travel bottom-up, and VALUE messages (Phase 3) travel top-down, only via tree-edges.

Phase 2 – UTIL Propagation: The agents (starting from the leaves) send UTIL messages to their parents, containing the optimal, aggregated utility of their subtrees, as a function of other variables higher up. To do so, each agent *joins* (i.e. sums) received UTIL messages with its own utility function. It then *projects out* its variable by optimizing over it. This process is illustrated in Fig. 2a.

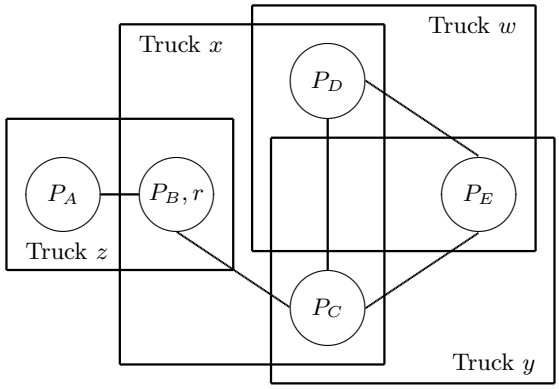

Fig. 1. A Truck Task Coordination problem. Circles are cities, with lists of packages P_X waiting for pickup, and rectangles are pickup areas for the various trucks.

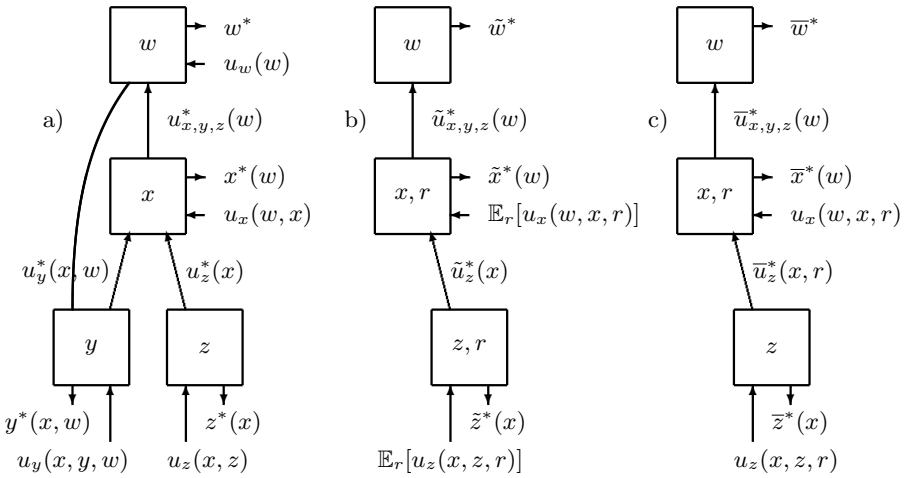

Fig. 2. A possible DFS tree for the truck variables corresponding to Fig. 1: a) for DPOP; b) for the Simple Algorithm and Local-𝔼[DPOP] (which only differ by the fact that the latter has x and z use the same sample set, chosen by $x = lca(r)$, to approximate $\mathbb{E}_r[\cdot]$), showing only differences with a) for clarity; and c) for Global-𝔼[DPOP], showing only differences with b). The presence of an r beside a decision variable indicates that the variable is responsible for projecting out random variable r.

Phase 3 – VALUE Propagation: This top-down propagation is initiated by the root, when Phase 2 has finished. Each node determines its optimal value from the computation in Phase 2 and the VALUE message received from its parent. It then sends this value to its children via VALUE messages.

3 DCOP under Stochastic Uncertainty

The following section first introduces an extension of the DCOP formalism from Section 2.1 that is able to model stochastic uncertainty. Section 3.2 then provides an illustrative, stochastic version of the example in Section 2.2.

