Distributed Constraint Optimization under Stochastic Uncertainty

Thomas Léauté and **Boi Faltings**
École Polytechnique Fédérale de Lausanne (EPFL)
Artificial Intelligence Laboratory (LIA)
firstname.lastname@epfl.ch

Abstract

In many real-life optimization problems involving multiple agents, the rewards are not necessarily known exactly in advance, but rather depend on sources of exogenous uncertainty. For instance, delivery companies might have to coordinate to choose who should serve which foreseen customer, under uncertainty in the locations of the customers. The framework of Distributed Constraint Optimization under Stochastic Uncertainty was proposed to model such problems; in this paper, we generalize this formalism by introducing the concept of *evaluation functions* that model various optimization criteria. We take the example of three such evaluation functions, *expectation*, *consensus*, and *robustness*, and we adapt and generalize two previous algorithms accordingly. Our experimental results on a class of Vehicle Routing Problems show that incomplete algorithms are not only cheaper than complete ones (in terms of *simulated time*, *Non-Concurrent Constraint Checks*, and *information exchange*), but they are also often able to find the optimal solution. We also show that exchanging more information about the dependencies of their respective cost functions on the sources of uncertainty can help the agents discover higher-quality solutions.

1 Introduction

Consider a set of delivery companies, which must coordinate to decide how to serve a set of customers, so as to minimize the total distance travelled. In real life, it is often the case that not all information about customers is known exactly, at the time when the allocation decisions must be made, and the problem is one of multi-agent optimization under uncertainty. Other examples include meeting scheduling with uncertain meeting durations, or sensor network configuration under uncertainty in target positions. Uncertainty can take many forms: for instance, the customers positions might be known beforehand, but their exact demands are not known a priori, and can only be forecast with some probability, based on historical data. In this paper, we focus particularly on sources of uncertainty that have competing effects on the agents. If one customer's demand is known, but her exact location is uncertain, then the optimal delivery company for her depends on her distance from the respective depots, and finding such an optimal allocation usually requires coordinated reasoning about uncertainty by the companies.

We propose the formalism of *Distributed Constraint Optimization under Stochastic Uncertainty (StochDCOP)* to model such multi-agent optimization problems. This paper is a significant improvement over our preliminary workshop paper (Léauté and Faltings 2009); the novel contributions are threefold. First, we enrich the StochDCOP formalism with the concept of *evaluation functions* (Section 2), which characterize the agents' approach to handling uncertainty. Second, we adapt and generalize previous distributed algorithms accordingly (Sections 3 and 4). Finally, we empirically compare the resulting algorithms on Vehicle Routing Problem benchmarks (Section 5), and we show in particular that incomplete algorithms can outperform complete ones, while still finding the optimal solution in most cases.

2 DCOP under Uncertainty

Section 2.1 first defines the DCOP under uncertainty, illustrated on a vehicle routing problem in Section 2.2.

2.1 Problem Definition

We first recall the definition of *Distributed Constraint Optimization (DCOP)*, of which StochDCOP is an extension.

Definition 1 (DCOP) *A DCOP is 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 controlled 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*[1] constraints *that assign* costs $\in \mathbb{R}$ *to assignments to subsets of decision variables. Each* c_i *is assumed to be known to all agents involved.*

A solution *is an assignment of values to all decision variables that minimizes the sum of all constraints* $\sum_i c_i$*.*

A DCOP is distributed in nature, in two respects:

1. The *decision power* is distributed: each agent is in charge of choosing values for the decision variables it controls;
2. The *knowledge* is distributed: each agent only knows the constraints over the decision variables it controls, and knows all the variables involved in these constraints.

[1] *Hard* constraints can be modeled as soft constraints with sufficiently large costs.

DCOP under Stochastic Uncertainty (StochDCOP) is an extension of DCOP that includes sources of uncertainty, which are modeled by random, uncontrollable variables.

