


1 Ex-Ante Constraint Elicitation in Incomplete 2 DCOPs

3 Roie Zivan ✉ 

4 Ben-Gurion University of the Negev, Israel

5 Shiraz Regev ✉

6 Ben-Gurion University of the Negev, Israel

7 William Yeoh ✉ 

8 Washington University in St. Louis, United States

9 — Abstract —

10 Distributed Constraint Optimization Problems (DCOPs) is a framework for representing and solving
11 distributed combinatorial problems, where agents exchange messages to assign variables they own,
12 such that the sum of constraint costs is minimized. When agents represent people (e.g., in meeting
13 scheduling problems), the constraint information that the agents hold may be incomplete. For such
14 scenarios, researchers proposed Incomplete DCOPs (I-DCOPs), which allow agents to elicit from
15 their human users some of the missing information. Existing I-DCOP approaches evaluate solutions
16 not only by their quality, but also the elicitation costs spent to find them (*ex-post*). Unfortunately,
17 this may result in the agents spending a lot of effort (in terms of elicitation costs) to find high-quality
18 solutions, and then ignoring them because previous lower-quality solutions were found with less
19 effort.

20 Therefore, we propose a different approach for solving I-DCOPs by evaluating solutions based
21 on their quality and considering the elicitation cost beforehand (*ex-ante*). Agents are limited in
22 the amount of information that they can elicit and, therefore, need to make smart decisions on
23 choosing which missing information to elicit. We propose several heuristics for making these decisions.
24 Our results indicate that some of the heuristics designed produce high-quality solutions, which
25 significantly outperform the previously proposed ex-post heuristics.

26 **2012 ACM Subject Classification** Theory of computation → Distributed algorithms

27 **Keywords and phrases** Distributed Constraint Optimization Problems; Preference Elicitation; Multi-
28 Agent Optimization

29 **Digital Object Identifier** 10.4230/LIPIcs.CP.2024.9

30 **Acknowledgements** This research is partially supported by US-Israel Binational Science Foundation
31 (BSF) grant #2022189 and by a J.P. Morgan Faculty Research Award.

32 **1** Introduction

33 The Distributed Constraint Optimization Problem (DCOP) formulation is widely used
34 for representing and solving combinatorial optimization problems that are distributed by
35 nature [5, 7, 15]. It includes agents holding variables, which are constrained with variables
36 held by other agents (their neighbors) and attempt to find an optimal assignment to their
37 variables that minimizes constraint costs, while exchanging messages with their neighbors.

38 When agents represent humans, such as in meeting scheduling problems [4, 1], the
39 information held by agents regarding the preferences of the humans that they represent may
40 be incomplete. Agents can elicit information from the humans by introducing queries to
41 their human users, However, humans might find that answering these queries is a tedious
42 task and may abandon the use of the system if the burden is too heavy. Thus, there is a
43 clear need to limit the amount of queries that the human users need to answer.



© Roie Zivan, Shiraz Regev, and William Yeoh;
licensed under Creative Commons License CC-BY 4.0

30th International Conference on Principles and Practice of Constraint Programming (CP 2024).

Editor: Paul Shaw; Article No. 9; pp. 9:1–9:16



Leibniz International Proceedings in Informatics

LIPICs Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

44 In order to represent such situations and allow agents to select high-quality assignments
 45 to their variables, while taking into consideration incomplete information, and make a limited
 46 use of elicitation queries, the Incomplete DCOP (I-DCOP) model was proposed [10, 11, 12].
 47 I-DCOP enables the representation of partial information by having agents hold constraint
 48 tables in which some entries include the costs for the corresponding combination of assignments
 49 and some do not. The agents can use elicitation queries to fill some of the empty entries and
 50 then use the information available to them in order to select the solution to the problem.

51 Tabakhi *et al.* [11, 12] proposed limiting the use of elicitation queries by evaluating the
 52 outcome of the I-DCOP solving process as a weighted sum of the quality of the selected
 53 solution and the effort (e.g., number of queries asked) for producing it. Thus, the agents
 54 aimed to find a solution that has not only a high quality, but also a low effort to find it.
 55 While this evaluation of outcomes incentivizes the algorithm to make efficient use of the
 56 human query resources, from a practical point of view, this method for evaluating possible
 57 outcomes does not make sense.

58 For example, imagine that an agent is searching for a hotel for the next trip of the person
 59 it represents. After a small search effort c_1 , the agent finds a decent hotel with solution
 60 quality q_1 . Then, the agent decides to spend more effort, searching for a better hotel and,
 61 after a costly effort $c_2 \gg c_1$, it manages to find one that is slightly better $q_2 > q_1$. According
 62 to the evaluation method proposed [11, 12], the agent will choose the first hotel because
 63 $c_1 - q_1 < c_2 - q_2$.¹ In other words, the second hotel is not as good because the marginal
 64 increase in quality is not worth the large amount of effort spent for it. However, intuitively,
 65 since the search effort was already spent, it does not make sense to not use the better solution
 66 found.

67 The key issue with the prior approach is that the search effort considered is done ex-post
 68 – *after* the effort was spent – when it should be done ex-ante – *before* the effort was spent.
 69 With this insight in mind, we propose a different approach for solving incomplete DCOPs.
 70 Inspired by others [3], we limit the amount of queries that agents can use (i.e., a query
 71 “budget”) and propose different heuristic strategies for the agents to follow when they decide
 72 what information to elicit. We compare the success of the proposed strategies in comparison
 73 with the existing ex-post approach, in combination with a complete SyncBB algorithm [2]
 74 and two incomplete DSA and MGM [15, 17] algorithms.

75 Our results indicate that all the ex-ante heuristic strategies we proposed outperformed
 76 the existing ex-post heuristic. Moreover, the heuristics that spend effort in identifying parts
 77 of the search space that have higher probability to be part of a high-quality solution are
 78 more successful.

79 **2 Background**

80 In this section, we present DCOPs and three algorithms for solving them: SyncBB, DSA,
 81 and MGM.

82 **2.1 Distributed Constraint Optimization Problems**

83 Without loss of generality, in the rest of this paper, we will assume that all problems are
 84 minimization problems, as it is common in the DCOP literature [1]. Thus, we assume that
 85 all constraints define costs and not utilities.

¹ We assume that we are minimizing costs in this paper.

