Adaptive Domain Abstraction in a Soft-Constraint Message-Passing Algorithm

Paul Maier and Martin Sachenbacher

Technische Universität München, Department of Informatics
Boltzmanstraße 3, 85748 Garching, Germany
{maierpa,sachenba}@in.tum.de
http://www9.in.tum.de

Abstract. The computational tasks of model-based diagnosis and planning in embedded systems can be framed as soft-constraint optimization problems with planning costs or state transition probabilities as preferences. Running constraint optimization in embedded systems requires to reduce complexity, which can be achieved by combining dynamic programming message-passing algorithms with message approximation. We found that current approximation approaches such as Mini-Cluster Tree Elimination (MCTE) lack flexibility in adapting to resource limits such as limited memory, e.g. imposed by embedded controllers.
We propose a new message approximation method based on the adaptive abstraction of domains and constraints, extending upon MCTE. We argue that our approach can be more flexibly adapted to imposed size limits when applied to constraint optimization problems with big constraints and big domains, which are typical for diagnosis and planning. It is further shown that the adaptation step is itself an optimization problem, which can be relaxed to and solved as a linear optimization problem.
From preliminary empirical tests we conclude that the method has potential for diagnosis problems, but is probably limited with regard to binary constraint optimization problems.

1 Introduction

New generations of technical devices are being developed that use models of themselves to implement self-awareness capabilities, for example in assistant automotive systems and cognitive manufacturing systems [1]. Many of the underlying computational tasks – such as automatically determining the most likely current state of the system (monitoring and diagnosis), or finding least-cost sequences of actions that drive the system towards desired states to compensate for contingencies (planning and reconfiguration) – can be framed as soft-constraint optimization problems [2] with planning costs or transition probabilities as preferences. Such a common representational basis enables tight integration of the different tasks.

However, constraint optimization is exponential in the number of system variables, which is especially a problem given the limited resources (memory

and CPU time) available in embedded controllers. The problem can be eased by decomposing the constraint model off-line into clusters and inferring solutions on-line by passing messages between the clusters [3]. Such message-passing approaches are especially promising as models of technical systems often present some inherent modularity (for instance, a car's controller area network). Actually, one of the first approaches to constraint-based diagnosis used a message-passing scheme [4].

Unfortunately, this can still lead to an infeasible message size (exponential in the separator width of the decomposition). In the literature, mini-clustering [5] has been suggested as a scheme to limit the memory requirements of structure-based constraint optimization algorithms, by approximating messages instead of computing them exactly. The idea of mini-cluster tree elimination (MCTE) is to restrict the size of messages in the tree by partitioning the constraints in the clusters into subsets (called mini-clusters) involving at most i different variables. By combining only the constraints in a mini-cluster, one gets a set of messages that approximate the cost function of the original message with an upper (optimistic) bound, and this can be used as a heuristic for subsequent search. The complexity of mini-cluster tree elimination is therefore $O(r \cdot k^i)$, where k is the largest domain size, and r is the total number of occurring soft constraints [3].

However, when we tried to use mini-clustering as a reasoning scheme in embedded constraint optimization, we encountered two problems. First, it is difficult to control the number of messages that need to be transmitted. If i is small, the messages themselves are computed faster but their total number will increase. Limiting the number of messages by suppressing some of them is hardly a solution, because constraints are unique to their respective mini-cluster and thus information about these constraints will be completely lost in the heuristic approximation. Second, the parameter i allows only limited control over the size of the resulting messages, because MCTE can only either omit a constraint or use it in a message, without any intermediate steps. While this works for models with small (binary) constraints and small variable domains, typical constraint models for diagnosis and planning tend to have large domains and big constraints. For instance, in a model of NASA's Earth Observing Satellite (EO-1) [6], variables have up to ten domain values and many constraints involve more than four variables. For such larger problems, mini-clustering offers only limited possibilities to form the mini-clusters and therefore it cannot control the size of the messages effectively. In fact, restricting the number of constraints occurring in messages (or even limiting the number of variables in a constraint by projecting them on a subset of their variables) will affect only the exponent of the space complexity; depending on k, this can lead to big jumps in the possible message size.

The work presented in this paper extends upon the work previously presented in [7]. Extending upon the mini-cluster scheme, we propose a more general approach to limit the size of messages exchanged between clusters. It allows for finer control over the message size, and therefore enables better adaptation of soft constraint message-passing algorithms to the tight resources in embedded

systems. Instead of omitting constraints from messages (as in mini-clusters), our approach adaptively reduces the size of constraints by abstracting them, similar to the automated generation of search heuristics (pattern databases) in game analysis and path search [8,9]. The key idea is to choose appropriate partitionings of the domains, aggregating the variable values into abstract values, that limit the worst-case complexity of messages by allowing only a limited number of total distinctions. This partitioning affects both the base and the exponent of the message complexity, and therefore allows for a more fine-grained control compared to the mini-cluster parameter i. It is worth noting that using such abstractions, one can reconstruct the behavior of the original mini-cluster algorithm as a special case where the identical abstraction (preserving all distinctions) is applied to the constraints inside a mini-cluster, and the trivial abstraction (eliminating all distinctions) is applied to the constraints outside the mini-cluster.