3.1 StochDCOP Formalism

A *Distributed Constraint Optimization Problem under Stochastic Uncertainty* (*StochDCOP*) is defined as a tuple $< \mathcal{A}, \mathcal{X}, \mathcal{D}, \mathcal{R}, \Delta, \mathcal{P}, \mathcal{C} >$:

- $\mathcal{A}$, $\mathcal{X}$ and $\mathcal{D}$ are defined as in Section 2.1;
- $\mathcal{R} = \{r_1, \ldots, r_q\}$ is a set of *random variables*;
- $\Delta = \{\Delta_1, \ldots, \Delta_q\}$ is a set of (not necessarily finite) domains for the random variables such that r_i takes values in Δ_i;
- $\mathcal{P} = \{\pi_1, \ldots, \pi_q\}$ is a set of *probability distributions*, where each distribution $\pi_i : \Delta_i \rightarrow \mathbb{R}$ defines the probability law for random variable r_i;
- $\mathcal{C} = \{c_1, \ldots, c_m\}$ is a set of soft constraints over mixed subsets of decision and random variables. Any such constraint is known to all agents controlling a decision variable in its scope, and is assumed to involve at least one decision variable (constraints involving only random variables can be modeled as probability distributions).

A *solution* is an assignment of values to all decision variables that maximizes the *expected* sum of all utilities $\mathbb{E}_{\mathcal{R}} \left[\sum_i c_i\right]$. Notice that random variables are not initially assigned to any specific agent; this is because random variables model uncertainty in the agents' common environment.

This definition assumes without loss of generality that all random variables are *independently distributed* (but not necessarily from the same distribution), i.e. all probability distributions are unary. A StochDCOP involving joint probability distributions can be reduced to match the definition by combining dependent random variables into one. Notice also that probability distributions do not depend on decision variables; in other words, only *exogenous uncertainty* is considered. Finally, it is assumed that a sampling procedure for random variable r is available to each agent constrained with r, and that the agents either have different sampling procedures, or the procedure is non-deterministic, such that if two agents independently call their respective procedures, they will produce two different sets of weighted samples.

3.2 Stochastic Truck Task Coordination

In the stochastic version of the Truck Task Coordination problem defined in Section 2.2, random variables are used to model the possibility for new packages to be announced. More formally, each random variable models a potential package to appear in a given city. Its domain contains possible values for the package's attributes, such as its destination city and its value. For instance, in Fig. 1, the random variable r represents a hypothetical package to be picked up in city B.

With respect to the previous DCOP formulation, there is now a new, third type of constraints in the problem:

– One binary (soft) constraint between any random variable and any truck variable whose area contains the random variable, defining the value obtained by the truck if it delivers the corresponding hypothetical package.

4 The 𝔼[DPOP] Algorithms for StochDCOP

Recall from Section 3.1 that random variables' domains may be infinite. In order to be able to apply standard DCOP techniques, each random variable's domain is discretized by sampling the probability distribution of that variable. This section presents several StochDCOP algorithms that differ in the way sampling is performed, and how samples are used to choose a solution to the problem.

4.1 Simple Algorithm

A simple approach to solve a StochDCOP problem is for each agent to independently sample the random variables it depends on in order to approximate its local, expected utility, without collaborating with other agents whose utilities might depend on the same random variables (Table 1). The agents then run DPOP on the expected utilities, as illustrated in Fig. 2b.

Table 1. Comparison between algorithms based on where they respectively choose sample sets and project random variables.

		sampling		
		at the leaves	at the *lca*s	at the root
projection	at the leaves	Simple Algorithm	Local-𝔼[DPOP]	
	at the *lca*s		Global-𝔼[DPOP]	
	at the root			Central-𝔼[DPOP]

The advantage of this approach is that it introduces no additional complexity compared to DPOP in terms of the number of messages and their sizes. Its main

drawback however is that, since agents independently sample random variables, they reason on inconsistent samples, which we show in Section 5 can lead to a loss in expected solution quality. To address this issue, the following section shows how agents can collaborate to reason on common samples.