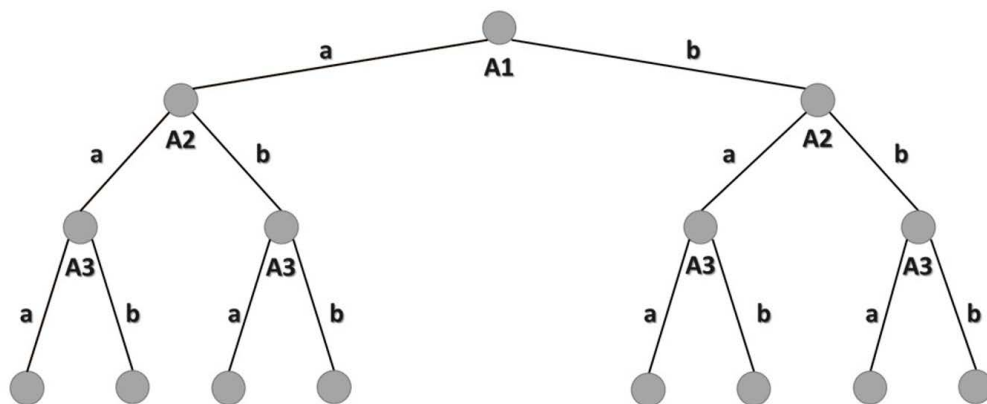
86 A DCOP is defined by a tuple $\langle \mathcal{A}, \mathcal{X}, \mathcal{D}, \mathcal{R} \rangle$. \mathcal{A} is a finite set of agents $\{A_1, A_2, \dots, A_n\}$.
 87 \mathcal{X} is a finite set of variables $\{X_1, X_2, \dots, X_m\}$. Each variable is held by a single agent, and
 88 an agent may hold more than one variable. \mathcal{D} is a set of domains $\{D_1, D_2, \dots, D_m\}$. Each
 89 domain D_i contains the finite set of values that can be assigned to variable X_i . We denote
 90 an assignment of value $x \in D_i$ to X_i by an ordered pair $\langle X_i, x \rangle$. \mathcal{R} is a set of relations
 91 (constraints). Each constraint $R_j \in \mathcal{R}$ defines a non-negative *cost* for every possible value
 92 combination of a set of variables, and is of the form $R_j : D_{j_1} \times D_{j_2} \times \dots \times D_{j_k} \rightarrow \mathbb{R}^+ \cup \{0\}$. A
 93 *binary constraint* refers to exactly two variables and is of the form $R_{ij} : D_i \times D_j \rightarrow \mathbb{R}^+ \cup \{0\}$.
 94 We say that a variable is *involved* in a constraint if it is one of the variables the constraint
 95 refers to and that an agent is involved in a constraint if one of its variables is involved
 96 in the constraint. We assume that agents hold all constraints that they are involved in.
 97 For each binary constraint R_{ij} , there is a corresponding cost table T_{ij} with dimensions
 98 $|D_i| \times |D_j|$ in which the cost in every entry e_{xy} is the cost incurred when x is assigned
 99 to X_i and y is assigned to X_j . A *binary DCOP* is a DCOP in which all constraints are
 100 binary. A *partial assignment* is a set of value assignments to variables, in which each variable
 101 appears at most once. $\text{vars}(PA)$ is the set of all variables that appear in partial assignment
 102 PA (i.e., $\text{vars}(PA) = \{X_i \mid \exists x \in D_i \wedge \langle X_i, x \rangle \in PA\}$). A constraint $R_j \in \mathcal{R}$ of the
 103 form $R_j : D_{j_1} \times D_{j_2} \times \dots \times D_{j_k} \rightarrow \mathbb{R}^+ \cup \{0\}$ is *applicable* to PA if each of the variables
 104 $X_{j_1}, X_{j_2}, \dots, X_{j_k}$ is included in $\text{vars}(PA)$. The set of constraints that are applicable to
 105 a partial assignment PA will be denoted by R_{PA} . $R_j(PA)$ is the cost of incurred which
 106 corresponds to R_j with respect to PA . When R_j does not apply to (PA) , $R_j(PA) = 0$. The
 107 *cost of a partial assignment* $C(PA)$ is the sum of costs of all constraints that are applicable
 108 to PA , i.e., $C(PA) = \sum_{R_j \in R_{PA}} R_j(PA)$. A *complete assignment* (or a *solution*) is a partial
 109 assignment that includes all the DCOP's variables (i.e., $\text{vars}(PA) = \mathcal{X}$). An *optimal solution*
 110 is a complete assignment with minimal cost.

111 For simplicity, we make the common assumption that each agent holds exactly one variable
 112 (i.e., $n = m$) and we concentrate on binary DCOPs. These assumptions are common in the
 113 DCOP literature [7, 13]. That being said, we emphasize that all methods and heuristics we
 114 propose in this paper apply to k -ary constraints as well, for $2 \leq k \leq n$.

115 2.2 Synchronous Branch-and-Bound (SyncBB)

116 Synchronous Branch-and-Bound (SyncBB) [2] is a complete, synchronous, search-based
 117 algorithm that can be considered as a distributed version of a standard branch-and-bound
 118 algorithm. It uses a complete ordering of the agents to extend a *Current Partial Assignment*
 119 (CPA) via a synchronous communication process. The CPA is exchanged by the agents
 120 according to the order. Agents add the assignments to their variables before sending the
 121 CPA forward and remove their assignments before sending it backwards. The CPA also
 122 functions as a mechanism to propagate bound information. The algorithm prunes those
 123 parts of the search space whose solution quality is sub-optimal by exploiting the bounds that
 124 are updated at each step of the algorithm. In other words, an agent backtracks when the
 125 cost of the CPA is not smaller than the cost of the best complete solution found so far.

126 The algorithm begins by the first agent in the order, which generates the CPA, assigns it a
 127 value and forwards it to the next in the order. The CPA includes a lower bound, which is the
 128 current cost of the partial assignment carried by the CPA and an upper bound (UB), which
 129 is the cost of the best solution found so far by the algorithm (initially infinity). When an
 130 agent A_i receives a CPA, it attempts to assign its variable X_i with one of the values $x \in D_i$
 131 and send it forward. When it is received back from the agent following it in the order (A_{i+1}),
 132 it attempts to reassign X_i with a different value from D_i . A CPA is sent back when the agent



■ **Figure 1** Example of a search tree

133 cannot assign a value to the CPA that has not been assigned to the CPA with the specific
 134 context (the partial assignment) before, or that does not cause a breach of UB. When the
 135 last agent in the order manages to assign its variable, without breaching UB, a new solution
 136 is generated and stored, and UB is updated with its cost. The algorithm terminates when
 137 the first agent sends the CPA back. The solution reported is the last complete assignment
 138 (solution) that caused an update of UB.

139 In order to analyze the performance of complete search algorithms, such as SyncBB, when
 140 solving constraint reasoning problems, such as DCOPs, it is common to use a search tree.
 141 The search tree is a tool that allows one to follow the advancement of the search process and
 142 analyze its properties. The root of the search tree is the first variable in the order, and each
 143 of the edges connecting it to its children represents a possible value assignment. Similarly the
 144 second layer represents the possible assignments of the second variable in the order and so
 145 forth, until the leaves of the tree, which represent the value assignments of the last variable
 146 in the order [14]. Thus, each value assignment is the root of a sub-tree in this search tree.

147 Figure 1 presents an example of a search tree with three agents A_1 , A_2 , and A_3 , each
 148 holding one variable with two values in its domain a and b .

149 2.3 Distributed Stochastic Algorithm (DSA)

150 The Distributed Stochastic Algorithm (DSA) [15] is a simple distributed local search algorithm
 151 in which, following an initial step where agents (randomly) choose an initial value for their
 152 variable, the agents perform a series of steps (looped iteratively) until some termination
 153 condition is met. In every step, an agent sends its value assignment to its neighbors in
 154 the constraint graph and collects the value assignments of its neighbors. Once the value
 155 assignments of all its neighbors have been collected, an agent decides whether to keep its
 156 value assignment or to modify it. This decision has a significant effect on the performance
 157 of the algorithm. If an agent in DSA cannot upgrade its current state by substituting its
 158 present value, it does not do so. On the other hand, if the agent can improve (or maintain,

159 depending on the version used) its current state, it decides whether to replace its value
 160 assignment using a stochastic strategy.

161 2.4 Maximum Gain Message (MGM)

162 Like DSA, Maximum Gain Message (MGM) is a distributed synchronous local search
 163 algorithm, in which agents perform in iterations. In each iteration the agents send messages
 164 to all their neighbors, receive messages from all of them and perform computation. The main
 165 difference from DSA is that, for each decision whether to replace an assignment, two iterations
 166 are performed. In the first, like in DSA, the agents exchange their value assignments. In
 167 the second, the agents exchange the maximal improvement they can achieve by replacing
 168 assignments. Only agents that suggested a positive improvement that is greater than all their
 169 neighbors (ties are broken deterministically according to the agents' identifying indexes),
 170 replace their assignments.