Definition 2 (StochDCOP) *A StochDCOP is defined as a tuple* $< \mathcal{A}, \mathcal{X}, \mathcal{D}, \mathcal{R}, \Delta, \mathcal{P}, \mathcal{C}, e >$*, where:*

- $\mathcal{A}$*,* $\mathcal{X}$ *and* $\mathcal{D}$ *are defined as in Definition 1;*
- $\mathcal{R} = \{r_1, \ldots, r_q\}$ *is a set of* random variables *modeling future, uncontrollable events;*
- $\Delta = \{\Delta_1, \ldots, \Delta_q\}$ *is a set of finite domains for the random variables;*
- $\mathcal{P} = \{\pi_1, \ldots, \pi_q\}$ *is a set of* probability distributions *for the random variables, such that* $\pi_i : \Delta_i \rightarrow [0, 1]$ *and the values of* π_i *sum up to* 1*;*
- $\mathcal{C} = \{c_1, \ldots, c_m\}$ *is a set of soft* constraints *over mixed subsets of decision and random variables;*
- e *is an* evaluation function*, which, given a constraint* $c_i(x_{i_1}, \ldots, x_{i_l}, r_{i'_1}, \ldots, r_{i'_k})$*, associates, to each assignment of values to* c_i*'s decision variables, a single cost* $e_{c_i}(x_{i_1}, \ldots, x_{i_l}) \in \mathbb{R}$ *that is independent of the* $r_{i'}$*'s.*

A solution *is an assignment of values to all decision variables that is independent of the random variables, such that:*

$$(x_1^*, \ldots x_n^*) = \arg \min_{x_1 \ldots x_n} \left\{ e_{\sum_i c_i}(x_1, \ldots x_n) \right\} .$$

StochDCOPs are still distributed in nature:

1. The decision power is distributed like in DCOPs, but random variables are not under the control of the agents: their values are drawn by Nature from their respective probability distributions, independently of the decision variables, and after the agents must have chosen values for their decision variables. Solving a StochDCOP involves choosing values for the decision variables *only*, anticipating Nature's future decisions.
2. The knowledge of each random variable, its domain and its probability distribution is shared only among the agents in control of neighboring decision variables.

The evaluation function is used to summarize in one criterion the cost of a given assignment to the decision variables, which would normally depend on the random variables. An example is the *expectation* function: $e_{\sum_i c_i} = \mathbb{E}_{\mathcal{R}}[\sum_i c_i]$. Other evaluation functions are proposed in Section 2.3.

Following Definition 2, multiple agents' costs can depend on a common random variable. The algorithms discussed in this paper differ in how agents reason about such co-dependencies. Furthermore, the probability distributions are assumed to be independent of each other and of the decision variables, i.e. we assume *independent* sources of *exogenous* uncertainty. Two sources of uncertainty with correlated probability distributions can be represented by a single random variable in order to enforce independence.

2.2 Vehicle Routing Example

This section defines a problem class that is a variant of the well-known *Vehicle Routing Problem (VRP)*, which we use to illustrate our algorithms on the instance in Figure 1, and to evaluate them on problem instances that have a structure

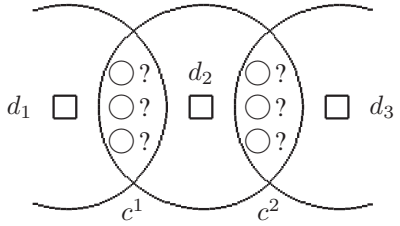

Figure 1: A StochDisSDMDVRP instance involving two customers c^1 and c^2 with each three possible uncertain locations, and three depots such that depot d_1 can only serve c^1, d_2 can serve both customers, and d_3 only c^2.

inspired by real-life problems (Section 5). It extends the *DisMDVRP* variant in (Léauté, Ottens, and Faltings 2010) in two respects: 1) customers have uncertain locations, and 2) a customer's demand can be split among multiple depots.

Definition 3 (StochDisSDMDVRP) *The* Stochastic, Distributed, Shared-Deliveries, Multiple-Depot VRP *is defined as a tuple* $< D, n_V, Q_{\max}, L_{\max}, H, C, R, Q >$*, where:*

- $D = \{d_1, \ldots, d_{n_D}\}$ *is a set of depots in the Euclidian plane, each controlled by a different delivery company;*
- n_V *is the number of vehicles at each depot;*
- $Q_{\max}$ *and* $L_{\max}$ *are respectively the maximum load and maximum route length of any vehicle;*
- $H \leq L_{\max}/2$ *is the visibility horizon of each company, which defines the boundaries of its knowledge of the overall problem. Depot* d_i *is only aware of and can only serve the customers that are within distance* H*;*
- $C = \{c^1, \ldots, c^{n_C}\}$ *is a set of customers;*
- $R = \{r_1, \ldots, r_{n_C}\}$ *are the uncertain locations for the customers, with known probability distributions;*
- $Q = \{q_1, \ldots, q_{n_C}\} \in \mathbb{N}^{n_C}$ *are the customer demands, which can be split among multiple companies.*

The goal is for the companies to agree on who should serve which customers, using which vehicle routes, so as to fully serve all visible customers, at minimal total route length.

