Fundamenta Informaticae 102 (2010) 263–286
DOI 10.3233/FI-2010-307
IOS Press

Cluster Tree Elimination for Distributed Constraint Optimization with Quality Guarantees

Ismel Brito[*], **Pedro Meseguer**[†]
IIIA, Artificial Intelligence Research Institute
CSIC, Spanish National Research Council
Campus UAB, 08193 Bellaterra, Spain
ismel@iiia.csic.es; pedro@iiia.csic.es

Abstract. Some distributed constraint optimization algorithms use a linear number of messages in the number of agents, but of exponential size. This is often the main limitation for their practical applicability. Here we present some distributed algorithms for these problems when they are arranged in a tree of agents. The exact algorithm, DCTE, computes the optimal solution but requires messages of size $exp(s)$, where s is a structural parameter. Its approximate version, DMCTE(r), requires smaller messages of size $exp(r)$, $r < s$, at the cost of computing approximate solutions. It provides a cost interval that bounds the error of the approximation. Using the technique of cost function filtering, we obtain DMCTEf(r). Combining cost function filtering with bound reasoning, we propose DIMCTEf, an algorithm based on repeated executions of DMCTEf(r) with increasing r. DIMCTEf uses messages of previous iterations to decrease the size of messages in the current iteration, which allows to alleviate their high size. We provide evidences of the benefits of our approach on two benchmarks.

Keywords: Distributed constraint optimization, distributed cluster-tree elimination, function filtering.

1. Introduction

In the last years, there is an increasing interest to solve constraint problems in a distributed form. This happens when several agents, which are related by constraints, look for a global consistent assignment

[*] This work has been partially supported by the project TIN2009-13591-C02-02.
[†] Address for correspondence: IIIA, CSIC, Campus UAB 08193 Bellaterra, Spain

satisfying all constraints (or, in a different version, as many constraints as possible). Typically, the whole problem can be seen as distributed constraint reasoning, where each agent owns a part of the instance but no agent knows the whole instance. Agents want to achieve a global consistent solution without joining all information into a single agent (there are several reasons for that: different formats, privacy requirements, etc.). New solving algorithms have been developed for this distributed model, where communication between agents is done by message passing. As examples of algorithms for distributed constraint reasoning, we mention ABT [25], ADOPT [14], DPOP [18]. As examples of problems requiring distributed solving, we mention –among many others– distributed meeting scheduling [23] and sensor networks [1].

Considering distributed constraint optimization, it is desirable that each agent, after some process, is able to compute locally the global optimum and an optimal assignment which is part of the global optimal assignment (although the global optimal assignment is known by no agent). The trivial approach, exchanging all information among all agents, is forbidden by the problem definition. However, this is not really needed. The contribution of this paper is to present some distributed algorithms that work on a special structure, a tree decomposition of the constraint network, able to compute the above mentioned goal. After their execution, it is enough to minimize a structure called *cluster* at each agent to obtain the global optimum and an assignment that is part of the global one. Interestingly, a tree decomposition can be computed in a distributed form.

Conceptually, DCTE is the simplest of the above mentioned algorithms. DCTE is able to achieve the exact solution of the distributed optimization problem, but requires messages of size $exp(s)$, where s is a structural parameter (the maximum separator size of the tree decomposition). DCTE suffers from the same drawback as DPOP [18], both use messages of exponential size (see section 7). This is often the main limitation for the practical applicability of these algorithms, so we try to decrease it. The approximated version of DCTE is DMCTE(r), that performs approximated solving using messages of size $exp(r)$, $r < s$. DMCTE(r) also computes a cost interval that bounds the error of the approximated solution with respect to the optimum cost. Applying the idea of cost function filtering we obtain DMCTEf(r), that performs the same kind of approximate solving using messages of shorter (or equal in the worst case) size. Combining cost function filtering with bound reasoning, we present the DIMCTEf algorithm. It is an iterative algorithm that executes DMCTEf(r) with increasing r, producing approximated solutions of increasing quality, and uses messages of previous iterations to decrease the size of messages at the current iteration. It keeps global lower and upper bounds during iterations, and it is able to stop execution when these bounds are close enough, according to user specifications. We have implemented and tested these algorithms on two different benchmarks. Experimental results show that DIMCTEf causes substantial decrements in largest message size and total data exchanged with respect to DCTE. We describe in detail a running example, which shows the benefits of our approach in message size. In some cases we are trading memory (shorter message size) for time, since DIMCTEf may require more computation than the exact algorithm. These algorithms can be seen as dynamic programming in a distributed context. We have taken inspiration from dynamic programming methods developed for the centralized case, adapted to a distributed context where each agent only knows its position in the tree decomposition, its own variables and constraints.

Throughout this paper it is assumed the existence of a tree decomposition where the problem instance is arranged. In the distributed case, there are algorithms able to build a tree decomposition in a distributed form (for example, the ERP algorithm [15]).

The paper is organized as follows. In section 2, we provide a precise definition of the problems we consider, and we present the basic solving algorithm for the centralized case working on a tree decomposition. In section 3, we describe its extension to the distributed case, the DCTE algorithm able to compute the exact solution but requiring messages of size $exp(s)$. In section 4, we describe DMCTE(r), that computes approximate solutions using messages of size $exp(r)$, $r < s$. The idea of function filtering appears in section 5, with the DIMCTEf algorithm. In section 6, we provide experimental results on two benchmarks. We summarize related approaches in section 7. Finally, we give a summary and a discussion in section 8. A running example along sections 3, 4 and 5 shows how algorithms work on the same instance.

2. Preliminaries

In a centralized setting, a *Constraint Optimization Problem* (COP) involves a finite set of variables, each one taking a value in a finite domain. Variables are related by cost functions that specify the cost of value tuples on some variable subsets. Costs are positive natural numbers (including zero and ∞). Formally, a finite COP is (X, D, C) where,

- $X = \{x_1, \ldots, x_n\}$ is a set of n variables;
- $D = \{D(x_1), \ldots, D(x_n)\}$ is a collection of finite domains; $D(x_i)$ is the initial set of x_i possible values;
- C is a set of cost functions; $f_i \in C$ on the ordered set of variables $var(f_i) = (x_{i_1}, \ldots, x_{i_{r(i)}})$ specifies the cost of every combination of values of $var(f_i)$, that is, $f_i : \prod_{j=i_1}^{i_{r_i}} D(x_j) \mapsto N^+$. The arity of f_i is $|var(f_i)|$.

The *overall cost* of a complete tuple (involving all variables) is the addition of all individual cost functions on that particular tuple. A *solution* is a complete tuple with acceptable overall cost, and it is *optimal* if its overall cost is minimal.

2.1. Distributed WCSP

Previous COP definition does not make explicit that there is an upper bound in the cost of acceptable value tuples, so those value tuples whose cost exceeds this upper bound can be safely removed. COP definition is refined to produce the so called *Weighted Constraint Satisfaction Problem* (WCSP). A WCSP is defined as a four tuple $(X, D, C, S(k))$, where X, D and C are as in the previous definition, and $S(k)$ is a valuation structure [9]. While in COPs, a cost function maps value combinations into the set of natural numbers, in a WCSP a cost function maps value combinations into a special set $\{0, 1, ..., k\}$. That is, $f_i : \prod_{j=i_1}^{i_{r_i}} D(x_j) \mapsto \{0, 1, ..., k\}$. Costs are elements of the set $\{0, 1, ..., k\}$, where 0 is the minimum cost and k is the minimum unacceptable cost. Costs lower than k are acceptable, while costs higher or equal to k are equally unacceptable. Costs are combined with $\oplus$: $a \oplus b = min\{a + b, k\}$, meaning that if the addition of two costs exceeds k, it automatically equals k. Costs are totally ordered with the standard order in naturals. We store cost function f as a set S_f containing all pairs $(t, f(t))$ with cost less than k. The *size* of f, denoted $|f|$, is the cardinal of S_f. This approach includes purely satisfaction instances, where tuples are either permitted or forbidden. A permitted tuple costs 0, a forbidden tuple costs 1, and k is 1. For further details on the relation between COPs and WCSPs see [9, 13].

We extend this definition to a distributed context. A *Distributed Weighted Constraint Satisfaction Problem* (DWCSP), is a WCSP where variables domains and cost functions are distributed among automated agents. Formally, a *cost-function-based* DWCSP is a 6-tuple (X, D, C, $S(k)$, A, β), where X, D, C and $S(k)$ define a WCSP, A is a set of p agents and β maps each cost function to one agent. Here we assume the DWCSP model: it is a refined version of distributed constraint optimization, where the notion of unacceptable cost is explicitly handled.

Next, some terminology used in the paper. An *assignment or tuple* t_S with scope S is an ordered sequence of values, each corresponding to a variable of $S \subseteq X$. The *projection* of t_S on a subset of variables $T \subseteq S$, written $t_S[T]$, is formed from t_S removing the values of variables that do not appear in T. This idea can be extended to cost functions: the projection of f on $T \subset var(f)$, is a new cost function $f[T]$ formed by the tuples of f removing the values of variables that do not appear in T, removing duplicates and keeping the minimum cost of the original tuples in f. The *cost* of a tuple t_X (involving all variables) is $\oplus_{f \in C} f(t_X)$, that is, the addition of the individual cost functions evaluated on t_X (implicitly, it is assumed that, for each $f \in C, f(t_X) = f(t_X[var(f)])$). A *solution* is a tuple with cost lower than k. A solution is *optimal* if its cost is minimal. The *join* of two tuples t_S and t'_T, written $t_S \cdot t'_T$, as a new tuple with scope $S \cup T$, formed by the values appearing in t_S and t'_T. This join is only defined when common variables have the same values in both tuples. *Summing* two functions f and g is a new function $f + g$ with scope $var(f) \cup var(g)$ and $\forall t \in \prod_{x_i \in var(f)} D(x_i)$, $\forall t' \in \prod_{x_j \in var(g)} D(x_j)$ such that $t \cdot t'$ is defined, $(f + g)(t \cdot t') = f(t) \oplus g(t')$. We say that function g is a *lower bound* of f, denoted $g \leq f$, if $var(g) \subseteq var(f)$ and for all possible tuples t of f, $g(t) \leq f(t)$. A set of functions G is a *lower bound* of f iff $(\sum_{g \in G} g) \leq f$. It is easy to check that for any $f, Y \subset var(f)$, $f[Y]$ is a lower bound of f, and $\sum_{f \in F} f[Y] \leq (\sum_{f \in F} f)[Y]$.