We have implemented this approach as a variant of the existing message-passing algorithm MCTE. The new algorithm, called Bucket Elimination with Domain Abstraction (BEDA), was integrated into the open-source constraint solver Toolbar [10] to allow comparison with MCTE and other constraint optimization algorithms. Together with Toolbar the new algorithm is available to the public from the Toolbar repository[1]. BEDA combines tree decomposition with automated partitioning of variable domains, which reduces domain sizes and yields smaller, abstract constraints that approximate the original constraints. Additionally, in order to measure the quality of the approximation, we combined the MCTE and BEDA inference methods with A* search, utilizing the inferred messages as heuristic. This idea was introduced in [5].

2 Adaptive Domain Abstraction

2.1 Mini-Cluster Tree Elimination

We start by recalling the inference method MCTE for constraint optimization problems. First we define the necessary constructs of a constraint optimization problem (COP), tree decomposition and appropriate operators on soft constraints. We use the well known framework of weighted CSPs (WCSPs) [11]. It defines soft constraints as functions which map variable assignments to costs $\{0, 1, \ldots, \top\}$, with $\top$ as the maximum cost.

Definition 1. *A constraint optimization problem (COP for short)* $\mathcal{R} = \langle X, D, F \rangle$ *is defined as follows:*

1. $X = \{X_1, \ldots, X_n\}$ *is a set of* n *variables.*
2. $D = \{D_1, \ldots, D_n\}$ *is the collection of the domains of the variables in* X *such that* D_i *is the domain of* X_i*. For a given variable* X_i *we may also denote its domain by* D_{X_i}*. The size of the domains are written as* $|D_i| = d_i$*. We write* D_Y *to denote the Cartesian product* $\prod_{X_i \in Y} D_i$ *for some subset of the*

[1] `http://mulcyber.toulouse.inra.fr/gf/project/toolbar/`

variables $Y \subseteq X$. Accordingly, D_X denotes the Cartesian product $\prod_{X_i \in X} D_i$ of all domains.

3. *$F = \{f_1, \ldots, f_r\}$ is a finite set of r soft constraints. A soft constraint is a function f on a sequence of variables $V \subseteq X$, called the scope of the constraint, such that f maps assignments (of variables in V to values in their domains) to cost values: $f : \prod_{X_i \in V} D_i \mapsto \{0, \ldots, \top\}$. If an assignment is mapped to $\top$, it is considered inconsistent.*
4. *A solution to $\mathcal{R}$ is a consistent assignment to all variables. An optimal solution minimizes the accumulated cost over all constraints.*

Within the WCSP framework, soft constraint marginalization and combination operators are given as

$$(\Downarrow_Y f)(t) = \min_{t' \in t_{\downarrow Y}} f(t')$$

and

$$(\bigotimes_{f \in C} f)(t) = \sum_{f \in C} f(t_{\downarrow scope(f)}),$$

respectively, for some subsets $Y \subseteq X$ and $C \subseteq F$. $t_{\downarrow Y}$ is assignment t projected onto Y. We write $t' \in t_{\downarrow Y}$ to refer to full assignments t' in the set $\{t' \in D_X | t'_{\downarrow Y} = t_{\downarrow Y}\}$, represented by $t_{\downarrow Y}$.

A tree decomposition for a given COP $\mathcal{R} = \langle X, D, F \rangle$ is a triple $\langle T, \chi, \psi \rangle$, where $T = (V, E)$ is a tree, and χ, ψ are labeling functions which associate with each node $v \in V$ two sets, $\chi(v) \subseteq X$ and $\psi(v) \subseteq F$. The sets satisfy three conditions, (1) for each function $f \in F$, there is exactly one node $v \in V$ such that $f \in \psi(v)$, $scope(f) \subseteq \chi(v)$, (2) for each variable $X_i \in X$, the set $\{v \in V | X_i \in \chi(v)\}$ induces a connected sub tree of T and (3) $\forall i : X_i \in \chi(v)$ for some $v \in V$.