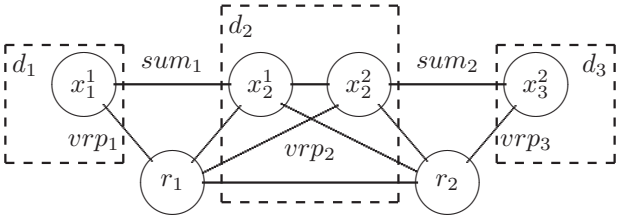

Figure 2: The StochDCOP constraint graph for Figure 1.

We propose the following StochDCOP formulation, in which each depot d_i is an agent, controlling a variable $x_i^j \in [0, q_j]$ for each visible customer c^j, modeling how much of c^j's demand it serves (Figure 2). One random variable r_j is also introduced for each customer c^j that corresponds to her uncertain location. The constraints are of two types:

1. For each customer c^j, if the set D^j of H-away depots is $\neq \emptyset$, then a hard constraint $sum_1 : \sum_{d_i \in D^j} x_i^j = q_j$ enforces that c^j's demand for q_j goods must be fully served.
2. For each depot d_i, if the set $C_i = \{c^{i_1}, \ldots, c^{i_n}\}$ of visible customers is $\neq \emptyset$, then the following soft constraint represents the cost of the optimal solution to d_i's own VRP:

$$vrp_i(x_i^{i_1}, \ldots, x_i^{i_n}, r_{i_1}, \ldots, r_{i_n}) \in [0, \infty] \; . \qquad (1)$$

Checking vrp_i against a given variable assignment is expensive, since it involves solving an NP-hard VRP.

2.3 Evaluation Functions

As presented in Definition 2, a solution to a StochDCOP is an assignment $(x_1^*, \ldots, x_n^*)$ that is *independent* of the random variables r_i. Its cost therefore depends on the r_i's, and is equal to $\sum_i c_i(x_1^*, \ldots, x_n^*, r_1, \ldots, r_q)$. In order to evaluate the quality of a solution using a single criterion, the StochDCOP defines an *evaluation function* e, such that the optimal solution must minimize $e_{\sum_i c_i}(x_1^*, \ldots, x_n^*) \in \mathbb{R}$.

Consider a constraint $c(x, r)$; we formalize three ways to evaluate the cost of an assignment to x: the *expectation* approach, the *consensus* approach, and the *robust* approach.

Expected Solution Quality The *expectation* function returns the *expected* value of $c(x, r)$, based on π_r:

$$e_c : x \to \mathbb{E}_r \left[c(x, r)\right] = \int_{r=-\infty}^{\infty} \pi_r(r) c(x, r) dr \; .$$

When the domain Δ_r of the random variable r is finite, the expectation can be computed as follows:

$$\mathbb{E}_r \left[c(x, r)\right] = \sum_{r_i \in \Delta_r} \pi_r(r_i) c(x, r_i) \; . \qquad (2)$$

The prefix *Exp-* will denote this approach.

***Consensus*: Maximum Probability of Optimality** Bent and Van Hentenryck (2004) introduced a new approach for centralized stochastic problems called *consensus*. They showed that when the number of scenarios used to approximate the probability distributions is limited, it is better to choose a solution that is the optimal one for the largest number of scenarios, rather than one that minimizes the expected cost across scenarios. This amounts to maximizing the probability of optimality, which we denote by $\mathbb{P}$, rather than the expected solution quality, and corresponds to choosing the evaluation function $e_c : x \to -\mathbb{P}_r[c(x, r)]$ that returns (minus) the probability that $c(x, r)$ is minimized:

$$\mathbb{P}_r[c(x, r)] = \int_{r=-\infty}^{\infty} \pi_r(r) \cdot \delta_{x = \arg\min_{x'} c(x', r)}(x, r) dr$$

$$\mathbb{P}_r[c(x, r)] = \sum_{r_i \in \Delta_r} \pi_r(r_i) \cdot \delta_{x = \arg\min_{x'} c(x', r_i)}(x, r_i) \quad (3)$$

where $\delta_X = 1$ if X is true, 0 otherwise. This approach will be referred to using the prefix *Cons-*.