2.2. Cluster Tree Elimination

Centralized WCSPs can be solved using tree decomposition methods. A *tree decomposition* (TD) of a WCSP $\langle X, D, C, S(k) \rangle$ is a triple $\langle T, \chi, \psi \rangle$, where $T = \langle V, E \rangle$ is a tree, χ and ψ are labeling functions which associate with each node $v \in V$ two sets, $\chi(v) \subseteq X$ and $\psi(v) \subseteq C$ such that

- for each function $f \in C$, there is exactly one node $v \in V$ such that $f \in \psi(v)$; in addition, $var(f) \subseteq \chi(v)$;
- for each variable $x \in X$, the set $\{v \in V | x \in \chi(v)\}$ induces a connected subtree of T.

Its *tree-width* is $tw = max_{v \in V} |\chi(v)|$. If u and v are adjacent nodes, its *separator* is $sep(u, v) = \chi(u) \cap \chi(v)$ [5]. There are several methods to compute a TD (for a summary see [2]). Finding the TD with the smallest tree width is NP-hard [5]. From now on, we will assume that a suitable TD exists for the WCSP instance.

Cluster-Tree (umbrella term for names such as join-tree or clique-tree clustering, used in different research areas) Elimination (CTE) is an algorithm that solves WCSP by sending messages along tree decomposition edges [6, 5] . Edge $(u, v) \in E$ has associated two CTE messages $m_{(u,v)}$, from u to v, and $m_{(v,u)}$, from v to u. $m_{(u,v)}$ is a function computed summing all functions in $\psi(v)$ with all incoming

```
procedure CTE($T = (V, E), \chi, \psi$)
1 for each $(u, v) \in E$ s.t. all $m_{(i,u)}, i \neq v$ have arrived do
2     $B \leftarrow \psi(u) \cup \{m_{(i,u)} \mid (i, u) \in E, i \neq v\}$;
3     $m_{(u,v)} \leftarrow (\sum_{f \in B} f)[sep(u, v)]$;
4     send $m_{(u,v)}$;
```

Figure 1. The CTE algorithm.

CTE messages except $m_{(v,u)}$ and projected on $sep(u, v)$. CTE appears in Figure 1. CTE complexity is time $O(d^{tw})$ and space $O(d^s)$, where d is the largest domain size and s is the maximum separator size.

CTE does more than simply solving a WCSP instance. After CTE execution (obviously CTE terminates), we define for node $v \in V$ the $cluster(v) = \{m_{i,v} | (i, v) \in E\} \cup \psi(v)$. Node v has enough information in $cluster(v)$ to be able to answer, at node level, different tasks that require knowledge of the whole constraint network [6]. For instance, the number of different solutions of variables in $\chi(v)$ with a particular global cost, or the global cost of a solution when a variable in $\chi(v)$ is forced to take a particular value. CTE has been proven to be correct [10]. If we want to solve the WCSP only, we could use the simpler Bucket Elimination algorithm [4], that sends cost functions up to the bucket tree (a special case of tree decomposition), and propagates value assignments down the bucket tree [5]. However, working with cost functions in both directions will alleviate its exponential memory complexity, as we will see next.

Mini-Cluster-Tree Elimination (MCTE(r)) approximates CTE [5]. If the number of variables in u is high, it may be impossible to compute $m_{(u,v)}$ due to memory limitations. MCTE(r) computes a lower bound by limiting to r the maximum arity of the functions sent in the messages. A MCTE(r) message, $M_{(u,v)}$, is a set of functions that approximate the corresponding CTE message $m_{(u,v)}$ ($M_{(u,v)} \leq m_{(u,v)}$). It is computed as $m_{(u,v)}$ but instead of summing all functions of set B (see Figure 1), it computes a partition $P = \{P_1, \ldots, P_q\}$ of B such that the arity of the sum of functions in every P_i does not exceed r. The MCTE(r) algorithm is obtained replacing line 3 of CTE by the following lines (where the projection is done on the variables of the separator that appear in the scope of the functions in the partition class),

```
3.1 $\{P_1, ..., P_q\} \leftarrow partition(B, r)$;
3.2 $M_{(u,v)} \leftarrow \{(\sum_{f \in P_i} f)[sep(u, v) \cap (\cup_{f \in P_i} var(f))] \mid i : 1...q\}$;
```

3. Distributed Cluster Tree Elimination

The CTE algorithm can be adapted to the distributed case, producing the *Distributed Cluster Tree Elimination* (DCTE) algorithm. We assume that the DWCSP instance (X, D, C, A, β) to solve is arranged in a rooted TD (T, ψ, χ). Each node of the TD represents a different agent, so we will use these terms interchangeably. Let us consider $self$, a generic agent. It owns a specific node in the tree: it knows its neighbors (parent and children), the separators with them, variables in $\chi(self)$ and cost functions in $\psi(self)$.

DCTE exchanges messages among agents. There are two message types: CF and SS. DCTE exchanges first CF messages, that contain cost functions, and second SS messages, to assure single assignments of variables in separators. DCTE processes CF messages as follows. When $self$ has received CF messages from all its neighbors except perhaps i, it performs the summation of the received cost functions in these CF messages (excluding cost function from i) with the cost functions of $\psi(self)$, producing a new cost function, which is projected on $sep(self, i)$ and sent to agent i (this also applies to agents at the leaves of the TD, which in fact are those that start sending CF messages). This process is repeated for all neighbors i. CF messages play the same role as function messages in centralized CTE. For each edge (i, j) in the tree (i and j are neighboring agents) there are two CF messages: one $CF(i, j)$ from i to j and other $CF(j, i)$ from j to i. As in the centralized case, we define $cluster(v) = \{CF(j, v) | \forall j \in neighbors(v)\} \cup \psi(v)$. When all CF messages have been exchanged for all edges in the tree, each agent v contains in $cluster(v)$ enough information to locally answer some tasks that require information from the whole constraint network. In particular, agent $self$ can solve to optimality the whole DWCSP instance by minimizing $cluster(self)$. It happens that the minimum cost of $cluster(self)$ is equal to the global optimum, as stated by the following theorem.

Theorem 3.1. After DCTE exchanges all CF messages, each agent v verifies

$$\sum_{g \in cluster(v)} g = min_{X-\chi(v)}(\sum_{f \in \cup\psi(u), u \in V} f)$$

Proof:
Direct application of CTE correctness from [10]. □

From this theorem, when $self$ minimizes $cluster(self)$, it finds exactly the global minimum of the sum of all initial functions $\sum_{f \in \cup\psi(u), u \in V} f$. As a consequence, when $self$ builds the tuple that minimizes $cluster(self)$, it will be part of a global optimal assignment (if O is such assignment, $self$ will find $O[\chi(self)]$). After exchanging all CF messages, if there is a single global optimal assignment, then any agent will compute an assignment that is part of it by minimizing its cluster. But it may happen that several global optimal assignments exist, all sharing the global optimum cost. Let us assume that s_1 and s_2 are global optimal assignments of a DWCSP instance distributed between agents a_1 and a_2. If a_1 finds $s_1[\chi(a_1)]$ when minimizing $cluster(a_1)$, and a_2 finds $s_2[\chi(a_2)]$ when minimizing $cluster(a_2)$, it may happen that each agent will assign different values to variables in $sep(a_1, a_2)$. To assure that a_1 and a_2 perform the same assignments to variables in $sep(a_1, a_2)$, after CF messages DCTE exchanges SS messages. The agent i at the root of the TD minimizes $cluster(i)$ and sends a SS message to each child j, with the values of variables in $sep(i, j)$. When j receives such message, it minimizes $cluster(j)$ keeping unchanged the values of variables in $sep(i, j)$. Because Theorem 3.1, this assignment exists and it is globally optimal. Then, j repeats the process, which ends when SS messages reach tree leaves.

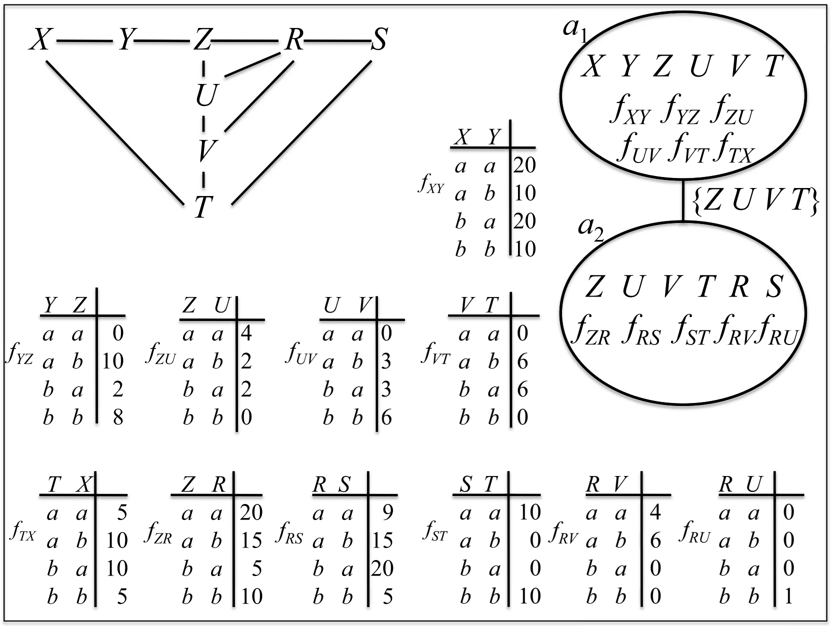
DCTE algorithm appears in Figure 2. It is executed on every agent, taking as input the TD (T, χ, ψ) and the agent identity ($self$). When DCTE terminates, $self$ knows the optimum cost and the value tuple of variables in $\chi(self)$ causing that cost. This value tuple is coherent with values of variables taken by other agents. The main procedure is `DCTE`, which works as follows. If $self$ has a single neighbor, j, $self$ does not have to wait for any incoming cost function. So the `ComputeSendFunction` procedure is called, summing all functions in $\psi(self)$, projecting the result on $sep(self, j)$ and sending the final result to j. Next, there is a loop that processes CF messages and checks the end loop condition (if $self$ has received/sent a cost function from/to each neighbor). A CF message is processed by the `NewCostFunction` procedure, that records the received cost function. If this cost function allows for

```
procedure DCTE($T, \chi, \psi$)
  if $neighbors(self) = \{j\}$ then ComputeSendFunction($self, j$);
  while $\neg$ (received and sent one $CF$ msg per neighbor) do
    $msg \leftarrow$ getMsg();
    if $msg.type = CF$ then NewCostFunction($msg$);
  PropagateSolution($T, \chi, \psi$);

procedure NewCostFunction($msg$)
  $function[msg.sender] \leftarrow msg.function$;
  for each $j \in neighbors(self)$ s.t. $self$ has not sent $CF$ to $j$ do
    if $self$ received $CF$ from all $i \in neighbors(self), i \neq j$ then
      ComputeSendFunction($self, j$);

procedure ComputeSendFunction($self, dest$)
  $Function \leftarrow \sum_{i \in neighbors(self), i \neq dest} function[i] + \sum_{f \in \psi(self)} f$;
  sendMsg($CF, self, dest, Function[sep(self, dest)]$);

procedure PropagateSolution($T, \chi, \psi$)
  if $self = root(T)$ then
    ComputeSolution($\emptyset$);
    for each $j \in children(self)$ do SendSolutionSeparator($self, j$);
  else
    $msg \leftarrow$ getMsg();
    if $msg.type = SS$ then
      ComputeSolution($msg.solsep$);
      for each $j \in children(self)$ do SendSolutionSeparator($self, j$);

procedure SendSolutionSeparator($self, dest$)
  sendMsg($SS, self, dest, \{sol[x] \mid x \in sep(self, dest)\}$);

procedure ComputeSolution($vars$)
  compute $sol$ minimizing $cluster(self)$, but keeping unchanged in $sol$ the values of variables passed in $vars$;
```

Figure 2. The Distributed CTE algorithm.

computing a new cost function to be sent to another neighbor, it is done by `ComputeSendFunction`. When execution exits the loop, if $self$ is at the root of the TD, it minimizes its cluster producing sol, which is propagated downwards the TD via SS messages. Otherwise, $self$ waits for a SS message from its parent. Upon reception, $self$ minimizes its cluster, keeping unchanged the values received in that SS message for the variables in the separator, and informs its children. Execution ends.