4.2 Collaborative Sampling and Local-$\mathbb{E}$[DPOP]

Designing a distributed algorithm allowing agents to agree on samples for a random variable r they share a common interest for is not a trivial task. The challenge is that, in the general case, two agents with a constraint over r might not be able to communicate directly, because they are not necessarily neighbors in the constraint graph. They might not even know that each other has a constraint over r and that they need to agree on the same samples for r.

This section introduces a new distributed algorithm called *collaborative sampling*, following which, for each random variable r, agents independently propose sample sets for r, and elect one agent among themselves as responsible for combining the proposed sample sets into one. A natural choice for such an agent could be the root of the DFS; this would however be a poor choice as it would put a high computational burden on this agent, which would then be a bottleneck responsible for choosing all samples. Furthermore, it would then have to communicate the samples to all other interested agents, using messages that would potentially have to traverse the whole DFS to reach their recipients.

We propose to elect, for each random variable r, the *lowest common ancestor* $lca(r)$ in the DFS of all variables constrained with r. This choice minimizes the communication cost, as $lca(r)$ corresponds to the agent that is the closest (in terms of number of tree edges) to all variables constrained with r. It also has the advantage of distributing the computational load among the agents.

Detailed Algorithm Algorithm 1 shows how agents compute *lca*s and agree on common sets of samples. It proceeds in two phases: during the first, bottom-up phase, messages containing random variables and proposed sample sets are propagated up the DFS pseudotree along tree-edges starting from the leaves, until the root knows the set of all random variables in the problem. During the second, top-down phase, messages back-propagate down the pseudotree along tree-edges, after which each agent knows whether it is the *lca* of some random variable(s). Sample sets are also chosen by re-sampling the proposed sets (line 19), and forwarded down the tree during this second phase.

We refer to the modification of the Simple Algorithm using collaborative sampling as *Local-*$\mathbb{E}$[*DPOP*], because each agent locally computes its expected utility (Table 1).

Complexity Analysis The first phase in Algorithm 1 generates $(n-1)$ messages of sizes $O(nqk)$, where n and q are respectively the numbers of decision and random variables, and k is the number of samples for each random variable. In the worst case of a single-branch DFS in which root and leaf are constrained

Algorithm 1 Collaborative sampling

Each agent $a(x)$ does the following:

1: // **Bottom-up phase**
2: $R_0 \leftarrow \{r \in \mathcal{R} \mid x \text{ is constrained with } r\}$
3: $\mathcal{S}_0 \leftarrow \{sample(\pi_r) \mid r \in R_0\}$
4: **wait** for all $\{R_i, \mathcal{S}_i\}_{i=1\ldots t}$ from all children
5: $R_x \leftarrow \cup_{i \geq 0} R_i$
6: $\mathcal{S}_x \leftarrow \cup_{i \geq 0} \mathcal{S}_i$
7: **send** $\{R_x, \mathcal{S}_x\}$ to parent (if any)
8: **if** $R_x = \emptyset$ **then**
9: **execute** UTIL phase

10: // **Top-down phase**
11: $\mathcal{S} \leftarrow \emptyset$ // all known sample sets
12: $R_x \leftarrow \cup_{1 \leq i < j} (R_i \cap R_j) \cup R_0$
13: **if** $a(x)$ has a parent p **then**
14: **wait** for $\{R_p, \mathcal{S}_p\}$ from parent
15: $\mathcal{S} \leftarrow \mathcal{S}_p$
16: $R_x \leftarrow R_x \cap R_p$
17: **for all** $r \in R_x$ **do**
18: $lca(r) \leftarrow a(x)$
19: $S_r \leftarrow sample(\cup_{S_r \in \mathcal{S}_x} S_r)$
20: $\mathcal{S} \leftarrow \mathcal{S} \cup \{S_r\}$
21: **for all** children $i = 1 \ldots t$ such that $R_i \neq \emptyset$ **do**
22: **send** $\{(R_i \cap R_p) - R_x, \{S_r \in \mathcal{S} \mid r \in R_i\}\}$ to child i
23: **execute** UTIL phase

with all random variables, the second phase generates $(n-1)$ messages of sizes $O(qk)$. Overall, Algorithm 1 therefore only generates a number of messages that is linear in the number of decision variables, and their sizes are linear in the total number of proposed samples across all variables.