***Robust* Decision-making: Worst-case Solution Quality** In some problem domains, agents might not want to choose a solution that can have a very high cost with some non-zero probability, even if this solution gives minimum expected cost. In such cases, it can be more desirable to choose the solution that minimizes the *worst-case* cost instead:

$$e_c : x \to \max_r \{c(x, r)\} \; .$$

The prefix *Rob-* will be used for this approach.

3 Related Work

Matsui et al. (2010) introduced the formalism of *Quantified DCOP (QDCOP)*, which is a *multi-stage* StochDCOP, except that they only considered the *robust* evaluation function. They showed that QDCOP algorithms can be obtained from existing DCOP algorithms, such as ADOPT (Modi et al. 2005), by assigning random variables to virtual, adversarial agents that perform the "opposite" DCOP algorithm, by *maximizing* the overall cost. In this paper, we propose to adapt this approach to solve StochDCOPs using the DPOP algorithm (Petcu and Faltings 2005), and we generalize it to the *expectation* and *consensus* evaluation functions.

One particularity of this approach is that it imposes that the variable ordering used in the algorithm must be *consistent*. Both ADOPT and DPOP use a *pseudo-tree* variable ordering, i.e. a tree in which *back-edges* with remote ancestors are allowed. *Consistency* imposes that decision variables must be at the top, and random variables at the bottom. In the example in Figure 3, if the task of simulating r_1 and r_2 is assigned to depot d_3 (which makes most sense in terms of message exchange since d_3 already controls variable x_3^2), then all three VRP constraints are centralized at d_3, which creates an important computational bottleneck.

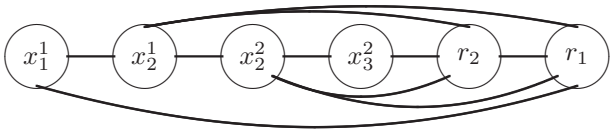

Figure 3: A consistent pseudo-tree for the constraint graph in Figure 2 (sideways to save space, with x_1^1 at the root).

This approach will be referred to with the *Comp-* prefix, because the use of consistent pseudo-trees guarantees *completeness*. The algorithms described in the following section do not impose consistency.

4 Incomplete StochDCOP Algorithms

This section proposes a family of StochDCOP algorithms, called $\mathbb{E}$[DPOP], based on DPOP (Petcu and Faltings 2005). It is worth pointing out that our techniques are not specific to DPOP, and could be applied to other DCOP algorithms.

4.1 General Approach

In contrast with Section 3, $\mathbb{E}$[DPOP] builds pseudo-trees that initially only involve decision variables; random variables

are inserted afterwards. The resulting pseudo-trees may violate the consistency rule that decision variables should be above random variables (Figure 4). Notice that the pseudo-tree that has lower depth and width than the *Comp-* approach (Figure 3), hereby improving performance (Section 5).

$\mathbb{E}$[DPOP] builds upon the general bucket elimination scheme by Dechter (2003), and proceeds in four phases:

1. *DFS pseudo-tree generation*: agents generate a pseudo-tree arrangement of the *decision* variables *only*;
2. *Assignment of random variables to decision variables*: random variables, which are not initially controlled by any agent, are associated to decision variables;
3. *UTIL propagation*: costs are aggregated up the pseudo-tree, *projecting out* variables along the way, until the root variable knows the optimal evaluated cost for the whole problem as a function of its possible values only;
4. *VALUE propagation*: optimal assignments to decision variables are propagated down the pseudo-tree.

We propose $\mathbb{E}$[DPOP] variants that differ on where each random variable r is inserted in the pseudo-tree during Phase 2. In *Local-*$\mathbb{E}$*[DPOP]*, r is assigned to each decision variable responsible for enforcing a constraint involving r. In Figure 4a), r_1 is involved in the two constraints vrp_1 and vrp_2 (Figure 2), enforced respectively by x_1^1 and x_2^2. During Phase 3, each decision variable x computes its optimal assignment x^* (as a function of higher decision variables) by minimizing the evaluation function, and reports to its parent the corresponding evaluated optimal cost c^*_x. Neither therefore depends on *any* random variable. This is shown for variable x_3^2 in Figure 4a), where:

$$x_3^{2^*}(x_2^2) = \arg\min_{x_3^2} \left\{ e_{vrp_3+sum_2}(x_3^2, x_2^2) \right\} \quad (4)$$