171 3 Ex-Ante Incomplete DCOP

172 An Ex-Ante Incomplete DCOP (EAI-DCOP) is defined by a tuple $\langle \mathcal{A}, \mathcal{X}, \mathcal{D}, \mathcal{R}, \tilde{\mathcal{R}}, \mathcal{E}, \mathcal{B} \rangle$,
 173 where \mathcal{A} , \mathcal{X} , \mathcal{D} and \mathcal{R} are defined the same as in DCOP. For each constraint $R_j \in \mathcal{R}$, there
 174 is a corresponding incomplete constraint $\tilde{R}_j \in \tilde{\mathcal{R}}$, where $\tilde{R}_j \in \tilde{\mathcal{R}} : D_{j_1} \times D_{j_2} \times \dots \times D_{j_k} \rightarrow$
 175 $\mathbb{R}^+ \cup \{0, ?\}$, where each of the D_{j_q} in \tilde{R}_j is also a member in R_j and $?$ is a special element
 176 denoting that the cost for a given combination of value assignments is not known to the
 177 agent. In I-DCOP, it is assumed that an agent does not hold the set of constraints that it is
 178 involved in, but rather the set of incomplete constraints that it is involved in.

179 For every incomplete constraint \tilde{R}_j , there is an elicitation cost function $E_j \in \mathcal{E}$, such that
 180 for each unknown cost of a combination of assignments $r \in R_j$ there is a positive elicitation
 181 cost in $e(r) \in E_j$ that the agent will need to "pay" for eliciting this constraint. An explored
 182 solution space $\tilde{\mathbf{x}}$ is the union of all solutions explored so far by a particular algorithm. $\tilde{\mathbf{x}}_{PA}$
 183 is the explored solution space at the time that PA was generated. The cumulative elicitation
 184 cost $\mathcal{E}(\tilde{\mathbf{x}})$ (and $\mathcal{E}(\tilde{\mathbf{x}}_{PA})$ respectively) is $\sum_{r \in R} e(r)$ such that r is an unknown constraint in
 185 $\tilde{\mathcal{R}}$, but it is not an unknown constraint in $\tilde{\mathbf{x}}$. In other words, it is the sum of the elicitation
 186 costs of all elicitation queries conducted while exploring $\tilde{\mathbf{x}}$.

187 In standard (Ex-Post) I-DCOP [12], the cost $C(PA)$ of a partial assignment is calculated
 188 as follows: $C(PA) = \sum_{R_j \in R_{PA}} C(R_j) + \mathcal{E}(\tilde{\mathbf{x}}_{PA})$, where R_{PA} is the set of constraints whose
 189 variables are in $vars(PA)$. In an Ex-Ante I-DCOP, the solution cost, like in standard DCOP,
 190 is $C(PA) = \sum_{R_j \in R_{PA}} C(R_j)$. However, agents are limited in the amount of information
 191 they can elicit. We formulate this limitation using a budget $\mathcal{B} = \{B_1, B_2, \dots, B_n\}$, where B_i
 192 is the amount of elicitation cost agent A_i may spend. These are taken into consideration
 193 during the search process and, thus, the agents take the budget limitations into consideration
 194 **before** they decide whether to elicit some information.

195 Figure 2 includes an example of an EAI-DCOP with seven agents. Each agent holds one
 196 variable with three values in its domain, and has an elicitation budget (we only present the
 197 budgets of A_2 and A_7 , which are relevant to the example). On the right hand side the cost
 198 table and the elicitation cost table of constraint $R_{2,7}$ are presented. There are three unknown
 199 costs in the cost table. Agent A_2 's budget allows it to elicit the cost for $\langle X_2 = a, X_7 = a \rangle$ or
 200 $\langle X_2 = b, X_7 = a \rangle$. Agent A_7 can afford to elicit any of the three missing costs, and even to
 201 elicit the costs for both $\langle X_2 = a, X_7 = a \rangle$ and $\langle X_2 = b, X_7 = a \rangle$.

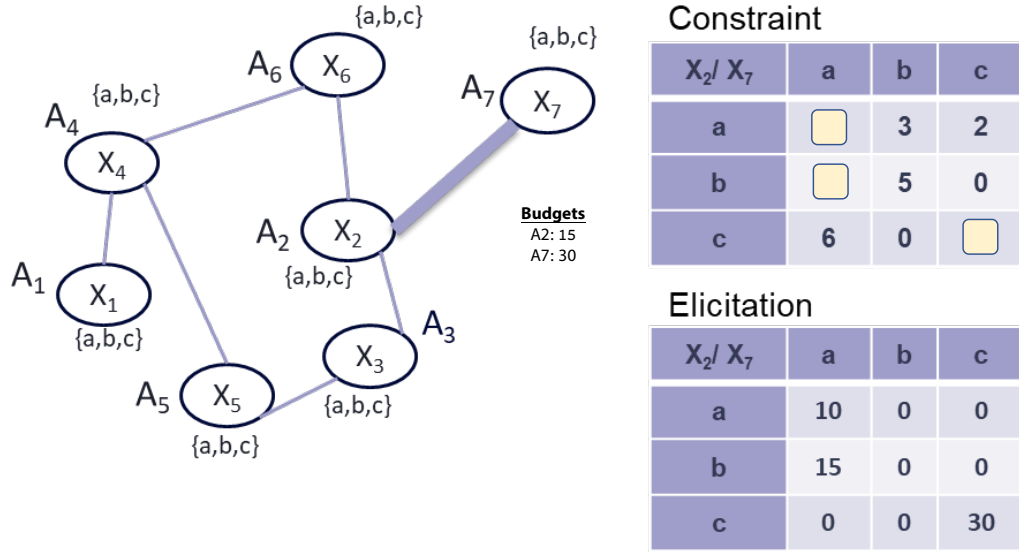


Figure 2 Example of an EAI-DCOP

4 Solving EAI-DCOPs

We propose ex-ante elicitation heuristics for three algorithms solving EAI-DCOPs – SyncBB [2], DSA [15], and MGM [4]. These well-known algorithms were selected for their simplicity, in order to emphasize the effect of the selected elicitation heuristic on the search process. SyncBB and MGM were also used in the previous I-DCOP studies [11, 12].

4.1 Solving EAI-DCOPs with SyncBB

The main difference between agents performing SyncBB to solve EAI-DCOPs from the agents performing SyncBB to solve standard DCOPs is that, in EAI-DCOPs, agents do not attempt to assign all values to their variables. Instead, when an agent that has partial information regarding the constraint costs of its variable receives a CPA, it needs to decide whether to elicit missing information and which missing information to elicit. We will assume that regardless of the heuristic being used, an agent will first attempt to assign values to its variable, for which it knows all costs of constraints with the value assignments included in the CPA. For the values in its domain for which it does not know all the constraint costs, the agent can decide either to elicit this information, and pay the corresponding cost (which is deducted from its budget), or to avoid eliciting this information. Obviously, if its budget is smaller than the elicitation cost, the first option is ruled out. After eliciting the constraint costs corresponding to a value $x \in D_i$, agent A_i treats x as any other value in its domain for which it has complete knowledge regarding its constraints, that is, it tries to assign x to X_i and send the CPA forward. On the other hand, if A_i decides not to elicit the costs that correspond to x , the subtree rooted by x (in the search tree) will not be explored. We propose the following heuristics for deciding whether to elicit the cost information by agents solving EAI-DCOPs with SyncBB.