Given a COP $\mathcal{R} = \langle X, D, F \rangle$ and a tree decomposition $\langle T, \chi, \psi \rangle$, solutions can be inferred by passing messages between the tree nodes. Each node sends and receives messages over the tree edges. Associated with each edge $e \in E$ of the decomposition tree T is the separator $sep(u, v) = \chi(u) \cap \chi(v)$ of the two nodes $u, v \in V$ linked through e. A message

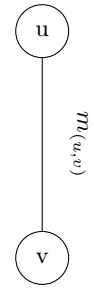
$$m_{(u,v)} = \Downarrow_{sep(u,v)} \bigotimes_{f \in \psi(u) \cup \{m_{(i,u)} | (i,u) \in T, i \neq v\}} f$$

for a node $u \in V$ is computed and send to the adjacent node v as soon as u has received all its messages. MCTE extends this approach by partitioning the set $\psi(u)$ for a tree node u into subsets $Q = \{Q_1, \ldots, Q_q\}$ of constraints such that the subsets do not contain more than i variables. The subsets are called mini-clusters. For each mini-cluster Q_l a mini-message $h_{(u,v)} = \Downarrow_{sep(u,v)} \bigotimes_{f \in Q_l} f$ is computed. These are sent to node v instead of the single message $m_{(u,v)}$. The combination of the mini-messages approximates the original message, somewhat informal expressed as $\bigotimes_{Q_l \in Q} \Downarrow_{sep(u,v)} \bigotimes_{f \in Q_l} f \leq m_{(u,v)}$ [3]. As stated earlier, MCTE complexity is bound by the total number of cost functions and i: $O(r \cdot k^i)$.

The parameter i provides only a rough means to control the message size and it is hard to control the number of messages. We assume, given tight resource limits, that we have a hard size limit $T_{max} \in \mathbb{N}$, bounding the number of tuples of messages generated during the message-passing phase. In our approach, we compute a defined number of abstract messages (currently one) per cluster strictly limited in size by T_{max}.

2.2 Reducing Message Size Through Domain Abstraction

Before we explain our approach in detail we illustrate in the following example how adaptive domain abstraction works compared to MCTE and CTE. We denote the abstraction of some structure (a constraint f, a message $m_{(u,v)}$, a domain D_i) by writing its symbol with superscript (α) ($f^{(\alpha)}$, $m^{(\alpha)}_{(u,v)}$, $D^{(\alpha)}_i$).



$\chi(u) = \{A, B, C, D, E\}$
$\psi(u) = \{f_1, f_2, f_3\}$

$\chi(v) = \{A, B, C, D\}$
$\psi(v) = \dots$

f_1:

A	B	C	cost
0	0	0	0
0	0	1	1
⋮			⋮
2	2	2	6
2	2	3	7
⋮			⋮

f_2:

A	B	C	D	E	cost
0	0	0	0	0	0
0	0	0	0	1	1
⋮					⋮
2	2	2	1	1	8
2	2	2	1	2	9
⋮					⋮

f_3:

A	E	cost
0	0	0
0	1	1
⋮		⋮
2	1	3
2	2	4
⋮		⋮

CTE ($m_{(u,v)}$):

A	B	C	D	cost
0	0	0	0	0
0	0	0	1	1
⋮				⋮
2	1	2	3	15
2	1	2	4	16
⋮				⋮

MCTE ($h_{(u,v)}(f_1, f_3)$):

A	B	C	cost
0	0	0	0
0	0	1	1
⋮			⋮
2	0	0	4
2	0	1	5
⋮			⋮

BEDA ($m^{(\alpha)}_{(u,v)}$):

A	B	C	D	cost
$0^{\alpha(0,1)}$	$0^{\alpha(0,1)}$	$0^{\alpha(0,1)}$	$0^{\alpha(0,1)}$	0
$0^{\alpha(0,1)}$	$0^{\alpha(0,1)}$	$0^{\alpha(0,1)}$	$1^{\alpha(2,3)}$	2
⋮				⋮
$2^{\alpha(3)}$	$1^{\alpha(2,3)}$	$2^{\alpha(3)}$	$1^{\alpha(2,3)}$	21
$2^{\alpha(3)}$	$1^{\alpha(2,3)}$	$2^{\alpha(3)}$	$2^{\alpha(4)}$	23
⋮				⋮

Fig. 1. Domain abstraction for a single tree decomposition node: From the soft constraints f_1, f_2, f_3 of node u, CTE computes message $m_{(u,v)}$, MCTE the mini-message $h_{(u,v)}(f_1, f_3)$ and BEDA message $m^{(\alpha)}_{(u,v)}$. The constraints and messages are shown as tables of their tuples. The domains of the variables are finite sets of integers: $D_A, D_C = \{0, 1, 2, 3\}$ and $D_B, D_D, D_E = \{0, 1, 2, 3, 4\}$. Their abstractions are $\{0^{\alpha(0,1)}, 1^{\alpha(2)}, 2^{\alpha(3)}\}$ and $\{0^{\alpha(0,1)}, 1^{\alpha(2,3)}, 2^{\alpha(4)}\}$, respectively.