$$c^*_{x_3^2}(x_2^2) = e_{vrp_3+sum_2}(x_3^{2^*}(x_2^2), x_2^2)\ . \quad (5)$$

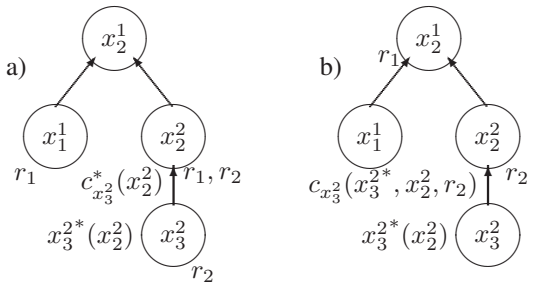

Figure 4: An (inconsistent) pseudo-tree for the constraint graph in Figure 2. The cost reported by x_3^2 to x_2^2 is also shown, a) for Local-$\mathbb{E}$[DPOP], and b) for Global-$\mathbb{E}$[DPOP].

In contrast, *Global-*$\mathbb{E}$*[DPOP]* assigns r to the lowest common ancestor $lca(r)$ of all decision variables enforcing a constraint over r. In Figure 4b), r_1 is no longer assigned to x_1^1 and x_2^2, but to their *lca* x_2^1. The optimal assignment to each decision variable x is computed as in Eq. (4), but x now reports its *effective* cost, *as a function of the random variables* whose lca is above. For instance, x_3^2 now reports:

$$c_{x_3^2}(x_3^{2^*}, x_2^2, r_2) = (vrp_3 + sum_2)(x_3^{2^*}(x_2^2), x_2^2, r_2)\ . \quad (6)$$

The position of r in the pseudo-tree therefore defines the amount of information agents exchange about the dependencies of their respective costs on r. In Local-$\mathbb{E}$[DPOP], no such information is exchanged, since agents only report their evaluated costs, while in Global-$\mathbb{E}$[DPOP], this information is propagated up the pseudo-tree until $lca(r)$. While this additional information exchange can yield higher-quality solutions (Section 5), both algorithms remain incomplete, because they use inconsistent pseudo-trees (Section 3). One exception to this incompleteness property is when the evaluation function is *linear*, as formally shown in (Léauté and Faltings 2009), which is the case of the *expectation* function. Therefore, Local-Exp-$\mathbb{E}$[DPOP] and Global-Exp-$\mathbb{E}$[DPOP] are complete and output the same solutions; in our experiments (Section 5), we only consider Local-Exp-$\mathbb{E}$[DPOP], which has a lower complexity. For the non-linear *consensus* and *robust* functions, completeness is not guaranteed.

Algorithm 1 Computational steps for the *expectation* approach, to compute $x^* = \arg\min_{x \in D_x} \{\mathbb{E}_r [c(x, r)]\}$ and report $\mathbb{E}_r [c(x^*, r)]$, using in Eq. (2).

```
1: x* ← null; c* ← ∞
2: for each decision x ∈ D_x do
3:     c ← 0
4:     for each scenario r_i ∈ Δ_r do
5:         c ← c + π_r(r_i)c(x, r_i) // 1 constraint check
6:         if c > c* then continue to next decision
7:     if c ≤ c* then
8:         x* ← x; c* ← c
9: E_r [c(x*, r)] ← c*
```

4.2 Hybrid *Consensus-Expectation* Approach

The general approach described in the previous section actually does not apply *as is* to the *consensus* evaluation function, because it is not *commensurable*: while the evaluations $\mathbb{E}_r [c(x, r)]$ and $\max_r c(x, r)$ remain costs, $\mathbb{P}_r[c(x, r)]$ is instead a *probability* of optimality (Section 2.3). In Figure 4a), this means variable x_3^2 cannot compute $c^*_{x_3^2}(x_2^2)$ as in Eq. (5) and report it to x_2^2, because x_2^2 would not be able to sum $c^*_{x_3^2}(x_2^2)$ (which would then be a *probability*) with its local constraints to compute the aggregated cost of its subtree.

To address this, we propose a Cons-/Exp- hybrid, in which agents use the *consensus* evaluation function in Eq. (4), but the *expectation* function in Eq. (5). This is inspired by previous work by Bent and Van Hentenryck (2004), who have shown that, even when the goal is to minimize the expected cost, it can be beneficial to use the consensus function to choose the optimal values for the decision variables, because *consensus* can require fewer expensive (vrp) constraint checks (*CCs*) that *expectation*.