As an example, let us consider the problem instance depicted in Figure 3. There are two agents a_1 and a_2, each executing DCTE. Agent a_1 computes function $f_1 \leftarrow f_{XY} + f_{YZ} + f_{ZU} + f_{UV} + f_{VT} + f_{TX}$, projects it on $sep(a_1, a_2)$ and sends the result, $f_2 = f_1[ZUVT]$, to a_2 in a CF message. Analogously, agent a_2 computes function $f_3 \leftarrow f_{ZR} + f_{RS} + f_{ST} + f_{RV} + f_{RU}$, projects it on $sep(a_1, a_2)$ and sends the result, $f_4 = f_3[ZUVT]$, to a_1 in a CF message (cost functions f_1, f_2, f_3, f_4 appear in Figure 4). Then, $cluster(a_1) = \{f_{XY}, f_{YZ}, f_{ZU}, f_{UV}, f_{VT}, f_{TX}, f_4\}$, and $cluster(a_2) = \{f_{ZR}, f_{RS}, f_{ST}, f_{RV}, f_{RU}, f_2\}$. In a_1, the minimum of its cluster is 40, with the assignment $XYZUVT \leftarrow abbaaa$. In a_2, the minimum of its cluster is again 40, with the assignment $SRZUVT \leftarrow bbbaaa$. There


X
Y
Z
R
S
U
V
T
a_1
X Y Z U V T
f_{XY} f_{YZ} f_{ZU}
f_{UV} f_{VT} f_{TX}
{Z U V T}
a_2
Z U V T R S
f_{ZR} f_{RS} f_{ST} f_{RV} f_{RU}

f_{XY}

X	Y	
a	a	20
a	b	10
b	a	20
b	b	10

f_{YZ}

Y	Z	
a	a	0
a	b	10
b	a	2
b	b	8

f_{ZU}

Z	U	
a	a	4
a	b	2
b	a	2
b	b	0

f_{UV}

U	V	
a	a	0
a	b	3
b	a	3
b	b	6

f_{VT}

V	T	
a	a	0
a	b	6
b	a	6
b	b	0

f_{TX}

T	X	
a	a	5
a	b	10
b	a	10
b	b	5

f_{ZR}

Z	R	
a	a	20
a	b	15
b	a	5
b	b	10

f_{RS}

R	S	
a	a	9
a	b	15
b	a	20
b	b	5

f_{ST}

S	T	
a	a	10
a	b	0
b	a	0
b	b	10

f_{RV}

R	V	
a	a	4
a	b	6
b	a	0
b	b	0

f_{RU}

R	U	
a	a	0
a	b	0
b	a	0
b	b	1

Figure 3. Example instance, and a tree decomposition. The separator between a_1 and a_2 is $\{Z\ U\ V\ T\}$.

is one global optimal assignment, so SS exchange causes no changes: each agent keeps its optimum and assignment. DCTE execution ends.

4. Distributed Mini-Cluster Tree Elimination

DCTE can be easily modified to produce its approximated version, the *Distributed Mini-Cluster Tree Elimination* (DMCTE(r)) algorithm. It includes a new parameter, r, the maximum arity of the cost functions exchanged between neighbors. While DCTE adds all cost functions of an agent and sends the projection on the separator, exchanging messages with a single cost function of size d^s (d is the domain size and s is the separator size), DMCTE(r) limits by r the arity of the exchanged functions, although several functions may appear in a CF message.

DMCTE(r) uses three message types: CF, SS and BB. CF and SS messages have the same meaning than in DCTE, while the role of BB messages is explained below. DMCTE(r) exchanges first CF messages, second SS messages and finally BB messages. In DMCTE(r), CF messages contain approximations of the exact cost functions used by DCTE, and its communication schema is the same as in DCTE. CF messages are computed as follows. When computing the function to be sent from $self$ to $dest$, let B be the set of functions received from all neighbors but $dest$, union the set of functions $\psi(self)$. B is partitioned in $\{P_1, \ldots, P_q\}$ in such a way that the arity of the function resulting from the addition of all functions of each class P_i, projected on the variables $sep(self, dest) \cap (\cup_{g \in P_i} var(g))$, does not exceed r (this is different from the criterion used in the centralized case, where the partition is

f_1:

X	Y	Z	U	V	T	
a/b	a	a	a	a	a	29 / 34
a/b	a	a	a	a	b	40 / 35
a/b	a	a	a	b	a	38 / 43
a/b	a	a	a	b	b	37 / 32
a/b	a	a	b	a	a	30 / 35
a/b	a	a	b	a	b	41 / 36
a/b	a	a	b	b	a	39 / 44
a/b	a	a	b	b	b	38 / 33
a/b	a	b	a	a	a	37 / 42
a/b	a	b	a	a	b	48 / 36
a/b	a	b	a	b	a	46 / 51
a/b	a	b	a	b	b	45 / 40
a/b	a	b	b	a	a	38 / 43
a/b	a	b	b	a	b	49 / 44
a/b	a	b	b	b	a	47 / 52
a/b	a	b	b	b	b	46 / 41
a/b	b	a	a	a	a	21 / 26
a/b	b	a	a	a	b	32 / 27
a/b	b	a	a	b	a	30 / 35
a/b	b	a	a	b	b	20 / 24
a/b	b	a	b	a	a	23 / 27
a/b	b	a	b	a	b	33 / 28
a/b	b	a	b	b	a	31 / 36
a/b	b	a	b	b	b	30 / 25
a/b	b	b	a	a	a	25 / 30
a/b	b	b	a	a	b	36 / 31
a/b	b	b	a	b	a	34 / 39
a/b	b	b	a	b	b	33 / 28
a/b	b	b	b	a	a	26 / 31
a/b	b	b	b	a	b	37 / 32
a/b	b	b	b	b	a	35 / 40
a/b	b	b	b	b	b	34 / 29

f_3:

S	R	Z	U	V	T	
a/b	a	a	a	a	a	43 / 39
a/b	a	a	a	a	b	33 / 49
a/b	a	a	a	b	a	45 / 41
a/b	a	a	a	b	b	35 / 51
a/b	a	a	b	a	a	43 / 39
a/b	a	a	b	a	b	33 / 49
a/b	a	a	b	b	a	45 / 41
a/b	a	a	b	b	b	35 / 51
a/b	a	b	a	a	a	28 / 24
a/b	a	b	a	a	b	18 / 34
a/b	a	b	a	b	a	30 / 26
a/b	a	b	a	b	b	20 / 36
a/b	a	b	b	a	a	28 / 24
a/b	a	b	b	a	b	18 / 34
a/b	a	b	b	b	a	30 / 26
a/b	a	b	b	b	b	20 / 36
a/b	b	a	a	a	a	45 / 20
a/b	b	a	a	a	b	35 / 30
a/b	b	a	a	b	a	45 / 20
a/b	b	a	a	b	b	35 / 30
a/b	b	a	b	a	a	46 / 21
a/b	b	a	b	a	b	36 / 31
a/b	b	a	b	b	a	46 / 21
a/b	b	a	b	b	b	36 / 31
a/b	b	b	a	a	a	40 / 15
a/b	b	b	a	a	b	30 / 25
a/b	b	b	a	b	a	40 / 15
a/b	b	b	a	b	b	30 / 25
a/b	b	b	b	a	a	41 / 16
a/b	b	b	b	a	b	31 / 26
a/b	b	b	b	b	a	41 / 16
a/b	b	b	b	b	b	31 / 26

f_2:

Z	U	V	T	
a	a	a	a	21
a	a	a	b	27
a	a	b	a	30
a	a	b	b	24
a	b	a	a	22
a	b	a	b	28
a	b	b	a	31
a	b	b	b	25
b	a	a	a	25
b	a	a	b	31
b	a	b	a	34
b	a	b	b	28
b	b	a	a	26
b	b	a	b	32
b	b	b	a	35
b	b	b	b	29

f_4:

Z	U	V	T	
a	a	a	a	20
a	a	a	b	30
a	a	b	a	20
a	a	b	b	30
a	b	a	a	21
a	b	a	b	31
a	b	b	a	21
a	b	b	b	31
b	a	a	a	15
b	a	a	b	18
b	a	b	a	15
b	a	b	b	20
b	b	a	a	16
b	b	a	b	18
b	b	b	a	16
b	b	b	b	20

Figure 4. Functions computed by DCTE in the example.

done to obtain functions of up to arity r as result of the addition of all functions at each class). Functions in each partition class are added, and the result is projected on $sep(self, dest) \cap (\cup_{g \in P_i} var(g))$. With all the functions built in this form (one per partition class), we construct a CF message, which is sent to $dest$. So CF messages do not contain a single cost function (as in DCTE) but a set of cost functions, each of arity lower or equal to r. When all CF messages of DMCTE(r) have been exchanged, there is no guarantee that the minimum of $cluster(self)$ will be the global optimum, as stated by the following theorem.

Theorem 4.1. After DMCTE(r) exchanges all CF messages, each agent v verifies

$$\sum_{g \in cluster(v)} g \leq min_{X-\chi(v)} (\sum_{f \in \cup\psi(u), u \in V} f)$$

Proof:
Direct application of MCTE correctness from [6]. □

Different agents may compute different minimum costs when minimizing their clusters, and these values are lower bounds of the exact cost.