Depth Dependent (DD): The decision whether to elicit the cost information for a value is decided stochastically. The probability for an agent A_i to elicit the costs of a value in its

227 domain is calculated using the Sigmoid function: $p(i) = \frac{e^i}{e^{n/2} + e^i}$, where i is the depth of the
 228 agent's variable in the search tree and n is the number of agents.² Note that this function
 229 does not distinguish between the values within a domain of a variable held by some agent
 230 and, thus, if the decision is to elicit, the agent will elicit the constraint cost information for
 231 all its values, until the budget is exhausted.

232 The intuition that led to the design of this heuristic was that the deeper an agent is in
 233 the search tree, the larger is the chance that a solution improving on former solutions will be
 234 found. This is because every layer in the search tree can require additional elicitation.

235 **Distance from Bound (DB):** The decision whether to elicit the cost information for a
 236 value $x \in D_i$ is based on its distance Δ_x of the cost of the CPA from the upper bound (UB)
 237 maintained by SyncBB and a threshold $t(i)$ of the agent a_i . Specifically, the costs for x are
 238 elicited if $\Delta_x > t(i)$.

239 The distance from the upper bound Δ_x for each value $x \in D_i$ is calculated as follows:
 240 $\Delta_x = UB - (C(PA) + \delta_x)$, where $C(PA)$ is the cost of the current partial assignment and
 241 δ_x is the lowest cost that was generated from constraints with assignments of variables held
 242 by agents that come after A_i in the order, in previous attempts to assign x to X_i . If there
 243 were no previous attempts, $\delta_x = 0$.

244 The threshold $t(i)$ is calculated as follows: $t(i) = g \cdot \left(1 - \frac{e^i}{e^{n/2} + e^i}\right)$, where g is a constant
 245 that is dependent on the distribution of constraint costs in the problem.

246 The intuition is that when the distance from the upper bound is larger, there is more
 247 chance that this part of the search tree will include relevant solutions, since for a small
 248 distance an additional cost is expected to breach the bound and cause a backtrack.

249 **Elicitation Required in Subtree (ERS):** The decision whether to elicit the costs corres-
 250 ponding to a value assignment is done according to the number of unknown constraints in
 251 the subtree (of the search tree) rooted by this value. The intuition is that there is a larger
 252 chance that in a subtree with a small number of unknown constraints, complete solutions
 253 will be found with low elicitation cost.

254 In EAI-DCOPs, it is possible to count the number of unknown constraints in each such
 255 subtree. The process is performed bottom up, having the last agent in the order count the
 256 unknown constraints for each of the values in its domain, and sending this information up
 257 to the agent preceding it in the order. This agent adds the amount of constraints for each
 258 of its value and sends it up to the agent preceding it, and so forth. The process ends when
 259 the first agent in the order, holding the root variable of the subtree, updates the number of
 260 unknown constraints in each of the subtrees rooted by the values in its domain.

261 In more details, after an agent A_i receives a message that includes a number C_{i+1} of
 262 unknown constraints from the agent following it in the order, it can calculate, for each value
 263 $x \in D_i$, the number of constraints it is involved in among them. Thus, in order to calculate
 264 the number of unknown constraints in the subtree rooted by x , it reduces from C_{i+1} the
 265 number of constraints that all values $x' \in D_i, x' \neq x$ are involved in. To this number, it adds
 266 the number of unknown constraints that x is involved in with variables held by agents that
 267 are before A_i in the order. The total number of unknown constraints that A_i sends to A_{i-1}
 268 is then calculated as follows: $C_i = \sum_{x \in D_i} C_x$.

269 Each agent performs the initial calculations of C_x for every value x in the domain of its
 270 variables before the algorithm begins as a preprocessing procedure. After the algorithm starts,

² Sigmoid functions are used as activation functions in neural networks [9], but are not related to their use here.

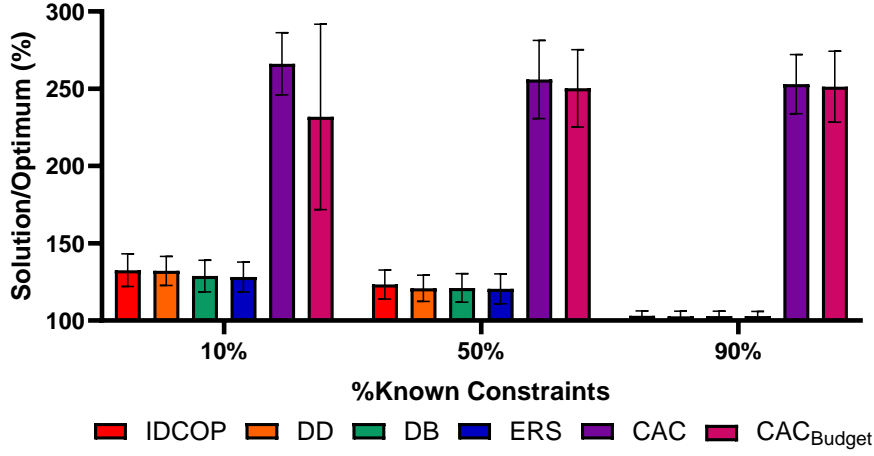


Figure 3 Solution costs when agents have a total budget of 105

271 these parameters are updated following each elicitation as follows: When agent A_i elicits
 272 the unknown constraints for a value $x \in D_i$, it updates the number of unknown constraints
 273 for this value, and updates each of the agents involved in the constraints that were elicited.
 274 These agents reduce the corresponding number of unknown constraints for the subtrees of
 275 the corresponding values.

276 4.2 Solving EAI-DCOPs with DSA and MGM

277 Like in the case of SyncBB, the main challenge when solving EAI-DCOPs using distributed
 278 local search algorithms, such as the DSA and MGM, is in deciding which constraints to elicit.
 279 However, in contrast to SyncBB, DSA and MGM are incomplete and, thus, not all values
 280 with known constraints need to be assigned. Therefore, the decision whether to elicit needs
 281 to take into consideration the amount of uncertainty regarding the cost of a possible value
 282 assignment and the potential improvement it offers. In more details, we propose a heuristic
 283 that we incorporated in both DSA and MGM, according to which the elicitation decision for
 284 value $x \in D_i$ is performed if the following two conditions hold:

- 285 1. The number of unknown constraints that x is involved in is smaller than $q \cdot CN_x$, where
 286 CN_x is the total number of constraints that x is involved in and $0 \leq q \leq 1$.
- 287 2. The difference between the sum of the known constraints that x is involved in and the
 288 cost of the current partial assignment is larger than $g \cdot \frac{e^i}{e^{n/2} + e^i}$. Here, g is a constant as
 289 defined above for the formula in SyncBB and i is the iteration number.