This is illustrated in Algorithms 1 and 2. In the worst case, *expectation* requires $|D_x| \cdot |\Delta_r|$ CCs. On the other hand,

#	inst.	H	$Q_{\max}$	split	Local-Rob-$\mathbb{E}$[DPOP]			Glocal-Rob-$\mathbb{E}$[DPOP]			Comp-Rob-$\mathbb{E}$[DPOP]		
					cost	NCCC	info	cost	NCCC	info	cost	NCCC	info
1	p04	18.1	61		465.8	**73986**	**5.9**	465.8	78339	38.0	465.8	139266	36.9
2			62	✓	*476.9*	**73986**	**5.9**	474.8	78339	38.0	474.8	139266	36.9
3			63		474.9	**73986**	**5.9**	474.9	78339	38.0	474.9	139266	36.9
4			64	✓	455.3	**73986**	**5.9**	455.3	78339	38.0	455.3	139266	36.9
5			65		446.3	**73986**	**5.9**	446.3	78339	38.0	446.3	139266	36.9
6			66	✓	453.3	**73986**	**5.9**	453.3	78339	38.0	453.3	139266	36.9
7	p08	55	490		1815.2	**33602**	**6.1**	1815.2	38403	41.5	1815.2	57602	16.9
8			494		1815.2	**33602**	**6.1**	1815.2	38403	41.5	1815.2	57602	16.9
9			498	✓	1740.0	**33602**	**6.1**	1740.0	38403	41.5	1740.0	57602	16.9
10	p11	26	140	✓	*466.3*	**495202**	**13.5**	460.9	514407	154.8	460.9	960002	243.8
11			144		445.2	**499202**	**13.5**	445.2	518403	154.8	445.2	960002	243.8

Table 1: Worst-case cost, NCCCs and information exchange (in kB) for algorithms using the *robust* evaluation function. The column labelled *split* specifies whether the optimal solution involves splitting at least one customer's demand among depots.

Algorithm 2 Computational steps for the *Local-Cons-* approach, to compute $x^* = \arg\min_{x \in D_x} \{-\mathbb{P}_r[c(x,r)]\}$ using Eq. (3), and report $\mathbb{E}_r[c(x^*,r)]$ as in Eq. (2).

1: $\forall x \in D_x \ \pi[x] \leftarrow 0$ // initial probabilities of optimality
2: $x^* \leftarrow$ **null**
3: $\pi_2 \leftarrow 0$ // second-best probability of optimality
4: **for** each scenario $r_i \in \Delta_r = \{r_1, \ldots, r_k\}$ **do**
5: $\quad x'^* \leftarrow$ **null**; $c'^* \leftarrow \infty$
6: $\quad$ **for** each decision $x' \in D_x$ **do**
7: $\quad\quad c' \leftarrow c(x', r_i)$ // 1 constraint check
8: $\quad\quad$ **if** $c' < c'^*$ **then**
9: $\quad\quad\quad x'^* \leftarrow x'; c'^* \leftarrow c'$
10: $\quad \pi[x'^*] \leftarrow \pi[x'^*] + \pi_r(r_i)$
11: $\quad$ **if** $\pi[x'^*] \geq \pi[x^*]$ **then**
12: $\quad\quad$ **if** $x'^* \neq x^*$ **then** $\pi_2 \leftarrow \pi[x^*]$
13: $\quad\quad x^* \leftarrow x'^*$
14: $\quad$ **else if** $\pi[x'^*] > \pi_2$ **then** $\pi_2 \leftarrow \pi[x'^*]$
15: $\quad$ **if** $\sum_{j=i+1}^{k} \pi_r(r_j) \leq \pi[x^*] - \pi_2$ **then break**
16: $\mathbb{E}_r[c(x^*,r)] \leftarrow \sum_{r_i \in \Delta_r} \pi_r(r_i) c(x^*, r_i)$ // $|\Delta_r|$ CCs

for each scenario (Algorithm 2, line 4), *consensus* computes an optimal decision x'^*, which requires 1 CC per $x' \in D_x$ (line 7), but it might not be necessary to consider all values of r (all *scenarios*) to find the decision x^* with the highest probability of optimality $\pi[x^*]$. The exploration can be interrupted as soon as the remaining scenarios are too improbable for the second-best solution (of probability of optimality π_2) to catch up with the first best (line 15). In the best case, when the first $\frac{1}{2}|\Delta_r|$ scenarios have the same optimal decision, *consensus* only performs $\frac{1}{2}|\Delta_r|(|D_x|+2)$ CCs (where the $+2$ comes from line 16).