After CF messages, DMCTE(r) needs to exchange some extra information. To assure that there is a single value for variables in separators, DMCTE(r) follows the same strategy as DCTE using SS messages: the root minimizes its cluster and sends an SS message to its children with the values of the variables in separators. When a child receives such a message, it minimizes its cluster keeping unchanged the received values, and it repeats the process with its children. Differently from DCTE, there is not guarantee that the minima computed this way are globally optimal.

After SS messages, all agents have agreed on a global assignment of variables in separators. This means that all agents have agreed on a global assignment, of which the values of variables in separators are known to more than one agent (the other values are known by their owning agent). The cost of this global assignment is an upper bound of the global optimum (the cost of any global assignment is an upper bound of the global optimum). On the other hand, $self$ minimizes $cluster(self)$ without any restriction on the values of variables in separators, obtaining a global lower bound lb_{self} (by Theorem 4.1). We take the maximum among all agents as lower bound of the instance $lb = max_{v \in V}\{lb_v\}$. To compute these bounds, DMCTE(r) uses BB messages with the same communication schema as CF messages. A BB message from u to v contains two parts: an upper bound of the cost of the global optimum in the subtree rooted at u that does not include v, and a global lower bound. When $self$ has received BB messages from all its neighbors except perhaps i, it adds the received upper bounds (excluding upper bound from i) with the cost of $\psi(self)$ on sol, producing a new upper bound. As lower bound, it takes the maximum between its lower bound and the lower bound contained in the message. A new BB message is formed containing these new upper and lower bounds, which is sent to agent i.

Summarizing, DMCTE(r) uses three message types: CF, SS and BB. CF and BB messages follow the same synchronous communication pattern: for each tree edge (u, v), there are two messages of each type associated with it, one from u to v and other from v to u. Agent u can send a message to v when messages of the same type from all neighbors but v have been received. Agents having only one neighbor start the process, sending its corresponding message to that neighbor. SS messages follow the tree structure, from the root to leaves. There is one message per arc, and each agent has to wait for the SS message from its parent to send SS messages to its children. Each message type has the following meaning:

- CF: cost function messages. They contain a list of cost functions of arity lower or equal to r, that are approximations of the single cost function sent by DCTE. They communicate as in DCTE.
- SS: solution separator messages. They contain the values of variables in the separator between two adjacent agents in the TD. When an agent receives an SS message from its parent, it sends a SS message to each child.

- BB: bound messages. Once agents agreed on a global assignment (with single values for variables in separators), agents exchange partial upper bounds on the cost of this assignment and global lower bounds via BB messages. This is a new message type with respect to DCTE.

When all message types have been exchanged, $self$ knows its part of a global assignment (the projection of a global assignment on $\chi(self)$) and the global cost of this assignment. This global solution may not be the optimal one, and its cost is an upper bound of the optimal cost.

DMCTE(r) appears in Figure 6, as a function that returns the pair (lower bound, upper bound) computed for a particular r (obviously, $r < s$ otherwise the exact DCTE applies). Incoming CF, SS and UB messages are processed by `NewCostFunctions`, `NewSolutionSeparator` and `NewBounds` procedures, that implement the processes described above. It is of interest to compare the `ComputeSendFunction` procedure, that performs the addition of cost functions limiting the resulting arity, with the corresponding procedure of DCTE (Figure 2) that performs the exact computation.

In the example of Figure 3, DMCTE($r = 3$) works as follows. Agent a_1 performs the partition $\{\{f_{YZ}, f_{ZU}, f_{UV}\}, \{f_{VT}, f_{XY}, f_{TX}\}\}$ and computes functions $g_1 = f_{YZ} + f_{ZU} + f_{UV}$ and $g_2 = f_{XY} + f_{TX} + f_{VT}$. It projects these functions on the corresponding variables, obtaining $g_3 = g_1[ZUV]$ and $g_4 = g_2[VT]$. It builds a CF message with g_3 and g_4, which sends to a_2. Analogously, a_2 partition is $\{\{f_{ZR}, f_{RU}, f_{RV}\}, \{f_{RS}, f_{ST}\}\}$. It computes $g_5 = f_{ZR}+f_{RU}+f_{RV}$ and $g_6 = f_{RS}+f_{ST}$, projects them on the corresponding variables, obtaining $g_7 = g_5[ZUV]$ and $g_8 = g_6[T]$. It builds a CF message with g_7 and g_8, which sends to a_1 (cost functions g_1 to g_8 appear in Figure 5). After reception, $cluster(a_1) = \{f_{XY}, f_{YZ}, f_{ZU}, f_{UV}, f_{VT}, f_{TX}, g_7, g_8\}$, and $cluster\ (a_2) = \{f_{ZR}, f_{RS}, f_{ST}, f_{RV}, f_{RU}, g_3, g_4\}$. The optimum of $cluster(a_1)$ is 40, with $XYZUVT \leftarrow abbaaa$, while the optimum of $cluster(a_2)$ is 39,

g_1:

Y	Z	U	V	
a/b	a	a	a	4 / 6
a/b	a	a	b	7 / 9
a/b	a	b	a	5 / 7
a/b	a	b	b	8 / 10
a/b	b	a	a	12 / 10
a/b	b	a	b	15 / 13
a/b	b	b	a	13 / 11
a/b	b	b	b	16 / 14

g_2:

X	Y	V	T	
a/b	a	a	a	25 / 30
a/b	a	a	b	36 / 31
a/b	a	b	a	31 / 36
a/b	a	b	b	30 / 25
a/b	b	a	a	15 / 20
a/b	b	a	b	26 / 21
a/b	b	b	a	21 / 26
a/b	b	b	b	20 / 21

g_3:

Z	U	V	
a	a	a	4
a	a	b	7
a	b	a	5
a	b	b	8
b	a	a	10
b	a	b	13
b	b	a	11
b	b	b	14

g_4:

V	T	
a	a	15
a	b	21
b	a	21
b	b	20

g_5:

Z	R	U	V	
a/b	a	a	a	20 / 5
a/b	a	a	b	23 / 8
a/b	a	b	a	23 / 9
a/b	a	b	b	26 / 12
a/b	b	a	a	15 / 10
a/b	b	a	b	18 / 13
a/b	b	b	a	18 / 14
a/b	b	b	b	21 / 17

g_6:

R	S	T	
a	a	a	19
a	a	b	9
a	b	a	15
a	b	b	25
b	a	a	30
b	a	b	20
b	b	a	5
b	b	b	15

g_7:

Z	U	V	
a	a	a	15
a	a	b	18
a	b	a	18
a	b	b	21
b	a	a	5
b	a	b	8
b	b	a	9
b	b	b	12

g_8:

T	
a	5
b	9

Figure 5. Functions computed by DMCTE($r = 3$) in the example.

```
function DMCTE($T, \chi, \psi, r$)
  $thereIsSol \leftarrow$ false;
  if $neighbors(self) = \{j\}$ then ComputeSendFunction($self, j, r$);
  while $\neg$ (received-sent one $CF$ per neighbor) $\wedge$ (received one $SS$ from parent) $\wedge$
        (sent one $SS$ per child) $\wedge$ (received-sent one $BB$ per neighbor) do
    $msg \leftarrow$ getMsg();
    switch ($msg.type$)
    $CF$: NewCostFunction($msg, r$); $SS$: NewSolutionSeparator($msg$); $BB$: NewBounds($msg$);
  return ($LB, \sum_{j \in neighbors(self)} ub[j] + \sum_{f \in \psi(self)} f(sol)$)

procedure NewCostFunction($msg, r$)
  $functions[msg.sender] \leftarrow msg.functions$;
  for each $j \in neighbors(self)$ s.t. $self$ has not sent $CF$ to $j$ do
    if $self$ has received $CF$ msg from all $i \in neighbors(self), i \neq j$ then
      ComputeSendFunction($self, j, r$);
  if (by first time, received and sent one $CF$ msg/neighbor) then
    if $self = root(T)$ then
      ComputeSolution($\emptyset$);
      for each $j \in children(self)$ do SendSolutionSeparator($self, j$);

procedure NewSolutionSeparator($msg$)
  $sol \leftarrow$ ComputeSolution($msg.solsep$);
  for each $j \in children(self)$ do SendSolutionSeparator($self, j$);
  if $neighbors(self) = \{k\}$ then
    $lb \leftarrow$ minimum $cluster(self)$; ComputeSendBounds($self, k$);

procedure NewBounds($msg$)
  $ub[msg.sender] \leftarrow msg.upperBound$; $LB \leftarrow max\{msg.lowerBound, LB\}$;
  if $thereIsSol$ then
    for each $j \in neighbors(self)$ s.t. $self$ has not sent $BB$ to $j$ do
      if $self$ has received $BB$ msg from all $i \in neighbors(self), i \neq j$ then
        ComputeSendBounds($self, j$);

procedure ComputeSendFunction($self, dest, r$)
  $B \leftarrow \{functions[i] | i \in neighbors(self), i \neq dest\} \bigcup \psi(self)$;
  $\{P_1 ... P_q\} \leftarrow$ partition($B, r, sep(self, dest)$);
  sendMsg($CF, self, dest, \{(\sum_{g \in P_k} g)[sep(self, dest) \cap (\cup_{g \in P_k} var(g))] | k : 1...q\}$);

procedure SendSolutionSeparator($self, dest$)
  sendMsg($SS, self, dest, \{sol[x] \mid x \in sep(self, dest)\}$);

procedure ComputeSendBounds($self, dest$)
  $UB \leftarrow \sum_{j \in neighbors(self), j \neq dest} ub[j] + \sum_{f \in \psi(self)} f(sol)$;
  sendMsg($BB, self, dest, UB, LB$);

function ComputeSolution($vars$)
  $thereIsSol \leftarrow$ true;
  return assignment minimizing $cluster(self)$, keeping the values of $vars$;
```

Figure 6. The Distributed MCTE algorithm.

with $SRZUVT \leftarrow bbaaaa$ (observe that Z takes different values in these two local optima). Assuming that a_2 is the root of the tree, it imposes the values of variables in the separator $ZUVT \leftarrow aaaa$ with an SS message. a_1 computes the minimum in its cluster, keeping these values unchanged, obtaining $XYZUVT \leftarrow abaaaa$. Each agent i computes the cost of these assignments in its $\psi(i)$, obtaining costs of 21 and 20 for a_1 and a_2 respectively. These costs are exchanged using $BB(a_1 \rightarrow a_2, UB = 21, LB = 40)$ and $BB(a_2 \rightarrow a_1, UB = 20, LB = 39)$. DMCTE$(r = 3)$ returns the interval [40, 41] in both agents and ends.

5. Distributed Iterative Mini-Cluster Tree Elimination with Filtering

5.1. Cost Function Filtering

The idea of cost function filtering is a clever strategy to decrease the size of CF messages. It was introduced for the centralized case in [19]. The basic idea is to detect tuples that, although having acceptable cost when generated by an agent, they will always generate tuples with unacceptable cost when combined with cost functions coming from other agents. These initial tuples are removed before they are sent, decreasing the size of exchanged cost functions.

