An Artificial Bee Colony Based Algorithm for Continuous Distributed Constrained Optimization Problems

Tasnim Jubaier K. M. Merajul Arefin

April 18, 2021

Outline

Introduction

DCOPs - Distributed Constraint Optimization Problems

- Provides a model for multi-agent system
- Generalizes the Distributed Constraint Satisfaction Problem
- DCOP is NP-hard
- Usage:
  1. Allocating Resources
  2. Constructing Schedules
  3. Planning Activities

Classical DCOPs

- Works with discrete variables and domains
- Utilities/Cost are provided in tabular form
- Applied to:
  1. Sensor and wireless Networks
  2. Multi-robot Coordination

Continuous DCOPs (C-DCOPs)

- Works with continuous variables and domains
- Utilities/cost are provided in functions
- Applied to:
  1. Target Tracking Sensor Orientation
  2. Sleep Scheduling of wireless networks

Problem Definition

C-DCOPs can be defined as a tuple $\langle A, X, D, F, \alpha \rangle$ where,

- $A = \{a_i\}_{i=1}^{n}$ is a set of agents.
- $X = \{x_i\}_{i=1}^{m}$ is a set of continuous variable.
- $D = \{D_i\}_{i=1}^{m}$ is a set of continuous domains for each variable $x_i \in X$.
- $F = \{f_i\}_{i=1}^{l}$ is a set of utility functions, each f_i is defined over a subset $x^i = \{x_{i_1}, x_{i_2}, ..., x_{i_k}\}$ of variables X and the utility for that function f_i is defined for every possible value assignment of x^i.

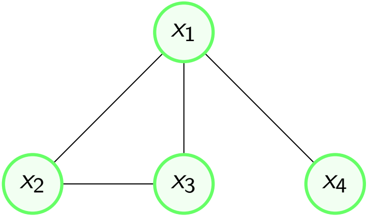
Problem Definition

- $\alpha : X \rightarrow A$ is a mapping function that associates each variable $x_j \in X$ to one agent $a_i \in A$. But an agent can control multiple variables.

The solution of a C-DCOP is an assignment X^* that maximizes the constraint aggregated utility functions as shown in Equation 7.

$$X^* = \underset{X}{\operatorname{argmax}} \sum_{f_i \in F} f_i(x^i)$$

Example of a C-DCOP



(a) Constraint Graph

$$\forall x_i \in X : D_i = [-10, 10]$$

$$f_{12}(x_1, x_2) = x_1^2 - cos(2\pi x_2)$$

$$f_{13}(x_1, x_3) = e^{\sqrt{x_1^2 + x_3^2}}$$

$$f_{14}(x_1, x_4) = (x_1 + 2x_4 - 7)^2$$

$$f_{23}(x_2, x_3) = x_2^2 + x_3^2 - x_2 x_3$$

(b) Utility Functions

Figure: Example of a C-DCOP

Related Works

- CMS and HCMS - Extension of discrete max-sum algorithm (2009-10)
- PFD - Population based algorithm based on Particle Swarm Optimization (2020)
- EC-DPOP, AC-DPOP and CAC-DPOP - Extension of inference based DPOP (2020)
- C-DSA - Extension of Distributed Stochastic Algorithm (DSA) (2020)

Reason for new algorithm

- CMS and HCMS doesn't provide good quality solutions based on empirical results
- EC-DPOP provides exact solution on only tree-structured systems
- AC-DPOP and CAC-DPOP provides good quality solutions but not usable on large systems due to memory consumption

Reason for new algorithm

- C-DSA is time efficient but doesn't provide good quality solutions
- PFD provides the best quality solutions among all the other algorithms but scalability remains an issue for this algorithm
- Continuous optimization methods such as gradient-based optimization require derivative calculations and thus they are not suitable for non-differentiable optimization problems

Artificial Bee Colony (ABC) algorithm

- Population based stochastic algorithm to find minimum or maximum value of a multi-dimensional numeric function
- Inspired from behaviour of honey bees
- Use case:
  - General Assignment Problem
  - Cluster Analysis
  - Structural Optimization

Artificial Bee Colony (ABC) algorithm

Algorithm 1: Artificial Bee Colony Algorithm

$P \leftarrow$ Set of random solutions
repeat
  $B \leftarrow$ Search improved solutions near solutions in P
  $P \leftarrow P \cup B$
  $C \leftarrow$ Search improved solutions near good solutions of P
  $P \leftarrow P \cup C$
  Discard solutions of P that did not improve
until *Requirements are met*

Challenges

- Store population in a distributed manner
- Calculate aggregated utility in a C-DCOP framework
- Update all the agents variables when a new solution is found
- Identify the best move in a single agent and what it can perceive

Proposed Solution

- ABCD algorithm which utilizes the ABC algorithm to work in a C-DCOP framework
- A noble technique to enhance the exploration ability of ABCD