It has been proven in [5] that DPOP produces a number of messages that is also linear in the number of decision variables. Its complexity lies in the size of the UTIL messages: the largest one is space-exponential in the induced width of the DFS used. The complexity of Algorithm 1 is hence negligible compared to that of the UTIL phase, and the overall complexity of Local-𝔼[DPOP] is therefore sensibly the same as that of DPOP.

4.3 The Global-𝔼[DPOP] Algorithm

As mentioned in Section 4.1, the main drawback of the Simple Algorithm is that it completely ignores the fact that several agents' utilities can depend on the same random variable. Local-𝔼[DPOP] provides a partial patch for this, by enforcing that all agents agree on using the same sample sets. However, agents still do not collaborate by exchanging information about how their utilities depend on the random variables.

The *Global-*$\mathbb{E}$*[DPOP]* algorithm is a modification of Local-$\mathbb{E}$[DPOP] that retains information about dependencies on random variables in the UTIL messages. Each agent x proceeds as follows: when it has received all UTIL messages from its children $u_1(x,\cdot),\ldots,u_n(x,\cdot)$, it joins them with its local utility $u_x(x,r_1,\ldots,r_m,\cdot)$, computes the expected joint utility:

$$\overline{u}_{x,1,\ldots,n}(x,\cdot) = \mathbb{E}_{R_{x,1,\ldots,n}}\left[u_x(x,r_1,\ldots,r_m,\cdot) + \sum_{i=1\ldots n} u_i(x,\cdot)\right],$$

where $R_{x,1,\ldots,n}$ is the set of random variables in the joint utility, and then projects out its variable by choosing the assignment to x that maximizes this expected utility: $\overline{x}^*(\cdot) = \arg\max_x\{\overline{u}_{x,1,\ldots,n}(x,\cdot)\}$. Like in Local-$\mathbb{E}$[DPOP], the chosen assignment is therefore *independent of any random variable*: the agent must make decisions without knowing the future. Finally, unlike Local-$\mathbb{E}$[DPOP], which would simply report $\tilde{u}^*_{x,1,\ldots,n}(\cdot) = \max_x\{\overline{u}_{x,1,\ldots,n}(x,\cdot)\}$, *the agent reports to its parent the effective utility achieved by its chosen assignment, as a function of the random variables*: $\overline{u}^*_{x,1,\ldots,n}(\cdot) = u_x(\overline{x}^*(\cdot),r_1,\ldots,r_m,\cdot) + \sum_{i=1\ldots n} u_i(\overline{x}^*(\cdot),\cdot)$.

Information about the dependency on all random variables could be propagated all the way to the root; this approach is referred to as Central-$\mathbb{E}$[DPOP] in Table 1, and corresponds to centralizing at the root the uncertain part of the DCOP. However this would result in an uncontrolled explosion in message sizes. It is hence important to remove all information about any given random variable r as soon as it is no longer necessary, i.e. no more agents higher in the DFS depend on r. This condition is given as soon as the messages reach $lca(r)$; therefore Global-$\mathbb{E}$[DPOP] has $lca(r)$ project out r once and for all by reporting the expected utility with respect to r, as Local-$\mathbb{E}$[DPOP] would (Table 1).