Imagine that we know that cost function f will be added (in the future) with cost function g, $var(g) \subseteq var(f)$, and we know that the set of functions G is a lower bound of g. We define the filtering of f from G, noted $\overline{f}^G$, as

$$\overline{f}^G(t) = \begin{cases} f(t) & if \quad (\bigoplus_{h \in G} h(t)) \oplus f(t) < UB \\ UB & otherwise \end{cases}$$

where UB is the upper bound on the maximum acceptable cost. A cost function f is defined by the set of pairs $(t, f(t))$ that do not reach the upper bound UB. The basic result that allows cost function filtering is stated next.

Theorem 5.1. Let f and g be two cost functions, $var(g) \subseteq var(f)$, and G a set of functions that is a lower bound of g. Filtering f with G before adding with g is equal to $f + g$,

$$f + g = \overline{f}^G + g$$

Proof:
Function f is

$$f = \{(t_1, f(t_1)) | t_1 \in P\} \cup \{(t_2, f(t_2)) | t_2 \in Q\}$$

where $P = \{t | t \in \prod_{x_i \in var(f)} D(x_i) \wedge (\bigoplus_{h \in G} \; h(t)) \oplus f(t) < UB\}$, $Q = \{t | t \in \prod_{x_i \in var(f)} D(x_i) \wedge (\bigoplus_{h \in G} \; h(t)) \oplus f(t) \geq UB\}$. Function $f + g$ is

$$f + g = \{(t, f(t) \oplus g(t))\} =$$

$$\{(t_1, f(t_1) \oplus g(t_1)) | t_1 \in P\} \cup \{(t_2, f(t_2) \oplus g(t_2)) | t_2 \in Q\}$$

but the second set is empty because $f(t_2) \oplus (\bigoplus_{h \in G} h(t_2)) \leq f(t_2) \oplus g(t_2)$, since G is a lower bound of g, so $f(t_2) \oplus g(t_2) \geq UB, \forall t_2 \in Q$. Then, $f + g$ is,

$$\{(t_1, f(t_1) \oplus g(t_1)) | t_1 \in P\}$$

which is exactly $\overline{f}^G + g$. □

How often do we know that function f will be added with function g? Let us consider agents a_u and a_v, that exchange messages $CF(u, v)$ and $CF(v, u)$. Upon reception, $CF(u, v)$ is included in $cluster(v)$, where all cost functions are added to compute the minimization. Therefore, any of the functions in $cluster(v)$ before arriving $CF(u, v)$ can filter $CF(u, v)$. A similar situation happens in agent u with message $CF(v, u)$.

The idea of cost function filtering can be integrated in DMCTE(r), producing the DMCTEf(r) algorithm whose only difference with original DMCTE(r) is that cost function summation is done with filtering, using the addfiltering procedure that appears in Figure 7, where a set of functions $\{f_1, ..., f_s\}$ is added filtered with a set of functions $\{g_1, ..., g_t\}$. In DMCTE(r), the set of cost functions in a CF message is a lower bound of the exact cost function (the one computed by DCTE), so $CF(v, u)$ can be used to filter the computation of $CF(u, v)$ and vice versa. This suggests a possible filter selection: before computing a $CF(u, v)$ message, check if the opposite message $CF(v, u)$ has arrived. If so, use it as filter for computing $CF(u, v)$, otherwise $CF(u, v)$ is computed without filtering.

Considering DMCTE(r), an interesting observation is that $CF(v, u)$ in iteration $r - 1$ is a lower bound of the exact cost function [19]. This suggests an iterative algorithm with increasing r, where CF messages of the previous iteration are used as filters of the current iteration. This new algorithm is *Distributed Iterative Mini-Cluster Tree Elimination with Filtering* (DIMCTEf). Obviously, the above mentioned idea can also be used here: at iteration r when computing $CF(u, v)$, check if $CF(v, u)$ has arrived; if so, use it as filter, otherwise use $CF(v, u)$ of iteration $r - 1$. In addition, the upper bound computed at the previous iteration is taken as the upper bound for the current iteration, used for function filtering.

In the example of Figure 3, DIMCTEf works as follows. To save space, we start with $r = 3, ub = \infty$. In the first iteration, this algorithm works exactly as DMCTE($r = 3$) of section 4. In the second iteration, $r = 4, ub = 41$. Agent a_1 computes f_1 filtering with g_7 and g_8, obtaining h_1 where tuples with cost $\geq ub$ are removed. It happens that h_1 has 3 tuples only, in contrast with the 64 tuples of f_1! The 61 missing tuples have been removed by the filtering effect: they generate tuples with costs higher than or equal to the upper bound. The projection of h_1 on the separator is h_2, which has 2 tuples. Analogously, a_2 computes f_2 filtering with g_3 and g_4, resulting h_3 which has 2 tuples only. Its projection on the separator

```
function addfiltering($\{f_1, ..., f_s\}, \{g_1, ..., g_t\}, ub$)
$result \leftarrow \emptyset$; $F \leftarrow \bigoplus_{i=1}^{s} f_i$; $G \leftarrow \bigoplus_{i=1}^{t} g_i$;
for each $t = t_1 \cdot ... \cdot t_s \cdot t'_1 \cdot ... \cdot t'_t$
  where $t_i = t[var(f_i)], t'_i = t[var(g_i)]$ do
  if $F(t_1 \cdot ... \cdot t_s) \oplus G(t'_1 \cdot ... \cdot t'_t) < ub$ then $result \leftarrow result \cup \{((t_1 \cdot ... \cdot t_s)F(t_1 \cdot ... \cdot t_s))\}$;
return $result$;
```

Figure 7. The addfiltering function.

is h_4, which has 2 tuples (functions h_1, h_2, h_3, h_4 appear in Figures 8 and 9). a_1 sends a CF message containing h_2 to a_2, which sends a CF message with h_4 to a_1. Upon reception, a_1 minimizes its cluster, obtaining 40 with the assignment $XYZUVT \leftarrow$ *abbaaa*. a_2 minimizes its cluster, obtaining 40 with the assignment $SRZUVT \leftarrow$ *bbbaaa*. a_2 sends a SS message to a_1 but this causes no change in the assignments of variables in the separator. Agents exchange their costs of their local optimal assignment, 25 and 15, achieving an upper bound of 40. Since $r = |sep(a_1, a_2)|$, the solution computed in this iteration is the exact solution. We have computed the exact solution exchanging functions of 2 tuples, instead of the complete functions of $2^4 = 16$ tuples.

5.2. Termination Conditions

While the basics of the DIMCTEf algorithm have been stated in the subsection 5.1, here we present in detail its termination conditions to produce a non-ambiguous description.

Looking for the optimum of a DWCSP instance arranged in a TD (T, ψ, χ), DIMCTEf iteratively executes DMCTEf(r) with increasing r, filtering CF messages of the current iteration with CF messages of the previous iteration. DIMCTEf terminates when one of the following conditions is satisfied:

1. $r = s$. When r is equal to the maximum separator size of the TD, the algorithm computes the exact solution as the UB at the end of the current iteration in the same conditions as DCTE, so it terminates. Since DIMCTEf is executed in every agent, the maximum separator size has to be known by each agent prior to the algorithm execution (for instance, agents may be informed of it after distributively compute the TD).

2. $UB = LB$. When the upper bound is equal to the lower bound, the optimum has been reached so the algorithm terminates (no matter the current value of r). Since LB is a true lower bound (Theorem 4.1), more iterations cannot improve it further. The upper bound UB is the global optimum and the assignment from which it was computed is an optimal global assignment.

3. Empty f. Imagine that in iteration r agent a_u computes an empty function f (a function where every tuple has been pruned), when adding all functions in class P_i. This means that allowed tuples of variables in P_i functions, when added with tuples coming from other cost functions, will have a cost greater than or equal to the upper bound. Some functions of P_i that arrived to a_u in CF messages may have some tuples pruned by other agents different from a_u. Since previously pruned tuples of variables in P_i functions had a cost greater than or equal to the upper bound, we can conclude that all tuples of variables in P_i functions reach or surpass the upper bound with P_i functions only. Obviously, when these tuples appear in a larger tuple (imagine a tuple formed by values of every variable), the cost of the larger tuple is greater than or equal to the upper bound. Therefore, in an empty function there is no tuple with cost lower than the current upper bound. In this case, the upper bound computed at the end of previous iteration is the global optimum and the assignment from which it was computed is an optimal global assignment.

While DCTE computes the exact cost of the optimal solution, DMCTEf(r) computes an interval $[lb, ub]$ that includes the optimum cost, where ub corresponds to the cost of a global assignment. If the user is willing to accept a solution with cost exceeding the optimum up to Δ units, termination condition 2 above is replaced by the condition $UB - \Delta \leq LB$, providing the assignment that produces the UB