The ABCD algorithm

- Population based stochastic algorithm
- Based on Artificial Bee Colony algorithm
- Tailored ABC to perform in distributed systems
- Provides better quality solutions than existing state of the art solutions
- An anytime algorithm
- Uses Elite set and dimension learning to improve the results
- A noble technique to enhance the exploration ability of ABCD
- ABCD-C to denote the classical ABCD and ABCD-E to denote ABCD with exploration technique

The ABCD algorithm - Initialization

- Create BFS pseudo-tree from the constraint graph
- Create S solutions for each agent randomly in the domain.

$$P_t^i.x_i = LB_i + r_t^i * (UB_i - LB_i)$$

  where r_t^i is a random number from $[0, 1]$ and P_t^i is the t-th object of the population stored by agent a_i
- Evaluate each P^i to calculate its aggregated utility

The ABCD Algorithm - Evaluate

- Each agent waits for the values from the neighbor agents N^i
- Aggregates all the functional utilities from those received values

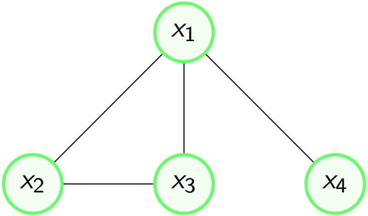
$$W_k^i.fitness \leftarrow \sum_{a_j \in N^i} f_{ij}(W_k^i.x_i, W_k^j.x_j)$$

- Wait for children agents CH^i to receive fitness values from them and aggregates them

$$W_k^i.fitness \leftarrow W_k^i.fitness + \sum_{a_j \in CH^i} W_k^j.fitness$$

The ABCD Algorithm - Evaluate

- Each agent except the ROOT agent sends fitness values to its parent
- ROOT agent divides the fitness values by 2 because each constraint utility function sums 2 times.



The ABCD Algorithm

- Take the best M solutions from the population into the E elite set
- Identify the best solution and propagate the solution in the pseudo-tree
- For each solution in the population, ROOT agent selects a random agent perform a search operation.

$$Q_u^i.x_i = \frac{1}{2}(E_l^h.x_h + Gbest_i.x_i) + \phi_u^i(P_u^h.x_h - E_l^i.x_i) \\ + \Phi_u^i(P_u^h.x_h - Gbest_i.x_i)$$

- Replace solutions whose updated version have higher utility than before
- ABCD-E marks that agent for that solution for future update.

The ABCD algorithm

- ROOT agent selects a random solution according to its selection probability

$$P_u^i.fit = \begin{cases} \frac{1}{1+abs(P_u^i.fitness)}, \text{if } P_u^i.fitness < 0 \\ 1 + P_u^i.fitness, \text{otherwise} \end{cases}$$

$$P_u^i.prob = \frac{P_u^i.fit}{\sum_{P_v^i \in P^i} P_v^i.fit}$$

- Agent performs search operation on that solution

$$R_m^i.x_i = \frac{1}{2}(E_m^h.x_h + Gbest_i.x_i) + \phi_m^i(P_u^h.x_h - E_l^i.x_i) + \Phi_m^i(P_u^h.x_h - Gbest_i.x_i)$$

The ABCD algorithm

- Replace solutions whose updated version have higher utility than before
- ABCD-E marks that agent for that solution for future update.
- in ABCD-E, ROOT agent checks for solution whose mark's are full, and replaces them with new solution
- ABCD-C holds an extra parameter *LIM* to see which solutions have been explored more than *LIM* times and replaces them with new solutions.

Experimental Results

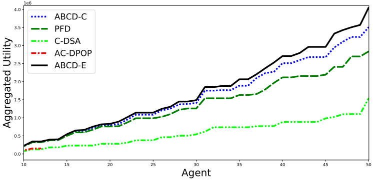


Figure: Random Graph(Dense)

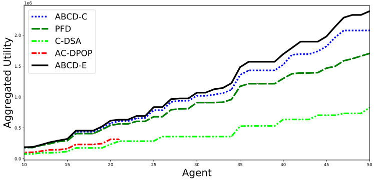


Figure: Random Graph(Sparse)

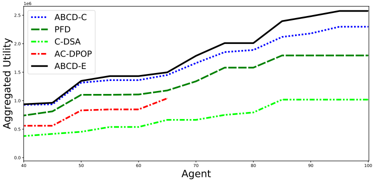


Figure: Scale Free

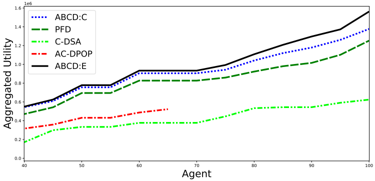


Figure: Small World

Future Works

- Explore applicability of ABC in DCOP framework
- Design new heuristics to fit in ABCD algorithm
- Explore other population based algorithms usage in C-DCOP framework

Thank You