This algorithm is illustrated in Fig. 2c. Agent z first computes the expected utility $\mathbb{E}_r[u_z(x,z,r)]$, and infers the optimal assignment to its variable as a function of x:

$$\overline{z}^*(x) = \arg\max_z\{\mathbb{E}_r[u_z(x,z,r)]\}. \tag{1}$$

It then sends to its parent $\overline{u}^*_z(x,r) = u_z(x,\overline{z}^*(x),r)$. Agent x then joins this with its local utility $u_x(w,x,r)$ and with the utility $u^*_y(x,w)$ received from agent y, to obtain the aggregated utility $\overline{u}_{x,y,z}(w,x,r)$. It then computes the optimal assignment to its variable: $\overline{x}^*(w) = \arg\max_x\{\mathbb{E}_r[\overline{u}_{x,y,z}(w,x,r)]\}$. Finally, because it is $lca(r)$, agent x projects out r by reporting directly $\overline{u}^*_{x,y,z}(w) = \max_x\{\mathbb{E}_r[\overline{u}_{x,y,z}(w,x,r)]\}$.

Complexity Analysis Consider the complexity of the UTIL phase in both algorithms. Local-$\mathbb{E}$[DPOP] produces $(n-1)$ messages, each of size $O(d^w)$, where d is the size of the largest decision variable domain and w is the induced width of the DFS pseudotree. The UTIL messages sent by Global-$\mathbb{E}$[DPOP] are the same as Local-$\mathbb{E}$[DPOP], except that they can carry additional random variables, whose sampled domains are of size k. In the worst case where the root as well as all leaves are constrained with all q random variables, the size of Global-$\mathbb{E}$[DPOP]'s largest UTIL message is $O(d^w k^q)$.

4.4 Equivalence Result

The three 𝔼[DPOP] algorithms presented in the previous sections all share the common property that all agents performing operations involving any random variable r use the same sample set for r, chosen by $lca(r)$. This property differentiates the 𝔼[DPOP] algorithms from the Simple Algorithm, in which each agent independently chooses its sample sets. As illustrated in Table 1, the only difference between these three algorithms is where the random variables are projected out: Local-𝔼[DPOP] projects them at the leaves (i.e. each agent projects out all known random variables before running the classical DPOP algorithm), Global-𝔼[DPOP] at the *lca*s, and Central-𝔼[DPOP] at the root. The following theorem establishes that the choice of where to project is actually irrelevant, and that the three algorithms are equivalent, i.e. they produce the same solutions.

Theorem 1. *The Local-𝔼[DPOP], Global-𝔼[DPOP] and Central-𝔼[DPOP] algorithms produce the same solutions.*

Proof. We only prove that Local-𝔼[DPOP] and Global-𝔼[DPOP] are equivalent; the proof easily extends to Central-𝔼[DPOP]. Starting from Global-𝔼[DPOP], we provide a procedure that transforms this algorithm into Local-𝔼[DPOP], without changing the solution chosen to the StochDCOP. The procedure is iterative on the random variables, and given a random variable r, recursive on the depth in the DFS where r is projected out.

Let r be a random variable. Let x be a variable in the DFS that is responsible for projecting out r (in the base case of Global-𝔼[DPOP], x is necessarily $lca(r)$). We now show how the projection of r can be pushed down into x's child variables $y_1, \dots y_n$, without changing x's optimal value $\overline{x}^*$ nor the utility it reports to its parent. Let $u_x(x, r, \cdot)$ be x's local utility function, which actually does not necessarily depend on r, and let $\overline{u}_i^*(x, r, \cdot)$ be the utility reported by y_i in its UTIL message to x (also possibly independent of r). Variable x computes and reports the following optimal aggregated utility:

$$\overline{u}_x^*(\cdot) = \max_x \left\{ \mathbb{E}_r \left[u_x(x, r, \cdot) + \sum_i \overline{u}_i^*(x, r, \cdot) \right] \right\} \tag{2}$$

$$= \max_x \left\{ \mathbb{E}_r \left[u_x(x, r, \cdot) \right] + \sum_i \mathbb{E}_r \left[\overline{u}_i^*(x, r, \cdot) \right] \right\}, \tag{3}$$