	X	Y	Z	U	V	T	
	a/b	a	a	a	a	a	$29 + 15 + 5\geq ub$ / $34 + 15 +5\geq ub$
	a/b	a	a	a	a	b	$40 + 15 + 9\geq ub$ / $35 + 15 +9\geq ub$
	a/b	a	a	a	b	a	$38 + 18 + 5\geq ub$ / $43 + 18 +5\geq ub$
	a/b	a	a	a	b	b	$37 + 18 + 9\geq ub$ / $32 + 18 +9\geq ub$
	a/b	a	a	b	a	a	$30 + 18 + 5\geq ub$ / $35 + 18 +5\geq ub$
	a/b	a	a	b	a	b	$41 + 18 + 9\geq ub$ / $36 + 18 +9\geq ub$
	a/b	a	a	b	b	a	$39 + 21 + 5\geq ub$ / $44 + 21 +5\geq ub$
	a/b	a	a	b	b	b	$38 + 21 + 9\geq ub$ / $33 + 21 +9\geq ub$
	a/b	a	b	a	a	a	$37 + 5 + 5\geq ub$ / $42 + 5 +5\geq ub$
	a/b	a	b	a	a	b	$48 + 5 + 9\geq ub$ / $36 + 5 +9\geq ub$
	a/b	a	b	a	b	a	$46 + 8 + 5\geq ub$ / $51 + 8 +5\geq ub$
	a/b	a	b	a	b	b	$45 + 8 + 9\geq ub$ / $40 + 8 +9\geq ub$
	a/b	a	b	b	a	a	$38 + 9 + 5\geq ub$ / $43 + 9 +5\geq ub$
	a/b	a	b	b	a	b	$49 + 9 + 9\geq ub$ / $44 + 9 +9\geq ub$
	a/b	a	b	b	b	a	$47 + 12 + 5\geq ub$ / $52 + 12 +5\geq ub$
h_1:	a/b	a	b	b	b	b	$46 + 12 + 9\geq ub$ / $41 + 12 +9\geq ub$
	a/b	b	a	a	a	a	$21 + 15 + 5\geq ub$ / $26 + 15 +5\geq ub$
	a/b	b	a	a	a	b	$32 + 15 + 9\geq ub$ / $27 + 15 +9\geq ub$
	a/b	b	a	a	b	a	$30 + 18 + 5\geq ub$ / $35 + 18 +5\geq ub$
	a/b	b	a	a	b	b	$20 + 18 + 9\geq ub$ / $24 + 18 +9\geq ub$
	a/b	b	a	b	a	a	$23 + 18 + 5\geq ub$ / $27 + 18 +5\geq ub$
	a/b	b	a	b	a	b	$33 + 18 + 9\geq ub$ / $28 + 18 +9\geq ub$
	a/b	b	a	b	b	a	$31 + 21 + 5\geq ub$ / $36 + 21 +5\geq ub$
	a/b	b	a	b	b	b	$30 + 21 +9\geq ub$ / $25 + 21 +9\geq ub$
	a/b	b	b	a	a	a	25 + 5 + 5 / 30 + 5 +5
	a/b	b	b	a	a	b	$36 + 5 + 9\geq ub$ / $31 + 5 +9\geq ub$
	a/b	b	b	a	b	a	$34 + 8 + 5\geq ub$ / $39 + 8 +5\geq ub$
	a/b	b	b	a	b	b	$33 + 8 + 9\geq ub$ / $28 + 8 +9\geq ub$
	a/b	b	b	b	a	a	26 + 9 + 5 / $31 + 9 +5\geq ub$
	a/b	b	b	b	a	b	$37 + 9 + 9\geq ub$ / $32 + 9 +9\geq ub$
	a/b	b	b	b	b	a	$35 + 12 + 5\geq ub$ / $40 + 12 +5\geq ub$
	a/b	b	b	b	b	b	$34 + 12 + 9\geq ub$ / $29 + 12 +9\geq ub$

	S	R	Z	U	V	T	
	a/b	a	a	a	a	a	$43 + 4 + 15 \geq ub$ / $39 + 4 + 15 \geq ub$
	a/b	a	a	a	a	b	$33 + 4 + 21 \geq ub$ / $49 + 4 + 21 \geq ub$
	a/b	a	a	a	b	a	$45 + 7 + 21 \geq ub$ / $41 + 7 + 21 \geq ub$
	a/b	a	a	a	b	b	$35 + 7 + 20 \geq ub$ / $51 + 7 + 20 \geq ub$
	a/b	a	a	b	a	a	$43 + 5 + 15 \geq ub$ / $39 + 5 + 15 \geq ub$
	a/b	a	a	b	a	b	$33 + 5 + 21 \geq ub$ / $49 + 5 + 21 \geq ub$
	a/b	a	a	b	b	a	$45 + 8 + 21 \geq ub$ / $41 + 8 + 21 \geq ub$
	a/b	a	a	b	b	b	$35 + 8 + 20 \geq ub$ / $51 + 8 + 20 \geq ub$
	a/b	a	b	a	a	a	$28 + 10 + 15 \geq ub$ / $24 + 10 + 15 \geq ub$
	a/b	a	b	a	a	b	$18 + 10 + 21 \geq ub$ / $34 + 10 + 21 \geq ub$
	a/b	a	b	a	b	a	$30 + 13 + 21 \geq ub$ / $26 + 13 + 21 \geq ub$
	a/b	a	b	a	b	b	$20 + 13 + 20 \geq ub$ / $36 + 13 + 20 \geq ub$
	a/b	a	b	b	a	a	$28 + 11 + 15 \geq ub$ / $24 + 11 + 15 \geq ub$
	a/b	a	b	b	a	b	$18 + 11 + 21 \geq ub$ / $34 + 11 + 21 \geq ub$
	a/b	a	b	b	b	a	$30 + 14 + 21 \geq ub$ / $26 + 14 + 21 \geq ub$
h_3:	a/b	a	b	b	b	b	$20 + 14 + 20 \geq ub$ / $36 + 14 + 20 \geq ub$
	a/b	b	a	a	a	a	$45 + 4 + 15 \geq ub$ / 20 + 4 + 15
	a/b	b	a	a	a	b	$35 + 4 + 21 \geq ub$ / $30 + 4 + 21 \geq ub$
	a/b	b	a	a	b	a	$45 + 7 + 21 \geq ub$ / $20 + 7 + 21 \geq ub$
	a/b	b	a	a	b	b	$35 + 7 + 20 \geq ub$ / $30 + 7 + 20 \geq ub$
	a/b	b	a	b	a	a	$46 + 5 + 15 \geq ub$ / $21 + 5 + 15 \geq ub$
	a/b	b	a	b	a	b	$36 + 5 + 21 \geq ub$ / $31 + 5 + 21 \geq ub$
	a/b	b	a	b	b	a	$46 + 8 + 21 \geq ub$ / $21 + 8 + 21 \geq ub$
	a/b	b	a	b	b	b	$36 + 8 + 20 \geq ub$ / $31 + 8 + 20 \geq ub$
	a/b	b	b	a	a	a	$40 + 10 + 15 \geq ub$ / 15 + 10 + 15
	a/b	b	b	a	a	b	$30 + 10 + 21 \geq ub$ / $25 + 10 + 21 \geq ub$
	a/b	b	b	a	b	a	$40 + 13 + 21 \geq ub$ / $15 + 13 + 21 \geq ub$
	a/b	b	b	a	b	b	$30 + 13 + 20 \geq ub$ / $25 + 13 + 20 \geq ub$
	a/b	b	b	b	a	a	$41 + 11 + 15 \geq ub$ / $16 + 11 + 15 \geq ub$
	a/b	b	b	b	a	b	$31 + 11 + 21 \geq ub$ / $26 + 11 + 21 \geq ub$
	a/b	b	b	b	b	a	$41 + 14 + 21 \geq ub$ / $16 + 14 + 21 \geq ub$
	a/b	b	b	b	b	b	$31 + 14 + 20 \geq ub$ / $26 + 14 + 20 \geq ub$

Figure 8. Functions computed by DIMCTEf in the example.

	Z	U	V	T	
h_2:	b	a	a	a	25
	b	b	a	a	26

	Z	U	V	T	
h_4:	a	a	a	a	20
	b	a	a	a	15

Figure 9. Functions computed by DIMCTEf in the example (cont.).

as the solution. Effectively, if $UB - \Delta \leq LB$ the optimum cannot be at a distance higher than Δ (otherwise the LB will not be a true lower bound, something proved in Theorem 4.1). Therefore, this is a legal termination condition for DMCTEf(r). DIMCTEf computes a sequence of intervals $[lb_r, ub_r]$ one for each r. In most cases, this sequence will be monotonically shrinking (that is, the interval computed at iteration r will be included in the interval computed at iteration $r-1$). However, in some special cases this may not be the case. Since lb_r and ub_r are true lower and upper bounds, no matter the r value, we can take as LB the highest lb_r computed so far, and as UB the lowest ub_r. Then, the previous condition is a legal termination condition for this algorithm. It is worth noting that this approach includes *absolute* and *relative* distances to the optimum. While the previous description considers an absolute distance Δ, we can also specify the percentage δ over the optimum that we are willing to accept. At the end of each iteration we apply this percentage to UB, obtaining an absolute distance Δ, on which we can apply the termination condition previously discussed. DIMCTEf is an anytime algorithm, able to provide an increasingly better solution as r increases. If no more resources (CPU time, memory, bandwidth) are available, it can be stopped getting the last UB found as solution and taking $UB - LB$ as the error bound for this approximation.

Combining the use of functions exchanged in the previous iteration as function filters with the termination conditions explained above, we obtain the DIMCTEf algorithm that appears in Figure 10. We assume that the user specifies a δ percentage, stating the relative distance between the cost of an acceptable solution and the optimum cost.

6. Experimental Results

We tested DCTE and DIMCTEf on two benchmarks: distributed random problems and distributed meeting scheduling problems. Random problems have no structure, while meeting scheduling are structured problems. We generated random instances according to the following parameters: number of agents, number of variables, size of domains and number of unary and binary cost functions. We uniformly

```
function DIMCTEf($(T, \chi, \psi, \delta)$)
  for each $j \in neighbors(self)$ do $filter[j] \leftarrow \emptyset$;
  $UB \leftarrow \infty$; $\Delta \leftarrow 0$; $r \leftarrow 0$;
  repeat
    $r \leftarrow r + 1$;
    $[lb_r, ub_r] \leftarrow$ DMCTEf$(T, \chi, \psi, r, UB - \Delta)$;
    if not received an empty function then
      $LB \leftarrow max_k\{lb_k | k : 1, ..., r\}$; $UB \leftarrow min_k\{ub_k | k : 1, ..., r\}$; $\Delta \leftarrow \frac{UB \times \delta}{100}$;
      for each $j \in neighbors(self)$ do $filter[j] \leftarrow functions[j]$;
  until $r = s \vee UB - \Delta \leq LB \vee$ received an empty function;
  return $UB$;
```

Figure 10. The DIMCTEf algorithm.

spread variables and cost functions among agents and variables, respectively. We randomly filled out cost functions with costs taken from the natural interval $\{0, \dots, 9\}$. For that problem, a solution is to assign values to variables in such a way the overall cost be minimal.

We generated instances of the distributed meeting scheduling problem considering department hierarchies [11]. Each department consists of a set of people working on it, which have to participate in a set of meetings. For the distributed meeting scheduling problem, a solution is to schedule the meetings in such a way the overall cost be minimal according to the preferences that people have of meetings and time-slots on their own agendas. Every agent represents one person. An agent has multiple variables: one for the start time of each meeting the agent takes part in. Variable domains have 8 time-slots as values. All meetings last one time-slot. There exist two meeting types: internal meetings, involving people working on the same department, and external ones, involving people from different departments. Variables of an agent share mutual exclusion constraints and variables of all agents involved in the same meetings share equality constraints. Unary constraints represent agents' personal preferences. For all instances, the number of attendants for meetings is at most 4.

Experiments have been performed on the FRODO [17] simulation platform, designed for implementation and testing of distributed optimization algorithms. FRODO simulates a multiagent system where agents execute asynchronously. Each agent is simulated by a Java thread, and communicates with other agents via message passing. FRODO is publicly available at `http://liawww.epfl.ch/Research/sensornets/`. On top of FRODO, our algorithms have been implemented in Java, and executed on a laptop PC, with a 2Ghz CPU, with 1GB RAM.