290 5 Experimental Evaluation

291 In order to evaluate the success of our approach for generating high-quality I-DCOP solutions,
 292 under different elicitation “budget” restrictions, we performed experiments in which we
 293 compared the different heuristics that we proposed for solving EAI-DCOPs incorporated in
 294 DCOP algorithms, with the previous proposed approach for solving I-DCOP [12]. All our
 295 experiments were performed on a simulator, implemented in Python, on a Lenovo Carbon

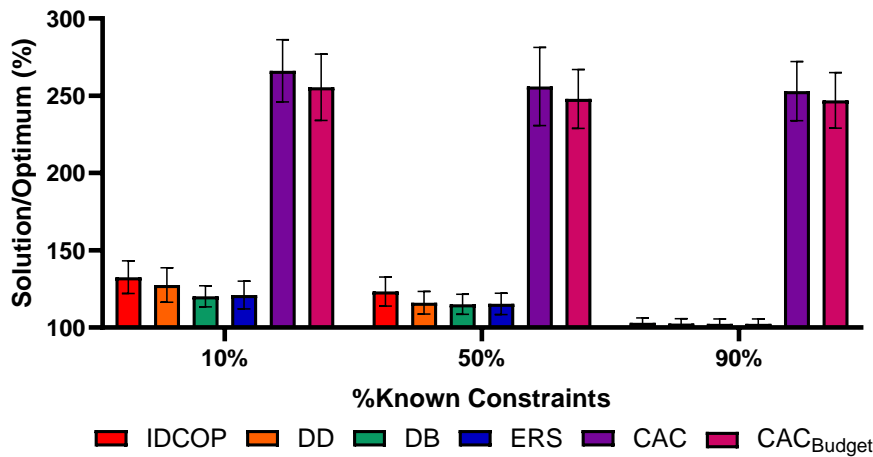


Figure 4 Solution costs when agents have a total budget of 525

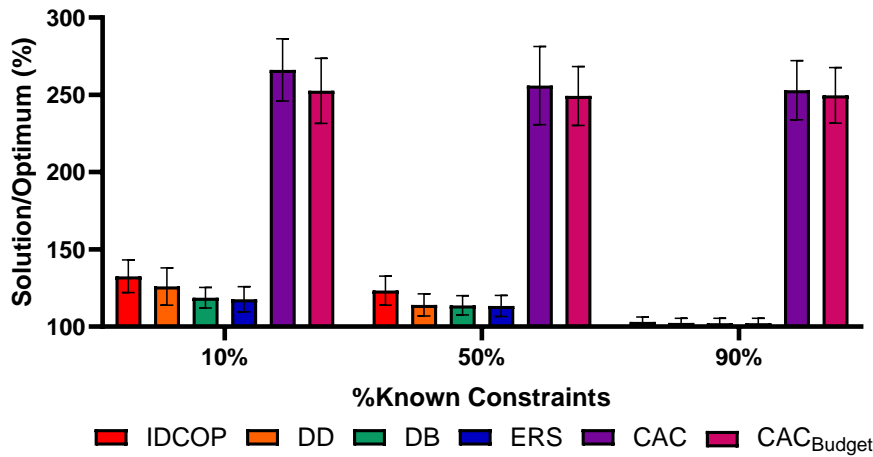


Figure 5 Solution costs when agents have a total budget of 1050

296 X1 Gen 9 computer with an 11th Generation Intel(R) Core (TM) i7-1165G7 @ 2.80GHz 2.80
 297 GHz processor.

298 The complete algorithms solved I-DCOPs including seven agents, each holding one
 299 variable with four values in its domain. The average number of neighbors that agents had
 300 was 3. Constraint costs for combinations of assignments of neighboring agents were selected
 301 uniformly between 2 and 5. The percentage of unknown constraints varied. We generated
 302 problems in which the fraction of known constraints was 10%, 50%, and 90%. For each
 303 unknown cost, an elicitation cost was selected uniformly from the range [0, 20].

304 5.1 SyncBB with Global Budgets

305 Previous I-DCOP approaches [12] with SyncBB assumed agents considered global elicitation
 306 costs, which are summations of elicitation costs over all agents. Therefore, to fairly compare
 307 against them, we also consider a variant, where agents have access to a global budget that

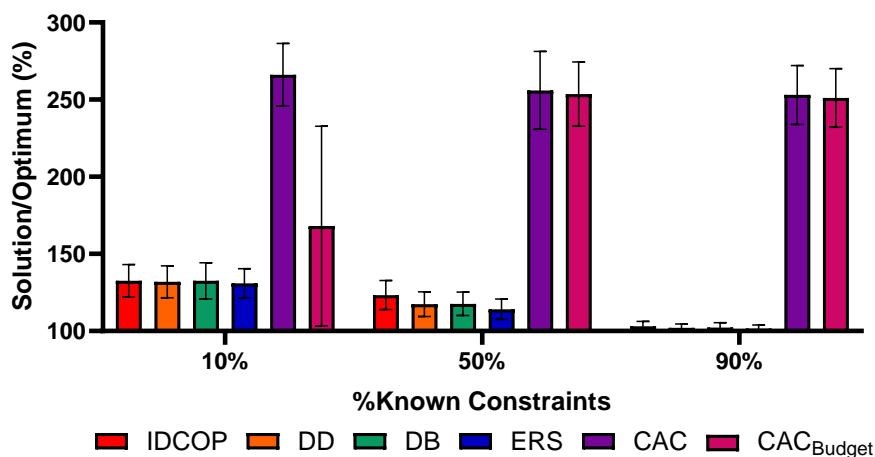


Figure 6 Solution costs when agents have a personal budget of 15

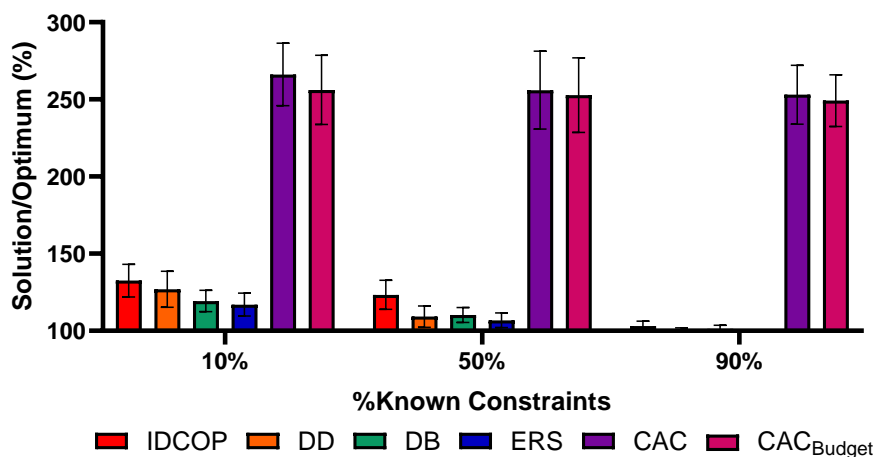


Figure 7 Solution costs when agents have a personal budget of 75

308 can be accessed by all agents. Specifically, we set the global budget to either 105, 525, or
 309 1050, so that the highest budget was an order of magnitude more than the lowest and they
 310 are easy to split among the agents in the personal budget version, which will be presented
 311 next. (We consider the variant where agents have personal budgets in the next section.)