where (3) follows from the linearity of the operator $\mathbb{E}_r[\cdot]$. Furthermore:

$$\mathbb{E}_r \left[\overline{u}_i^*(x, r, \cdot) \right] = \mathbb{E}_r \left[u_i(x, \overline{y}_i^*(x, \cdot), r, \cdot) \right] \tag{4}$$

$$= \max_{y_i} \left\{ \mathbb{E}_r \left[u_i(x, y_i, r, \cdot) \right] \right\}, \tag{5}$$

where (5) follows from the definition of $\overline{y}_i^*(x, \cdot)$:

$$\overline{y}_i^*(x, \cdot) = \arg\max_{y_i} \left\{ \mathbb{E}_r \left[u_i(x, y_i, r, \cdot) \right] \right\}.$$

Therefore, we can modify the current algorithm into one that has each y_i project out r and report $\overline{v}_i^*(x,\cdot) = \max_{y_i}\{\mathbb{E}_r[u_i(x,y_i,r,\cdot)]\}$. Notice that this is possible because x and y_i use the same sample set for r, and therefore the same sampled expectation operator $\mathbb{E}_r[\cdot]$. In this modified algorithm, variable x reports:

$$\overline{u}_x^*(\cdot) = \max_x \left\{ \mathbb{E}_r[u_x(x,r,\cdot)] + \sum_i \overline{v}_i^*(x,\cdot) \right\},$$

which is the same utility as in (2), and is the one that Local-$\mathbb{E}$[DPOP] would report. It follows from the same derivation that x also makes the same decision:

$$\begin{aligned}\overline{x}^*(\cdot) &= \arg\max_x \left\{ \mathbb{E}_r \left[u_x(x,r,\cdot) + \sum_i \overline{u}_i^*(x,r,\cdot) \right] \right\} \\ &= \dots \\ &= \arg\max_x \left\{ \mathbb{E}_r[u_x(x,r,\cdot)] + \sum_i \overline{v}_i^*(x,\cdot) \right\},\end{aligned}$$

which is also the decision that Local-$\mathbb{E}$[DPOP] would make. Therefore, without changing the solution to the StochDCOP, we have transformed the initial algorithm in which variable x performs as Global-$\mathbb{E}$[DPOP], into an algorithm in which it performs as Local-$\mathbb{E}$[DPOP], and the projection of r has been pushed down into its children in the DFS. We can now recursively apply the same procedure to each of the children. □

Based on Theorem 1, we refer to Local-$\mathbb{E}$[DPOP] simply as $\mathbb{E}$[DPOP], since Global-$\mathbb{E}$[DPOP] and Central-$\mathbb{E}$[DPOP] produce the same solutions, but have a higher computational complexity (as shown in Sections 4.3 and 5.1).

4.5 The *Consensus* Approach

[9] introduced a new approach for centralized stochastic problems called *consensus*, which differs from the traditional expectation-based approach. They show that when the number of samples is limited, for instance by time constraints, it is better to choose a solution that is the optimal one for the largest number of samples, rather than one that maximizes the expected utility across samples.

We propose a distributed implementation of the consensus method, which differs from the previous algorithms in how each agent chooses the optimal assignments to its variable. For instance, in the example from Fig. 2c, rather than computing Eq. (1), agent z now computes:

$$\overline{z}^*(x) = \arg\max_{z^*} \left\{ \left| \left\{ r \in S_r | z^* = \arg\max_z \{u_z(x,z,r)\} \right\} \right| \right\}. \tag{6}$$

By applying this modification to all previous algorithms, we obtain respectively the *consensus-based Simple Algorithm* (*Cons-SA*), *Cons-Local-*$\mathbb{E}$*[DPOP]*, *Cons-Global-*$\mathbb{E}$*[DPOP]*, and *Cons-Central-*$\mathbb{E}$*[DPOP]*. It is important to note that the tree latter algorithms are all different, since Theorem 1 no longer holds: the proof breaks in particular at Eq. (5). Each consensus-based algorithm has the same complexity in terms of messages as its expectation-based counterpart.

