ABSTRACT
In this paper, we develop a novel MO-DCOP algorithm based on dynamic programming techniques which guarantees to find the complete Pareto front. We also propose a bounded version which reduces the size of the messages using an adjustable parameter.

ACM Reference Format:

1 INTRODUCTION
A Distributed Constraint Optimization Problem (DCOP) [2, 6, 8] is a fundamental problem that can formalize various applications related to multi-agent cooperation. Multi-Objective DCOP (MO-DCOP) [1, 5] was proposed as an extension of DCOP with multiple objectives. In this problem, since trade-offs exist among objectives, the goal is to find the Pareto front which is the set of cost vectors obtained by Pareto optimal solutions.

Compared to DCOPs, there exist few algorithms for solving MO-DCOPs. MO-ADOPT [5] generalizes the ADOPT algorithm [6] to the multi-objective case and is the state-of-the-art complete algorithm. The Bounded Multi-Objective Max-Sum (B-MOMS) [1] extends the Bounded Max-Sum algorithm [10] and is the state-of-the-art approximation algorithm.

In this paper, we develop a novel complete algorithm for MO-DCOPs called Multi-Objective Distributed Pseudo-tree Optimization Procedure (MO-DPOP). This algorithm extends DPOP, the representative dynamic programming algorithm for DCOPs. We also provide an incomplete version of our algorithm that uses a bounding function to reduce the size of the messages exchanged between the agents, reducing the memory complexity while still guaranteeing to find a subset of the Pareto front.

2 PRELIMINARIES
2.1 Multi-objective Distributed Constraint Optimization Problem

Definition 1 (Multi-Objective DCOP). A Multi-Objective Distributed Constraint Optimization Problem (MO-DCOP) [1, 5] with \( m \) objectives is defined as a tuple \( \text{MO-DCOP} = (X, V, D, F) \) where \( X = \{x_1, \ldots, x_n\} \) is a set of agents, \( V = \{v_1, \ldots, v_m\} \) is a set of variables, \( D = \{D_1, \ldots, D_n\} \) is a set of domains, and \( F = (f_1, \ldots, f_m) \) is a set of multi-objective cost functions. The scope of function \( f_k \), denoted \( \text{var}(f_k) \subseteq V \) is the set of arguments of \( f_k \), indicating that the variables in \( \text{var}(f_k) \) share a constraint relation. A multi-objective cost function \( f_k \in F \) is then defined as \( f_k : \times_{v_i \in \text{var}(f_k)} D_i \rightarrow \mathbb{R}^m \). These functions produce cost vectors of the form \((u_1, \ldots, u_m)\) where \( u_i \) is the cost of objective \( i \).

The assignment of a variable \( v_i \in V \) with the value \( d_i \in D_i \) is denoted \( v_i \leftarrow d_i \). An assignment \( A = (v_1 \leftarrow d_1, \ldots, v_j \leftarrow d_j) \) is a set of assignments to different variables and we denote the set of variables included in an assignment \( \text{var}(A) \). \( A \) is said to be partial if \( \text{var}(A) \subset V \) and complete if \( \text{var}(A) = V \). The cost vector of a complete assignment is calculated by the objective function \( F(A) = \sum_{f_k \in F} f_k(A) \), where the sum of vectors is the usual component-by-component sum.

Optimal solutions of a MO-DCOP are characterized using the concept of Pareto optimality.

Definition 2 (Pareto Dominance). Given two vectors \( u = (u_1, \ldots, u_m) \) and \( w = (w_1, \ldots, w_m) \), we say that \( u \) dominates \( w \), denoted by \( u < w \), if it holds \( u_i \leq w_i \) for all objectives \( i \), and there exists at least one objective \( i' \), such that \( u_{i'} < w_{i'} \).

Definition 3 (Pareto optimal solution). For a MO-DCOP, a complete assignment \( A \) is a Pareto optimal solution iff there does not exist another complete assignment \( A' \) such that \( F(A') < F(A) \).

Solving a MO-DCOP consists in finding the set of cost vectors obtained by Pareto optimal solutions \( P^F \) (called Pareto front), and for each vector \( u \in P^F \) at least one complete assignment \( A \) such that \( F(A) = u \).

An MO-DCOP can be represented using a constraint graph which has a node for each variable and where an edge connects any two nodes whose variables appear in the scope of the same function. Considering this graph,

the corresponding pseudo-tree structure[12] is a rooted tree with the same nodes as the constraint graph (corresponding to agents) and with the property that nodes adjacent in the graph must belong to the same branch of the pseudo-tree. Such structure can be obtained using a depth-first traversal of the constraint graph.

After such structure is generated, each agent \( x_i \) is aware of its parent \( P_i \), its set of children \( CH_i \), and its set of pseudo-parents \( PP_i \). An agent \( x_j \) is a pseudo-parent of \( x_i \) if and only if it is an ancestor of \( P_i \) in the pseudo-tree and a neighbor of \( x_i \) in the constraint graph.

An important concept of pseudo-trees for the algorithm presented in this paper is the separator of an agent.