312 Figure 3 depicts the average overhead in the solution quality with respect to the optimal
 313 solution (when all information is known), of solutions that the different algorithms produced
 314 when solving the problems with SyncBB, when agents were allocated the smallest global
 315 elicitation budget (105). The figure includes three batches of bars, one for each percentage
 316 of knowledge known to the agents in the beginning of the run of the algorithm (from left
 317 to right: 10%, 50%, and 90%). The two bars on the right in each batch are the results of
 318 the I-DCOP algorithm using the *CAC* heuristic proposed previously [12]. In one version
 319 (labeled *CAC_{budget}*), the algorithm was limited by a budget, similar to the amount used by
 320 the EAI-DCOP heuristics (i.e., 105). In the second (labeled *CAC*), the algorithm was not
 321 bounded by a budget. Surprisingly, the version that was limited by a budget produced better

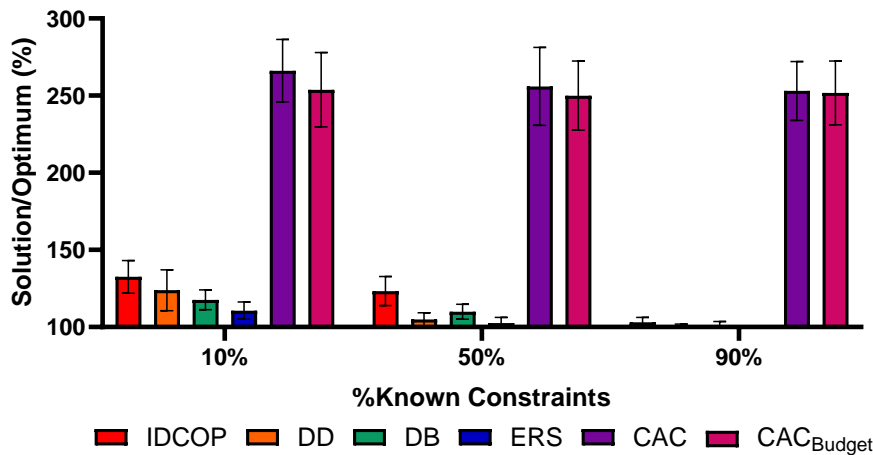


Figure 8 Solution costs when agents have a personal budget of 150

322 results on average. We assume that a limited budget limits also the elicitation cost that is
 323 taken into consideration in the lower bound of the algorithm and, thus, the agents explore
 324 more solutions when they have a limited budget. It is apparent that both of these versions
 325 produce solutions with much greater costs than the solutions produced by the other versions.
 326 Among them, the bar on the left (labeled IDCOP) is a version of the algorithm that does not
 327 perform elicitation at all, while the three others present the average results of the algorithm
 328 using the limited budget according to the three heuristics described in Section 4.1.

329 The results clearly indicate that it is enough to solve the I-DCOP using SyncBB with
 330 no elicitation, in order to get a much better solution in comparison with the solutions
 331 produced by the algorithm implementing the ex-post approach and heuristic suggested by
 332 the literature [12]. Yet, performing elicitation using the allocated budget can reduce costs
 333 further. The best result is achieved by the ERS heuristic. However, its advantage over DD
 334 and DB is not significant. Figures 4 and 5 present similar results produced by scenarios in
 335 which agents had larger budgets (525 and 1050 respectively). While the trends seem similar,
 336 it is clear that the advantage of the proposed heuristics over the I-DCOP version without
 337 elicitation, is more apparent (as expected).

338 5.2 SyncBB with Personal Budgets

339 Figures 6, 7, and 8 present results for the same algorithms solving the same problems, only
 340 in this case the EAI-DCOP heuristics, that is, DD, DB, and ERS use personal budgets
 341 instead of a global budget as used in the experiments presented above. We divided the
 342 global costs such that there will be no difference in the total budget used by the agents.
 343 However, these scenarios present the more realistic case where an agent represents a user,
 344 and the budget limits the effort a user must spend in replying to queries during search.
 345 In these personal budget scenarios, the versions using the EAI-DCOP heuristics produced
 346 solution with a more significant advantage in general over the vanilla I-DCOP version. The
 347 difference was most apparent when the agents had medium or high budgets, and the initial
 348 knowledge available was 50%. The EAI-DCOP heuristics were able to produce solutions with
 349 a significant advantage over the vanilla I-DCOP version, and for the 150 budget per agent,
 350 the DD and ERS heuristics produce solutions that their quality was close to optimal. It

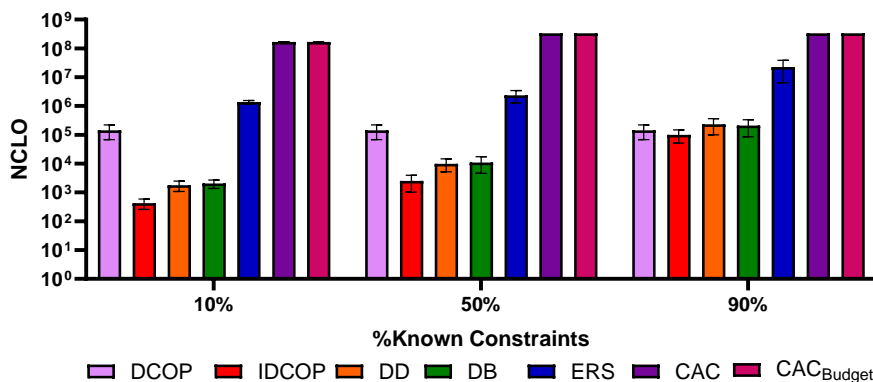


Figure 9 Runtime in terms of NCLOs with a budget of 105

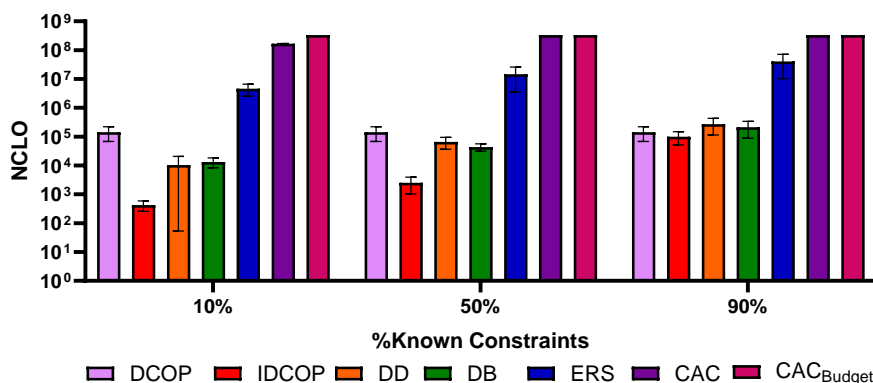
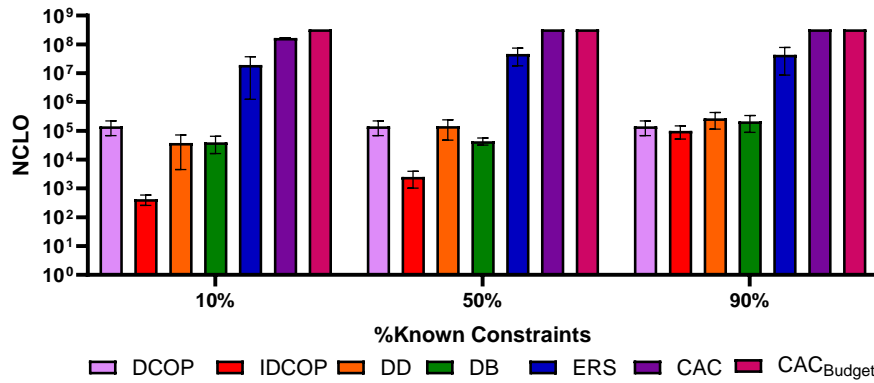


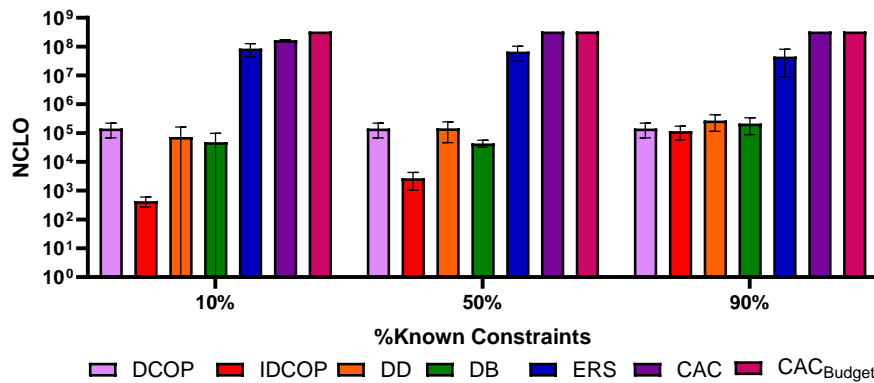
Figure 10 Runtime in terms of NCLOs with a budget of 525

351 seems that, besides being a more realistic scenario in a multi-agent environment, the budget
 352 per agent settings allows the algorithm to use elicitation in different parts of the search space
 353 and explore high quality solutions.