5 Experimental Results

All algorithms were implemented inside the new open-source FRODO 2.0 framework for Distributed Constraint Optimization that has recently been released under the GNU Affero GPL [11]. Random instances of the Truck Task Coordination Problem were generated using a modified version of the generator used in [1]. The instances involve 10 trucks and 20 cities, 15 of which on average fall in the intersection of two truck pickup areas. The trucks have random capacities of between 1 and 5 units of weight. Nine unit-weight packages are used, three of which are considered random packages, with fixed destination cities and random values. All random variables share the same distribution, which assigns a probability of 0.33 to packages of value 0 (i.e. no package appears), and uniform remaining probabilities to 19 other possible values. Values for packages are of the same order of magnitude as the average cost of delivering them. Each datapoint corresponds to the median over 114,107 random instances.

5.1 Maximum Message Size

We first evaluate the influence on maximum UTIL message size (in terms of number of variables) of the choice of the position in the DFS where random variables are projected out. Recall from Table 1 that Local-𝔼[DPOP] projects out random variables at the leaves, while Global-𝔼[DPOP] does it at the *lca*s, and Central-𝔼[DPOP] (Section 4.3) at the root. Table 2 shows that, while Global-𝔼[DPOP] generates larger messages than Local-𝔼[DPOP], the gain incurred by projecting at the *lca*s rather than at the root is significant.

Table 2. Average maximum dimension of UTIL messages. Recall that the space complexity is exponential in this parameter.

Local-𝔼[DPOP]	Global-𝔼[DPOP]	Central-𝔼[DPOP]
2.09	2.905	3.735

5.2 Expected Solution Quality

Fig. 3 shows the loss in expected solution quality (*ESQ*) compared to the *clairvoyant* algorithm that knows in advance the realized values of the random variables. The ESQ of an algorithm is computed by taking its output solution, computing its utility as a function of the random variables, and then taking the expectation, using the non-sampled distributions. Similarly, the clairvoyant ESQ is obtained by computing the optimal solution to the problem as a function of the random variables, and then taking its exact expectation. A loss in ESQ of 0% is therefore never attainable by any *anticipatory* algorithm like the ones in this paper, which do not know the future and output a single solution independent of

the random variables. As a means of comparison, the original DPOP algorithm (ignoring all random variables) was also run on the same problem instances. DPOP provides the baseline of an approach to solve the Truck Task Coordination Problem that does not attempt to seize the opportunity of increasing its utility by reasoning about potential future customer requests.

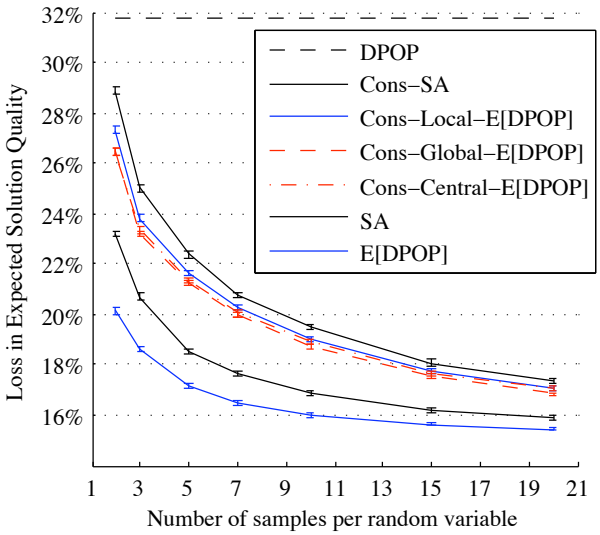

Fig. 3. Median loss in ESQ, with 95% confidence intervals.