5 Experimental Results

Experiments were performed using the FRODO platform (Léauté, Ottens, and Szymanek 2009), on a 2.53-GHz computer, with 2 GB of Java heap. The OR-Objects Library (OpsResearch 2010) was used to provide approximate solutions to the local VRP instances. We adapted some of the Cordeau MDVRP benchmark instances from (Dorronsoro 2007), adding uncertainty in the positions of the customers that can be served by more than one depot, using 4 uncertain positions with the probability distribution $[\frac{1}{10}, \frac{2}{10}, \frac{3}{10}, \frac{4}{10}]$.

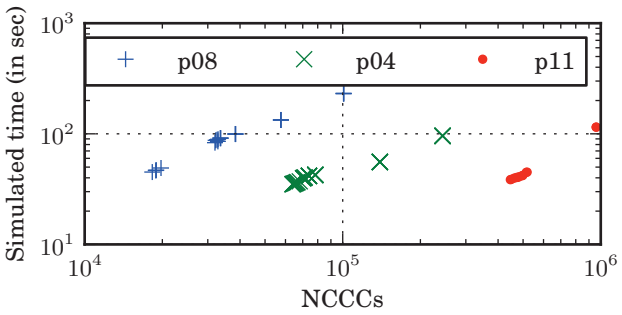

Figure 5: Median simulated time (over 27 runs) for the problem instances and the algorithms in Tables 1 and 2.

To evaluate runtime performance, we used two competing metrics: the number of *Non-Concurrent Constraint Checks (NCCCs)* by Gershman et al. (2008) (only counting the expensive vrp constraint checks, ignoring the inexpensive sum constraints), and the *simulated time* metric by Sultanik, Lass, and Regli (2007). Profiling some algorithm executions revealed that the vrp constraint checks are, by far, the computational bottleneck, so we expected the two metrics to be consistent. Figure 5 confirms that the simulated time is indeed proportional to the number of NCCCs (in *log-log*, the lines have slope 1). This was *not* observed when also counting sum constraint checks. The average cost of one NCCC (the distance from the x axis) depends on the number of customers in the VRPs, and therefore varies with the problem instance. For a given instance, the vrp NCCC costs also vary, but always average out in the same way, because E[DPOP] performs Dynamic Programming. Therefore, we hereafter only report runtime in terms of (vrp) NCCCs.

Table 1 compares Local-, Global- and Comp- using the *robust* evaluation function. While theoretically incomplete, Global- systematically found the optimal solution output

#	split	Local-Cons-$\mathbb{E}$[DPOP]			Global-Cons-$\mathbb{E}$[DPOP]			Comp-Cons-$\mathbb{E}$[DPOP]			Local-Exp-$\mathbb{E}$[DPOP]		
		cost	prob.	NCCC	cost	prob.	NCCC	cost	prob.	NCCC	cost	prob.	NCCC
1		465.15	100	**64362**	465.15	100	65924	465.15	100	243716	465.15	100	70961
2	✓	472.49	20	66003	472.49	20	**64358**	472.49	20	243716	472.49	20	70162
3	✓	471.25	46	68139	471.25	46	**66796**	471.25	46	243716	471.25	46	70244
4	✓	453.63	100	**65218**	453.63	100	65278	453.63	100	243716	453.63	100	70392
5	✓	445.28	80	63904	445.28	80	**63852**	445.28	80	243716	445.28	80	71316
6		*452.17*	*20*	**64572**	*452.17*	*20*	66539	*452.17*	*20*	243716	452.1	50	70974
7		1789.93	41	18859	1789.93	41	**18281**	*1791.22*	*30*	100804	1789.93	41	32576
8		*1790.43*	*29*	19763	*1790.43*	*29*	**18947**	*1791.22*	*30*	100804	1789.93	41	32182
9	✓	1735.95	75	32830	1735.95	75	**31990**	1735.95	75	100804	1735.95	75	32884
10	✓	454.95	33	449824	454.95	33	**446301**	454.95	33	1675204	454.95	33	476382
11		436.67	81	**461091**	436.67	81	470501	436.67	81	1680004	436.67	81	482360