354 Figures 9, 10, and 11 present the runtimes in a logarithmic scale, of the algorithms in
 355 terms of NCLOs [16, 6]. In each figure, the budget allocated to the agents were different.
 356 The differences between the complete DCOP version and the I-DCOP version is identical
 357 in all figures because it is not affected by the budget. It is however affected by the amount
 358 of knowledge known to agents: In the 10% scenario, the algorithm solving I-DCOP is
 359 much faster than the algorithm solving the complete DCOP while, in the 90% scenario,
 360 the runtimes are much closer. It is also apparent that the CAC and ERS versions of the
 361 algorithms are much slower than all other versions. The reason is that these heuristics require
 362 exponential computation before the algorithm begins its search (preparing the heuristic data
 363 in preprocessing). Moreover, unlike the SyncBB algorithms that performs some level of
 364 pruning, the preprocessing heuristics aggregate information from the entire search space
 365 and thus, their runtime is orders of magnitude larger. On the other hand, The DD and DB
 366 heuristics are much faster. Additionally, their advantage over the ERS heuristic in runtime
 367 is much more significant than the advantage that ERS provides in solution quality. Similar
 368 results were obtained when agents used personal budget as presented in Figures 12, 13



■ **Figure 11** Runtime in terms of NCLOs with a budget of 1050



■ **Figure 12** Runtime in terms of NCLOs when each agent had a personal budget of 15

369 and 14

370 5.3 MGM and DSA with Personal Budgets

371 In the second set of experiments we performed, we compared incomplete algorithms, that is,
 372 MGM and DSA, which were implemented in the ALS framework [17], in order to produce
 373 the anytime solutions as was done previously in the literature [12]. In this set of experiments,
 374 the problems included 50 agents, each holding a single variable with 10 values in its domain.
 375 Agents had 20 neighbors in average. The costs of constraints were randomly selected between
 376 2 and 5. We present the solution costs of the algorithms in the first 50 iterations, because
 377 that was the number of iterations that were required for the algorithms to converge.

378 We first discuss the MGM results, since this was the algorithm that was implemented
 379 in the literature [12]. Figure 15 presents the solution cost as a function of the number of
 380 iterations performed by the algorithms. The DCOP version is the omniscient algorithm that
 381 knows all constraints. The dashed lines are the versions implementing the NHC heuristic [12].
 382 The different lines represent the amount of initial knowledge. The other solid lines represent
 383 the results of the EAI-DCOP version in which the elicitation was performed according to
 384 the heuristic presented in Section 4.2. Each agent was allocated 180 elicitation queries. The
 385 significant advantage of the EAI-DCOP version is apparent regardless of the amount of initial

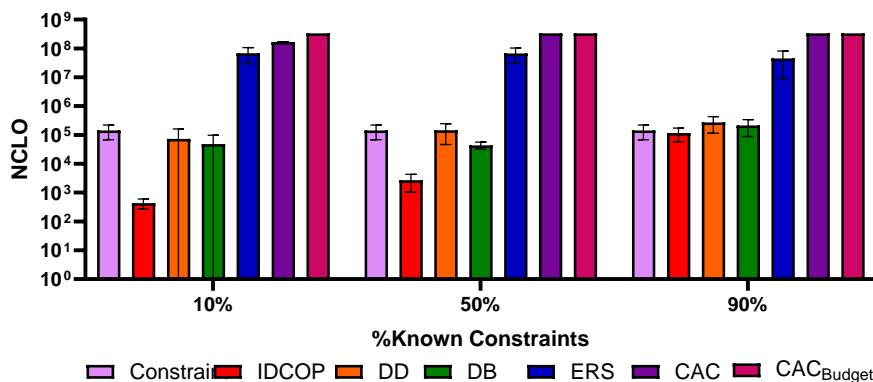


Figure 13 Runtime in terms of NCLOs when each agent had a personal budget of 70

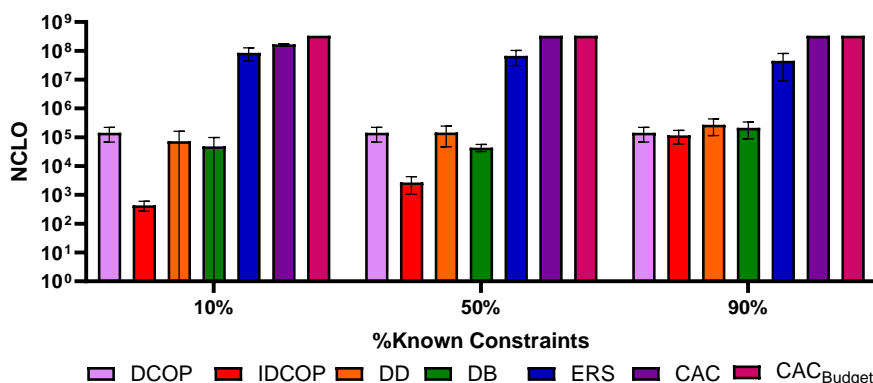


Figure 14 Runtime in terms of NCLOs when each agent had a personal budget of 150

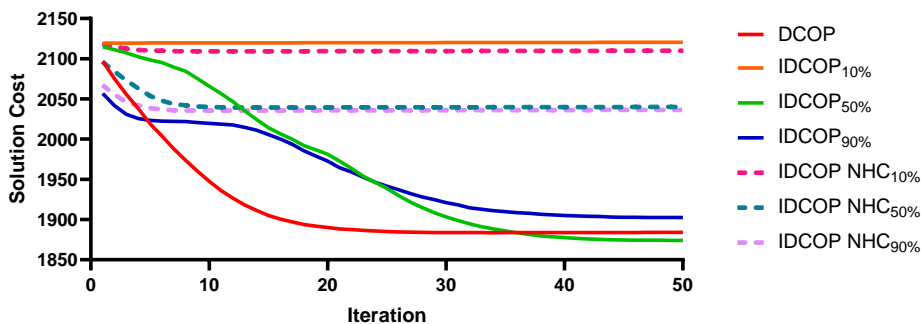


Figure 15 Solution cost as a function of the number of iterations

386 knowledge the agents had.