The first observation that should be made is that $\mathbb{E}$[DPOP] always outperforms the Simple Algorithm. This demonstrates the superiority of collaborative sampling, compared to the approach in which agents produce samples independently. The same observation can be made for Cons-SA versus Cons-*-$\mathbb{E}$[DPOP].

Second, as pointed out in Section 4.5, we observe that the equivalence result obtained for the various versions of $\mathbb{E}$[DPOP] do not apply to their consensus-based counterparts. While Cons-Global-$\mathbb{E}$[DPOP] and Cons-Central-$\mathbb{E}$[DPOP] do seem to perform empirically the same, Cons-Local-$\mathbb{E}$[DPOP] produces solutions of slightly lower quality when the number of samples is low. The disjoint 95% confidence intervals for up to 7 samples per random variable indicate that this performance loss in Cons-Local-$\mathbb{E}$[DPOP] is statistically significant; for higher numbers of samples, all three algorithms empirically perform the same.

Finally, Fig. 3 shows that all expectation-based algorithms always outperform all consensus-based algorithms. While this might seem to contradict known results in the centralized case [9], we believe that this is an artifact of the simplifying assumption made in Section 2.2 that all entries in the utility tables are pre-computed offline. The results in [9] show that, when computing each entry

is NP-hard (such as for our Truck Task Coordination Problem), and when this computation is performed online, the consensus approach requires to compute fewer of these entries, and can therefore process more samples than the expectation approach under time constraints, hence producing higher-quality solutions. Future work will aim to remove this simplifying assumption.

6 Conclusion

This paper introduces a new formalism for DCOP problems under stochastic uncertainty. The main contribution is the *collaborative sampling* method, which underlies the 𝔼[*DPOP*] algorithms we propose to solve such problems, including distributed implementations of the *consensus* algorithm. We provide a performance analysis that demonstrates the benefit of reasoning about uncertainty, and that using collaborative sampling produces even higher-quality solutions.

We also compare various methods of reasoning about the sources of uncertainty, and investigate the effects of information sharing between agents on the tradeoff between the expected solution quality and the complexity in terms of message sizes. We show in particular that propagating this information only locally rather than to all agents significantly lowers the complexity, while producing solutions of comparable qualities.

References

1. Ottens, B., Faltings, B.: Coordinating agent plans through distributed constraint optimization. In: MASPLAN'08. (2008)
2. Hirayama, K., Yokoo, M.: Distributed partial constraint satisfaction problem. In: Proceedings of CP'97. (1997) 222–236
3. Modi, P.J., Shen, W.M., Tambe, M., Yokoo, M.: ADOPT: Asynchronous distributed constraint optimization with quality guarantees. Artificial Intelligence **161** (2005) 149–180
4. Mailler, R., Lesser, V.R.: Solving distributed constraint optimization problems using cooperative mediation. In: AAMAS'04. Volume 1. (2004) 438–445
5. Petcu, A., Faltings, B.: DPOP: A Scalable Method for Multiagent Constraint Optimization. In: IJCAI'05. (2005) 266–271
6. Gershman, A., Meisels, A., Zivan, R.: Asynchronous forward-bounding for distributed constraints optimization. In: Proceedings of ECAI'06. (2006) 103–107
7. Chechetka, A., Sycara, K.: No-commitment branch and bound search for distributed constraint optimization. In: AAMAS'06. (2006) 1427—1429
8. Atlas, J., Decker, K.: Task scheduling using constraint optimization with uncertainty. In: Proceedings of CAPS'07. (2007) 25–27
9. Bent, R., Hentenryck, P.V.: The value of consensus in online stochastic scheduling. In: ICAPS'04. (2004) 219–226
10. Dechter, R.: Constraint Processing. Morgan Kaufmann (May 5 2003)
11. Léauté, T., Ottens, B., Szymanek, R.: FRODO 2.0: An open-source framework for distributed constraint optimization. In: Proceedings of the IJCAI'09 DCR Workshop. (2009) `http://liawww.epfl.ch/frodo/`.