Experimental results on distributed random instances appear in Figure 11. We provide the largest message size and the total data exchanged (both in Kbytes) for DCTE, the exact algorithm, and DIMCTEf for each iteration (until one of the termination conditions is satisfied). We consider two scenarios: one where we accept optimal solutions only ($\delta = 0\%$), and other where we accept solutions which are at most 5% distant from the optimum ($\delta = 5\%$). In addition, for each DIMCTEf iteration, we provide the returned interval $[lb, ub]$. When lb is higher or equal $ub(1 - \frac{\delta}{100})$, the computed solution is within δ distance from the optimum so DIMCTEf terminates.

Considering optimal solutions ($\delta = 0\%$), we provide largest message size and total data exchanged. Regarding largest message size, DIMCTEf causes a substantial improvement with respect to DCTE: DIMCTEf largest messages are from 28% to 90% shorter than the corresponding DCTE largest messages. In addition, most instances tested present savings higher than 80%. Regarding total data exchanged (for DIMCTEf, we compute a grand total adding the data exchanged at the executed iterations, to be compared with the total data exchanged by DCTE), we observe a similar picture: DIMCTEf causes a substantial improvement with respect to DCTE, with the only exception of instance E for which DIMCTEf exchanges 16% more data than DCTE. For all other instances, DIMCTEf exchanges from 37% to 90% less data than DCTE. We observe that savings in total data exchanged tends to be lower than savings in largest message size, due to the following reasons. First, data exchanged may include tables of dimensions lower than r (simply because the separator of a link is lower than r or as result of function partition). In tables with small dimensions, cost function filtering is not as effective as in tables of higher dimensions (tables with small dimensions are produced from adding few cost functions and the cost accumulated in their elements is often not enough for cost function filtering to prune), so the effect of cost function filtering is not uniform. Second, while DCTE exchanges complete tables, with all their elements, DIMCTEf exchanges partial tables which do not contain all their elements, due to the effect of

instance	A		B		C		D		E	
#agents	12		36		38		19		41	
#variables	35		90		100		66		137	
domain size	8		10		10		8		8	
#cost functions	38		80		100		80		162	
largest separator	3		3		3		4		4	
optimum cost	48		181		208		70		174	
Algorithm	Largest message	Total data	Largest message	Total data	Largest message	Total data	Largest message	Total data	Largest message	Total data
DCTE	2(48)	7.3	3.9(181)	12.6	3.9(208)	24.7	16(70)	53.6	16(174)	87.8
DIMCTEf($r = 2$)	0.3[47,48]	3.9	0.4[173,181]	6.3	0.4[204,208]	11.8	0.3[68,75]	9.8	0.3[160,181]	18.0
DIMCTEf($r = 3$)	*0.02[48,48]	*0.2	*0.1(empty f)	*1.1	*0.1(empty f)	*0.8	2.0[70,70]	6.2	2.0[174,176]	46.5
DIMCTEf($r = 4$)									*11.6[174,174]	*37.1
Savings $\delta = 0\%$	88%	43%	90%	41%	90%	49%	88%	70%	28%	-16%
Savings $\delta = 5\%$	88%	46%	90%	50%	90%	53%	88%	70%	88%	27%

instance	F		G		H		I		J		K	
#agents	26		53		101		77		142		10	
#variables	89		207		375		218		558		63	
domain size	8		8		8		10		8		5	
#cost functions	106		262		478		236		724		86	
largest separator	5		5		5		5		6		8	
optimum cost	119		238		773		105		623		83	
Algorithm	Largest message	Total data	Largest message	Total data	Largest message	Total data	Largest message	Total data	Largest message	Total data	Largest message	Total data
DCTE	128(119)	335.4	128(238)	837.1	128(773)	921.5	390.6(105)	2471.8	1024(623)	7777	1525.9(83)	3433.8
DIMCTEf($r = 2$)	0.3[115,121]	12.6	0.3[216,242]	30.0	0.3[713,793]	58.1	0.4[98,105]	35.6	0.3[541,677]	89.4	0.1[65,100]	6.6
DIMCTEf($r = 3$)	*2.0[118,120]	*12.7	2.0[231,241]	94.4	2.0[763,779]	193.0	3.9[102,105]	10.7	2.0[581,635]	297.6	0.5[68,100]	16.5
DIMCTEf($r = 4$)	*16.0[119,120]	*61.5	*16.0[238,238]	*90.0	*16.0[773,773]	*159.6	*38.3(empty f)	*141.3	16.0[607,625]	1163.1	2.5[79,89]	51.6
DIMCTEf($r = 5$)	*79.7[119,119]	*123.2							*128.0[623,623]	*835.3	12.2[79,89]	51.3
DIMCTEf($r = 6$)											60.6[79,89]	374.5
DIMCTEf($r = 7$)											61.0[79,83]	265.3
DIMCTEf($r = 8$)											*855.5[83,83]	*1101.4
Savings $\delta = 0\%$	38%	37%	88%	74%	88%	55%	90%	92%	88%	69%	44%	46%
Savings $\delta = 5\%$	99%	96%	98%	85%	98%	73%	99%	98%	98%	80%	96%	78%

Figure 11. Largest message and total data transfer of DCTE and DIMCTEf for increasing r on random instances, in Kbytes. Instances are ordered left-to-right for increasing d^s. For DCTE, the optimum global cost appears between parenthesis. For DIMCTEf, the lower and upper bounds returned at each iteration appear between brackets. The user is willing to accept solutions whose cost surpasses the optimum up to δ ($\delta = 0\%$ means optimum only). With asterisk (*) iterations that are executed for $\delta = 0\%$ but not for $\delta = 5\%$.

instance	1		2		3		4	
#participants	40		50		40		70	
#departments	10		10		8		10	
#meetings	15		22		18		24	
#calendar slots	8		8		8		8	
#agents	50		68		60		72	
#variables	171		266		242		267	
domain size	8		8		8		8	
#cost functions	210		340		324		342	
largest separator	4		5		6		6	
optimum cost	126		183		137		187	
Algorithm	Largest message	Total data	Largest message	Total data	Largest message	Total data	Largest message	Total data
DCTE	16(126)	101	128(183)	1041.5	1024(137)	5493.9	1024(187)	4476.5
DIMCTEf($r = 2$)	0.3[122,141]	25.1	0.3[168,201]	42.8	0.3[128,158]	40.6	0.3[171,211]	41.1
DIMCTEf($r = 3$)	2.0[126,126]	56.8	2.0[182,183]	145.3	2.0[130,156]	119.2	2.0[184,190]	151.9
DIMCTEf($r = 4$)			*6.6(empty f)	*49.7	16.0[133,137]	457.0	*16.0[187,187]	*77.5
DIMCTEf($r = 5$)					*31(empty f)	*150.0		
Savings $\delta = 0\%$	88%	19%	95%	77%	97%	86%	98%	94%
Savings $\delta = 5\%$	88%	19%	98%	82%	98%	89%	99%	96%

instance	5		6		7		8	
#participants	60		80		100		90	
#departments	20		20		25		15	
#meetings	25		22		52		32	
#calendar slots	8		8		8		8	
#agents	98		104		160		112	
#variables	429		408		730		457	
domain size	8		8		8		8	
#cost functions	570		536		968		606	
largest separator	7		7		7		8	
optimum cost	279		287		447		263	
Algorithm	Largest message	Total data	Largest message	Total data	Largest message	Total data	Largest message	Total data
DCTE	8192(279)	40017	8192(287)	21813	8192(447)	88532	65536($-$)	($-$)
DIMCTEf($r = 2$)	0.3[237,298]	84.6	0.3[260,330]	75.3	0.3[376,493]	145.0	0.3[234,316]	75.1
DIMCTEf($r = 3$)	2.0[263,284]	291.1	2.0[260,313]	202.6	2.0[408,486]	486.7	2.0[250,289]	273.9
DIMCTEf($r = 4$)	16.0[278,279]	609.1	16.0[284,292]	930.7	16.0[423,467]	2233.0	16.0[250,278]	727.2
DIMCTEf($r = 5$)	*128.0(empty f)	*1283.1	*128.0[287,287]	*288.6	128.0[446,447]	5704.0	121.9[261,272]	1194.0
DIMCTEf($r = 6$)					*102.0(empty f)	*9111.0	*821.3[262,263]	4202.5
DIMCTEf($r = 7$)							*987.6(empty f)	2930.7
Savings $\delta = 0\%$	98%	94%	98%	93%	98%	80%	(97.04%)	
Savings $\delta = 5\%$	99%	98%	99%	94%	98%	90%	(99.83%)	

Figure 12. Largest message and total data transfer of DCTE and DIMCTEf for increasing r on meeting scheduling instances, in Kbytes. Instances are ordered left-to-right for increasing d^s. For DCTE, the optimum global cost appears between parenthesis ("$-$" means that the algorithm execution exhausted memory before reaching the solution). For DIMCTEf, the lower and upper bounds returned at each iteration appear between brackets. The user is willing to accept solutions whose cost surpasses the optimum up to δ ($\delta = 0\%$ means optimum only). With asterisk (*) iterations that are executed for $\delta = 0\%$ but not for $\delta = 5\%$. Between parenthesis, savings in largest message size of those instances on which DCTE exhausted memory.

cost function filtering. In these partial tables some extra pointers are needed in order to indicate the right position of the elements in the table. This causes transferring some extra data, which are not needed when transferring complete tables.

In several cases, DIMCTEf does not reach the iteration with $r = s$; it stops before either because $LB = UB$ or an empty function is computed. This causes substantial savings in communication cost and in computation effort, since as r increases, both communication and computation requirements also increase. Even in the cases where the last iteration of DIMCTEf reaches s, the DIMCTEf largest message is lower than the DCTE largest message, by the effect of function filtering.

If we allow for a solution with a cost within 5% of the optimum, benefits increase because the termination condition on bounds becomes looser: it pass from $LB = UB$ to $LB \geq UB \times \frac{95}{100}$. For many problem instances, DIMCTEf stops before than when computing the optimum, saving several iterations for the highest values of r, which represents further savings in communication and computation. Regarding largest message size, DIMCTEf savings with respect to DCTE are really good: they range from 88% to 99%. Regarding total data exchanged, a similar picture appears: DIMCTEf exchanges from 27% to 98% less data than DCTE (even at the instance E, where total data exchanged were higher for DIMCTEf when looking for optimal solutions). It is really illustrative to observe the capacity of reasoning with bounds, able to stop execution when bounds are close enough and, in many cases, saving some of the most costly iterations in largest message size and total data exchanged (marked with * in Figure 11).