Definition 4 (Separator). In a pseudo-tree, the separator \( Sep \) of a node \( x_i \) is the set of all ancestors of \( x_i \) which are pseudo-parents of either \( x_i \) or one of its descendants:

\( \text{Sep} = \{ y_j \mid y_j \text{ is an ancestor of } x_i \text{ and } x_j \text{ is a pseudo-parent of } x_i \} \)
3 MO-DPOP

In this section, we present the Multi-Objective Distributed Pseudo-tree Optimization Algorithm (MO-DPOP). This algorithm is based on DPOP [8] and we only present here how we modify the UTIL and VALUE phases of DPOP.

3.1 UTIL propagation

Starting from the leaf agents. UTIL messages carrying the best cost vectors of each agent’s subproblem are sent up the pseudo-tree.

Definition 5 (UTIL message). A message UTIL\[i\] sent from agent x\[i\] to agent x\[j\] is a multi-dimensional matrix with one dimension for each variable in Sep\[i\] and we denote var(UTIL\[i\]) the set of variables considered by the message.

For an assignment A, var(A) \subseteq var(UTIL\[i\]), UTIL\[j\][A] is a matrix of dimension var(UTIL\[i\]) \times var(A) such that:

\[ UTIL\[j\][A] = \bigcup_{A': var(A') = var(UTIL\[i\]) \cap var(A)} UTIL\[i\][A \cup A'] \]

A message UTIL\[i\] expresses the best cost vectors that can be obtained by the sub-tree rooted at x\[i\] based on the values taken by the variables in the separator Sep\[i\]. UTIL messages are built from cost functions and we assume that a function f\[k\] : \times_{x\[i\] \in var(f\[k\])} D\[i\] → R\[m\] is represented as a matrix of dimension var(f\[k\]). When the root agent x\[r\] computes the message UTIL\[r\]null, it contains the best cost vectors that can be obtained by the whole tree, corresponding to the Pareto front of the problem.

To compute the UTIL message it will send to its parent, an agent x\[i\] has to join all messages received from its children as well as the cost functions it shares with its parent and pseudo-parents: \{f\[k\] \in F\mid var(f\[k\]) \subseteq \{x\[i\]\} \cup PP\[i\] \cup \{x\[j\]\}, x\[j\] \in var(f\[k\])\}.

Definition 6 (Join Operator in MO-DPOP). Joining two matrices M and M\['\] written M \[\oplus\] M\['\], produces a new matrix M\['\] such that var(M\['\]) = var(M) \cup var(M\['\]) and:

\[ VA, var(A) = var(M\['\]), M\['\][A] = \{u + u'| u \in M[A], u' \in M'[A]\} \]

Before sending an UTIL message, an agent x\[i\] projects variable x\[i\] out of the matrix, reducing its dimension by one and merging the content of some cells, which are filtered using Pareto dominance. For simplicity, we consider a function which takes a set of vectors U and returns the corresponding set of non-dominated vectors: ND(U) = \{u \in U\mid \exists u' \in U \text{ s.t. } u < u'\}.

Definition 7 (Projection Operator in MO-DPOP). Projecting variable x\[i\] out of matrix M, written M\[\perp\]x\[i\] and requiring x\[i\] \in var(M), is the projection of the matrix M along the x\[i\] dimension such that:

\[ VA, var(A) = var(M) \setminus \{x\[i\]\}, M\[\perp\]x\[i\][A] = ND(M[A]) \]

The root agent x\[r\] receives from each of its children an UTIL message of dimension \{x\[r]\} which, when joined, provide the Pareto front of the problem.

3.2 Limiting the Size of Messages

UTIL messages being of exponential space complexity, we propose a technique to limit their size using a bounding function B\[b\] which takes a set of vectors W and returns a set \( W' \subseteq W \text{ s.t. } |W'| \leq b \). This guarantees an upper bound for the size of the messages and, depending on the function used, can still guarantee to find Pareto optimal solution with our algorithm.

Property 1 (Maximum Bounded Message Size). If the subset \( U \) yielded by \( B \) is bounded in size (\( |U| \leq b \)), the maximum message size becomes bounded by the maximum separator size \( |Sep_{max}| \) with a space complexity in \( O(bm \times |D_{max}| \times |Sep_{max}|) \).

For example, bounding functions based on weighted-sums [4] or lexicographic orderings [3] guarantee to still find some Pareto optimal solutions when bounding the messages of MO-DPOP.

4 EXPERIMENTS

To evaluate MO-DPOP and its extension with bounded messages, we conducted experiments on the extended graph-coloring problem [1]. We compared MO-DPOP with the existing algorithms MO-ADOPT [5], and B-MOMS [1].

Algorithms were implemented in Java and experiments were carried on a 4.2 GHz 8 cores CPU, measuring the average simulated runtime [13] over 40 random instances.

Figure 1 shows the simulated runtime when varying the number of variables with problems of low density (0.01). We observe that our algorithm provides a significant improvement over the previous complete algorithm, allowing us to solve instances of up to 70 variables within 10s whereas MO-ADOPT cannot solve problems of 25 variables within that time.

5 CONCLUSION

In this paper, we developed a new complete algorithm for MO-DPOP and provided a technique to reduce the size of its messages. In our experiments, we showed that our complete algorithm outperforms the state-of-the-art complete MO-DPOP algorithm.

In future works, we will study additional ways to reduce the complexity of our algorithm by considering techniques such as the Mini-Bucket Elimination [11] or the Memory-Bounded DPOP [9] and p-reduced graph technique [7].
REFERENCES