Table 2: Expected cost, probability of optimality (in %), and NCCCs for the *consensus* and *expectation* approaches. Information exchange is the same as in Table 1: it only depends on the method (Local-, Global- or Comp-) and not on the evaluation function.

by the (complete[2]) Comp- approach. We believe this is due to the multiplicity of optimal solutions in these benchmarks. Local- also found the optimal solution, except on instances (in italics) for which the optimal worst-case solution involves splitting one customer's demand, which happens when the number of vehicles one depot needs depends on the demand allocation. These solutions are intuitively harder to find because they require a finer-grained coordination among the depots. This coordination comes at an additional price that can make Global- send more information than Comp- (except on p11). Comp- is however significantly slower, and they are both dominated by Local- (in bold).

Table 2 shows the results for the *consensus* and *expectation* evaluation functions. The Cons- algorithms almost always found the solution with minimum expected cost, with a few rare exceptions (in italics). In terms of runtime, Local-Cons- and Global-Cons- perform roughly the same; their relative performances depend on how much of the uncertainty space they can prune (Algorithm 2, line 15) on each problem instance. The expectation approach is always outperformed, and Comp-Cons-$\mathbb{E}$[DPOP] is also clearly dominated.

6 Conclusion

We have introduced the *StochDCOP* framework for multi-agent optimization under uncertainty, represented by random variables. We have illustrated it on a Vehicle Routing Problem variant with uncertain customer locations. We have proposed several StochDCOP algorithms, which differ: 1) on how much information agents exchange about the dependencies of their cost functions on the random variables (*local*, *global* or *complete* reasoning), and 2) on the criterion they use to evaluate solution quality (*expectation*, *consensus* or *robustness*). Our experiments have shown that incomplete algorithms can be significantly cheaper than complete ones, while still finding the optimal solution on most benchmark instances. Future work includes studying the convergence when *sampling* is used to approximate the uncertainty space, and generalizing to *multi-stage* stochastic optimization.

[2]modulo the fact that the VRP algorithm provided by the OR-Object library is incomplete, and may report sub-optimal routes.

References

Bent, R., and Van Hentenryck, P. 2004. The value of consensus in online stochastic scheduling. In *14th Intl Conf. on Automated Planning and Scheduling (ICAPS'04)*, 219–226.

Dechter, R. 2003. *Constraint Processing*. Morgan Kaufmann.

Dorronsoro, B. 2007. The VRP Web benchmark repository. `http://neo.lcc.uma.es/radi-aeb/WebVRP/`.

Gershman, A.; Zivan, R.; Grinshpoun, T.; Grubshtein, A.; and Meisels, A. 2008. Measuring distributed constraint optimization algorithms. In *Proceedings of DCR'08*, 17–24.

Léauté, T., and Faltings, B. 2009. E[DPOP]: Distributed constraint optimization under stochastic uncertainty using collaborative sampling. In *Proceedings of DCR'09*, 87–101.

Léauté, T.; Ottens, B.; and Faltings, B. 2010. Ensuring privacy through distributed computation in multiple-depot vehicle routing problems. In *Proc. of AILog'10*, 25–30.

Léauté, T.; Ottens, B.; and Szymanek, R. 2009. FRODO 2.0: An open-source framework for distributed constraint optimization. In *Proceedings of the DCR'09 Workshop*, 160–164. `http://liawww.epfl.ch/frodo/`.

Matsui, T.; Matsuo, H.; Silaghi, M.-C.; Hirayama, K.; Yokoo, M.; and Baba, S. 2010. A quantified distributed constraint optimization problem. In *AAMAS'10*, 1023–1030.

Modi, P. J.; Shen, W.-M.; Tambe, M.; and Yokoo, M. 2005. ADOPT: Asynchronous distributed constraint optimization with quality guarantees. *Artificial Intelligence* 161:149–180.

OpsResearch. 2010. Operations Research – Java Objects. `http://or-objects.org/`.

Petcu, A., and Faltings, B. 2005. DPOP: A Scalable Method for Multiagent Constraint Optimization. In *Proc. 19th Intl Joint Conf. on Artificial Intelligence (IJCAI'05)*, 266–271.

Sultanik, E. A.; Lass, R. N.; and Regli, W. C. 2007. DCOPolis: A framework for simulating and deploying distributed constraint optimization algorithms. In *Proc. of DCR'07*.