387 Figure 16 presents the results of MGM and DSA performing the EAI-DCOP heuristic for
 388 deciding on elicitation, with an allocation 180 queries per agent (with 50% initial knowledge
 389 available). Surprisingly, the EAI-DCOP version of MGM is more successful than the DSA
 390 version. This is in contrast to the well-known advantage that DSA has over MGM when

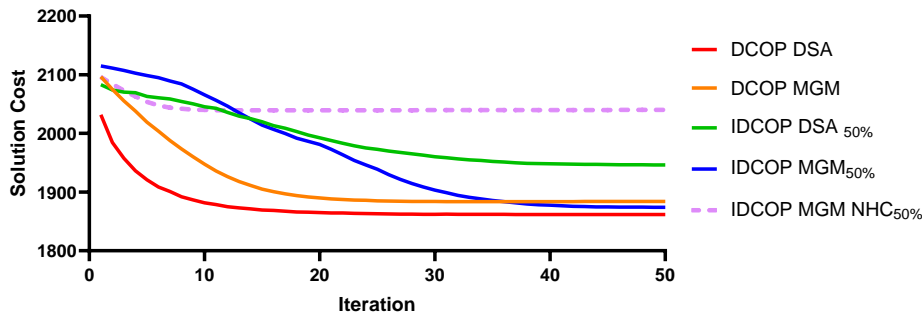


Figure 16 Solution cost as a function of the number of iterations

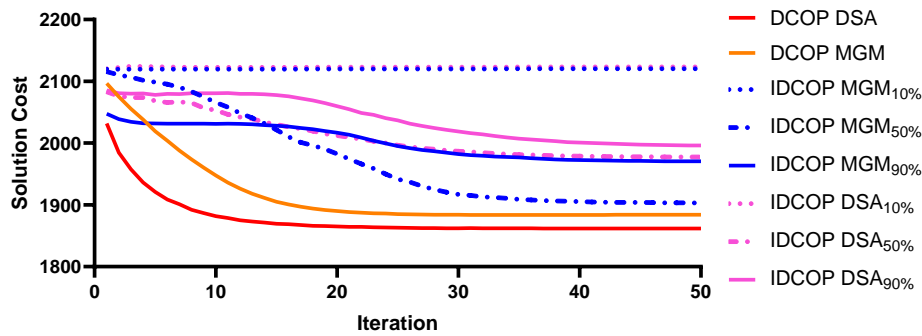


Figure 17 Solution cost as a function of the number of iterations

391 solving standard DCOPs.

392 Figure 17 presents the EAI-DCOP versions of MGM and DSA, when allocated fewer
 393 (120) queries per agent. Again, it is apparent that the MGM versions outperform the DSA
 394 versions, except for the versions solving problems with 10% initial knowledge, where both
 395 algorithms struggle. Moreover, the versions that solved problems with 50% initial knowledge
 396 outperform the versions that solved problems with 90% knowledge. We assume that less
 397 knowledge resulted in a positive exploration effect, as was reported for environments with
 398 imperfect communication [8].

399 6 Conclusions

400 We introduced a novel approach for solving I-DCOPs in this paper. In contrast to previous
 401 studies on I-DCOPs in which elicitation costs were considered *after* elicitations were made,
 402 we consider the costs *before* the elicitations in EAI-DCOPs.

403 The EAI-DCOP approach is not only more realistic, as it is not reasonable that agents
 404 will not use a high-quality solution after spending much effort to find it, it is also better in
 405 finding higher quality solutions *and* finding them with less runtime. These empirical results
 406 were shown on both complete and incomplete algorithms that are commonly used in the
 407 literature. Therefore, this seems to be one of the rare occasions where the new approach
 408 outperforms prior work on all three key relevant dimensions – practicality, solution quality,
 409 and runtime.

410 — References —

- 411 1 Ferdinando Fioretto, Enrico Pontelli, and William Yeoh. Distributed constraint optimization
412 problems and applications: A survey. *Journal of Artificial Intelligence Research*, 61:623–698,
413 2018.
- 414 2 Katsutoshi Hirayama and Makoto Yokoo. Distributed partial constraint satisfaction problem.
415 In *Proceedings of the International Conference on Principles and Practice of Constraint*
416 *Programming (CP)*, pages 222–236, 1997.
- 417 3 Tiep Le, Atena M. Tabakhi, Long Tran-Thanh, William Yeoh, and Tran Cao Son. Preference
418 elicitation with interdependency and user bother cost. In *Proceedings of the International*
419 *Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pages 1459–1467, 2018.
- 420 4 Rajiv Maheswaran, Jonathan Pearce, and Milind Tambe. Distributed algorithms for DCOP:
421 A graphical game-based approach. In *Proceedings of the International Conference on Parallel*
422 *and Distributed Computing Systems (PDCS)*, pages 432–439, 2004.
- 423 5 Pragnesh J. Modi, Wei-Min Shen, Milind Tambe, and Makoto Yokoo. ADOPT: Asynchronous
424 distributed constraint optimization with quality guarantees. *Artificial Intelligence*, 161(1–
425 2):149–180, 2005.
- 426 6 Arnon Netzer, Alon Grubshtein, and Amnon Meisels. Concurrent forward bounding for
427 distributed constraint optimization problems. *Artificial Intelligence*, 193:186–216, 2012.
- 428 7 Adrian Petcu and Boi Faltings. A scalable method for multiagent constraint optimization. In
429 *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, pages
430 1413–1420, 2005.
- 431 8 Ben Rachmut, Roie Zivan, and William Yeoh. Communication-aware local search for distributed
432 constraint optimization. *Journal of Artificial Intelligence Research*, 75:637–675, 2022.
- 433 9 Siddharth Sharma, Simone Sharma, and Anidhya Athaiya. Activation functions in neural
434 networks. *International Journal of Engineering Applied Sciences and Technology*, 4(12):310–316,
435 2020.
- 436 10 Atena M. Tabakhi, Tiep Le, Ferdinando Fioretto, and William Yeoh. Preference elicitation
437 for DCOPs. In *Proceedings of the International Conference on Principles and Practice of*
438 *Constraint Programming (CP)*, pages 278–296, 2017.
- 439 11 Atena M. Tabakhi, Yuanming Xiao, William Yeoh, and Roie Zivan. Branch-and-bound heur-
440 istics for incomplete DCOPs. In *Proceedings of the International Conference on Autonomous*
441 *Agents and Multiagent Systems (AAMAS)*, pages 1677–1679, 2021.
- 442 12 Atena M. Tabakhi, William Yeoh, and Roie Zivan. Incomplete distributed constraint op-
443 timization problems: Model, algorithms, and heuristics. In *Proceedings of the International*
444 *Conference on Distributed Artificial Intelligence (DAI)*, pages 64–78, 2021.
- 445 13 William Yeoh, Ariel Felner, and Sven Koenig. BnB-ADOPT: An asynchronous branch-and-
446 bound DCOP algorithm. *Journal of Artificial Intelligence Research*, 38:85–133, 2010.
- 447 14 William Yeoh, Pradeep Varakantham, Xiaoxun Sun, and Sven Koenig. Incremental DCOP
448 search algorithms for solving dynamic DCOP problems. In *Proceedings of the International*
449 *Conference on Web Intelligence and Intelligent Agent Technology (WI-IAT)*, pages 257–264,
450 2015.
- 451 15 Weixiong Zhang, Guandong Wang, Zhao Xing, and Lars Wittenburg. Distributed stochastic
452 search and distributed breakout: Properties, comparison and applications to constraint
453 optimization problems in sensor networks. *Artificial Intelligence*, 161(1–2):55–87, 2005.
- 454 16 Roie Zivan and Amnon Meisels. Message delay and DisCSP search algorithms. *Annals of*
455 *Mathematics and Artificial Intelligence*, 46:415–439, October 2006.
- 456 17 Roie Zivan, Steven Okamoto, and Hilla Peled. Explorative anytime local search for distributed
457 constraint optimization. *Artificial Intelligence*, 212:1–26, 2014.