Further experimental results on the distributed meeting scheduling benchmark appear in Figure 12. Results solving these structured instances are similar to those observed solving random instances. When computing the optimum ($\delta = 0\%$), the largest DIMCTEf messages are substantially shorter than the corresponding largest DCTE messages: from 88% to 98% shorter (from 1 to 2 orders of magnitude reduction in largest message size). In addition, in one of the eight tested instances, DCTE was unable to compute the exact solution because it exhausted memory (messages of size 65536Kb cannot be handled by our simulator). In this instance, DIMCTEf was able to compute the exact solution using substantially smaller messages (with a saving of 98% in largest message size, with respect to the message size that DCTE would have used). Regarding total data transferred, DIMCTEf exchanges from 19% to 94% less data than DCTE, which represents a substantial improvement in communication.

As happens with random instances, allowing solutions with cost within 5% from the optimum causes further benefits. Although there is no change in the smallest instance, in the other instances the last iteration of DIMCTEf when computing the optimum (the two last iterations for instance 8), marked with * in Figure 12, is not needed, with the corresponding savings in communication cost and communication effort. It is worth noting that iterations tend to be more costly as r increases, so saving last iterations usually causes important savings in largest message size and data exchanged. These results illustrate clearly the value of our approach and show the applicability enhancements of DMCTEf with respect to DCTE, which was unable to solve one of the eight instances considered.

These experiments demonstrate the benefits that DIMCTEf may cause when applied to practical problems, with respect to the exact algorithm DCTE. First, using cost function filtering, DIMCTEf is able to achieve the optimal solution requiring shorter messages and, in most cases, less data exchanged than DCTE. These reductions are substantial. Second, DIMCTEf reasons on problem (lower/upper) bounds. It stops execution when these bounds are close enough according to user specifications (which causes further benefits, as shown in the experiments). Third, DIMCTEf terminates when it detects empty functions (functions with all their tuples pruned by the current upper bound). Empty functions appear in the last, more costly, iterations so its detection produces important savings. In a more general setting, DIMCTEf is an anytime algorithm, so it can be stopped at any time (for instance, broadcasting a special message to all agents). Then, agents can take as best available solution the assignment computed at the previous iteration, with the current upper bound as cost.

7. Related Work

In the last years several solving algorithms for DCOPs have been proposed. Broadly speaking, these algorithms can be grouped in two main families: algorithms based on distributed search, and algorithms based on distributed dynamic programming. Algorithms based on distributed search exchange messages containing assignments and costs. These messages contain little information and they have a small size, but their number can be quite high (in the worst case, proportional to the number of nodes of the search tree). On the other hand, algorithms based on dynamic programming exchange cost functions, These algorithms exchange a relatively low number of messages, but these messages may be of high (exponential) size.

Considering distributed search algorithms, we mention the synchronous approaches SBB [8] and NCBB [3]. About asynchronous approaches, we mention ADOPT [14] and its new versions ADOPT-ng[20] and BnB-ADOPT [24]. We also mention AFB [7]. While ADOPT uses a best-first strategy, all the others follow a depth-first branch-and-bound strategy. ADOPT discards a partial solution if another potential solution looks more promising. But later, this second solution may appear less promising than the discarded one, so ADOPT has to reconstruct previous partial solutions that were discarded before. ADOPT is able to compute quality guarantees when providing approximate solutions.

Considering distributed dynamic programming, DPOP [18] has to be mentioned. It works on a depth-first tree (similar to pseudotree) of the constraint graph. Basically, a DPOP agent waits for cost functions computed by its children, adds these cost functions with its own ones and sends the result to its parent, projecting out the variable contained in the agent. When this bottom-up information reaches the root, it looks for the value that minimizes the received cost function, and informs its children and pseudochildren, which repeat the process until this information flow reaches tree leaves. DPOP uses a linear number of messages, but the size of the largest message is exponential in the induced width w^* of the pseudotree. We also mention Action-GDL [22], which performs a closely related process on a TD.

The approach presented here clearly belongs to the dynamic programming group. In that sense, it is close to DPOP, with the aim of decreasing the high size of the largest message (exponential in the induced width w^*). DCTE has similar requirements to DPOP on largest message size: given a pseudotree of induced width w^*, it is always possible to build a TD with maximum separator size $w*$ (the bucket tree, see [10]). In practice, with the use of cost function filtering DIMCTEf largely decreases the size of the largest message exchanged, maintaining the optimality of the solution (as seen in the experimental section).

Finally, we have to mention other approaches like OptAPO [12]. This algorithm is partially centralized so it cannot be easily related with the present approach.

8. Summary and Discussion

We have presented DCTE, DMCTE(r), DMCTEf(r) and DIMCTEf, four distributed synchronous algorithms for solving DWCSPs (a more precise version of the well-known distributed COPs). First, a

DWCSP instance can be arranged in a TD by a distributed algorithm. Working on such TD, DCTE solves the DWCSP instance exactly: after exchanging some messages each agent minimizes its cluster and finds the global optimum. DCTE requires a number of messages linear in the number of agents, but message size can be up to $exp(s)$, where s is the size of the largest separator in the TD. DMCTE(r) limits message size to $exp(r), r < s$, at the extra cost of achieving approximate solutions. It also provides a cost interval where the optimum appears, bounding the error of the approximation. DMCTEf(r) uses the function filtering technique to compute smaller messages (or of equal size, in the worst case) than DMCTE(r), although both reach solutions of the same quality. DIMCTEf is an iterative algorithm that executes DMCTEf(r) with increasing r, using cost functions received at the previous iteration as filters for the computation of cost functions at the current iteration. Reasoning with bounds, it is able to provide either exact or approximate solutions according with user specifications. The running example shows clearly the benefits of DIMCTEf, that causes a drastic decrement in the number of exchanged tuples.

DIMCTEf is a good example of exchanging memory for time, situation that often appears in AI. While DMCTE(r) reaches an approximate solution, DIMCTEf is able to improve the solution quality, but requires some extra computation. DIMCTEf is also a good example of dynamic programming strategies, able to build up a solution of an instance working from solutions of its subinstances. It has some common flavor with *Russian Doll Search* [21].

Let us assume that we are uniquely interested in solving the problem instance. If we are willing to handle messages of size $exp(s)$, DCTE messages from the root to leaves are not strictly needed to compute a solution (it would be enough with sending the values of variables in the separator). However, exchanging messages that contain more than needed to compute the optimal solution, allows us to define the approximate versions of DCTE. Combined with the filtering strategy and adequately iterated, we obtain the DIMCTEf algorithm, able to find approximated solutions using smaller messages than DCTE, being able to find exact solutions in many cases. In addition, as the experimental results show, there are instances with large separators for which the exact algorithm DCTE exhausts memory before reaching the solution, so it is not a real solving option. For all these instances, DIMCTEf is able to compute the exact solution using much shorter messages.

References

[1] Bejar R., Fernandez C., Valls M., Domshlak C., Gomes C., Selman S., Krishnamachari B. Sensor networks and distributed CSP: Communication, computation and complexity. *Artificial Intelligence*, 161:117–147, 2005.

[2] Bodlaender H. Treewidth: algorithmic techniques and results *Proc. MFCS-97*, 19–36, 1997.

[3] Chechetka A., Sycara K. No-commitment branch and bound search for distributed constraint optmization *Proc. of AAMAS-06* 1427–1429, 2006.

[4] Dechter R. Bucket elimination:A unifying framework for reasoning. *Artificial Intelligence*, 113:41–85, 1999.

[5] Dechter R. *Constraint Processing*. Morgan Kaufmann, 2003.

[6] Dechter R., Kask K., Larrosa J. A general scheme for multiple lower bound computation in constraint optimization. *Proc. CP-01*, 346–360, 2001.

[7] Gershman A., Meisels A., Zivan R. Asynchronous forward-bounding for distributed constraint optimization *Proc. of ECAI-06* 103–107, 2006.

[8] Hirayama K., Yokoo M. Distributed partial constraint satisfaction problem *Proc. of CP-97*, 222–236, 1997.

[9] Larrosa J. Node and arc consistency in weighted CSP *Proc. of AAAI-02*, 48–53, 2002.

[10] Kask K., Dechter R. Larrosa J., Dechter A. Unifying cluster-tree decompositions for reasoning in graphical models. *Artificial Intelligence*, 166:165–193, 2005.

[11] Maheswaran R., Tambe M., Bowring E., Pearce J., Varakantham P. Taking DCOP to the Real World: Efficient Complete Solutions for Distributed Multi-Event Scheduling *Proc. of AAMAS-04*, 310–317, 2004.

[12] Mailler R., Lesser V. Solving distributed constraint optimization problems using cooperative mediation *Proc. of AAMAS-04*, 438–445, 2004.

[13] Meseguer P., Rossi F., Schiex T. Soft constraints *Handbook of Constraint Programming* Ed van Beek, Rossi, Walsh, 281–328, 2006.

[14] Modi P. J., Shen W.M., Tambe M., Yokoo M. Adopt: asynchronous distributed constraint optimization with quality guarantees. *Artificial Intelligence*, 161:149–180, 2005.

[15] Paskin M., Guestrin C., McFadden J. A robust architecture for distributed inference in sensor networks. *Proc. of IPSN*, 55–62, 2005.

[16] Perlman R. An algorithm for distributed computation of a spanning tree in an extended LAN. *ACM SIGCOMM Computer Communication Review*, 44–53, 1985.

[17] Petcu A. FRODO: A FRamework for Open and Distributed constraint Optimization. Technical Report 2006/001. Swiss Federal Institute of Technology (EPFL) Laussane, Switzerland, 2006.

[18] Petcu A., Faltings B. A scalable method for multiagent constraint optimization *Proc. of IJCAI-05*, 266–271, 2005.

[19] Sanchez M., Larrosa J., Meseguer P. Improving Tree Decomposition Methods with Function Filtering. *Proc. of IJCAI-05*, 1537–1538, 2005.

[20] Silaghi M. Yokoo M. Nogood-based Asynchronous Distributed Optimization (ADOPT-ng). *Proc. of AAMAS-06*, 1389–1396, 2006.

[21] Verfaillie G., Lemaître M., Schiex T. Russian doll search. *Proc. of AAAI-96*, 181–187, 1996.

[22] Vinyals M., Rodriguez-Aguilar J.A., Cerquides J. Generalizing DPOP: Action-GDL, a new complete algorithm for DPOPs. *Proc. of AAMAS-09*, 1239–1240, 2009.

[23] Wallace R., Freuder E. The Distributed Constraint-based reasoning and privacy/efficiency tradeoffs in multiagent problem solving. *Artificial Intelligence*, 161:209–227, 2005.

[24] Yeoh W., Felner A. Koenig S. BnB-ADOPT: An Asynchronous Branch-and-Bound DCOP Algorithm *Proc. of AAMAS-08*, 591–598, 2008.

[25] Yokoo M., Durfee E., Ishida T., Kuwabara K. The Distributed Constraint Satisfaction Problem: Formalization and Algorithms. *IEEE Trans. Know. and Data Engin.*, 10:673–685, 1998.