Figure 1 shows two nodes u and v of a tree decomposition, where u has associated variables $\{A, B, C, D, E\}$ and associated constraints f_1, f_2, f_3 with scopes $\{A, B, C\}$, $\{A, B, C, D, E\}$ and $\{A, E\}$, respectively. Node v has associated variables $\{A, B, C, D\}$, its associated constraints are not important here.

The domains of the variables are finite sets of integers: $D_A, D_C = \{0,1,2,3\}$ and $D_B, D_D, D_E = \{0,1,2,3,4\}$. u sends the message $m_{(u,v)}$ to node v, its scope is $\{A,B,C,D\}$. The constraints f_i are check sum constraints, i.e. the cost of a tuple is simply its check sum. Figure 1 depicts message $m_{(u,v)}$ for the case of Cluster-Tree Elimination (CTE), where $m_{(u,v)} = \min_E(f_1 + f_2 + f_3)$.

Consider now a size limit $T_{max} = 200$ for the abstract messages in BEDA. A comparable MCTE parameter i must be $i < 5$, since choosing $i = 5$ would allow for messages as big as $|D_A| \cdot |D_B| \cdot |D_C| \cdot |D_D| = 4 \cdot 5 \cdot 4 \cdot 5 = 400$ (which is actually the size of CTE message $m_{(u,v)}$).

This means f_1 and f_3 can be combined in a mini-message ($h_{(u,v)}(f_1, f_3)$ in figure 1) because their combined arity is ≤ 3, but f_2 is simply too big and must be omitted by MCTE. The only possibility to keep f_2 would be to set $i = 5$, which would however result in the above mentioned violation of the size limit T_{max}.

Turning to BEDA, violation of T_{max} through $m_{(u,v)}$ is avoided by first adapting abstract domain sizes for all $X_i \in \mathit{scope}(m_{(u,v)})$ to T_{max}: $d_i^{(\alpha)} = \left\lfloor \sqrt[|\{A,B,C,D\}|]{T_{max}} \right\rfloor = \left\lfloor \sqrt[4]{200} \right\rfloor = 3$. Then we choose an according domain abstraction function $\tau^{d_i^{(\alpha)}}(y) = \tau^3(y)$ to create abstract domains with $d_i^{(\alpha)}$ partition elements. The partitionings could for example group values of similar cost: $\{0^{\alpha(0,1)}, 1^{\alpha(2)}, 2^{\alpha(3)}\}$ for variables A, C and $\{0^{\alpha(0,1)}, 1^{\alpha(2,3)}, 2^{\alpha(4)}\}$ for variables B, D, E. The superscript notation $0^{\alpha(0,1)}$ indicates that concrete variable values $0, 1$ are abstracted to value 0. These operations are part of the (possibly off-line) adaptation step, which is concluded (after processing all other separators) by forming a constraint abstraction function Γ.

During message passing, Γ is applied to the COP constraints. From these abstract constraints, abstract messages are then computed. The abstract message is shown in figure 1. The cost of an abstract tuple is a lower bound for the costs of all its concretions. Consequently, the abstraction is an admissible heuristic for the concrete constraint/message.

The abstract message now respects T_{max}: $|m_{(u,v)}^{(\alpha)}| = d_A^{(\alpha)} \cdot d_B^{(\alpha)} \cdot d_C^{(\alpha)} \cdot d_D^{(\alpha)} = 3^4 = 81$. The information of the big constraint f_2 is at least partly preserved, but one can also see that the greedy approach abstracts still too coarse, especially because it treats all domains equally. Note that in general, during the adaptation, other separators elsewhere in the tree decomposition might require even smaller domain sizes. For this example we assumed that this is not the case.

We argue that this approach allows a finer control over message sizes than MCTE, which is illustrated in the example: In MCTE, a constraint might have to be omitted due to the size limit, while using domain abstraction allows to keep all constraints as abstractions while respecting the size limit.

Now we formally define the notions of domain and constraint abstraction and show how to reduce the size of constraints and messages by reducing domain sizes.

Definition 2. *Given a COP $\mathcal{R} = \langle X, D, F \rangle$, we define the following:*

1. *A domain abstraction function* $\tau_i : D_i \mapsto D_i^{(\alpha)}$ *is a surjective function which maps the values in domain* D_i *to a set* $D_i^{(\alpha)}$*, called the abstraction of* D_i*, or simply abstract domain. Values in* $D_i^{(\alpha)}$ *are abstract values and are assigned to the abstract counter parts of variables* X_i*:* $X_i^{(\alpha)}$*. We write* $a^{\alpha(v_1,\dots,v_k)}$ *to indicate that values* $v_1, \dots, v_k \in D_i$ *are mapped to an abstract value* a *by some domain abstraction function* τ_i*.*
2. *An abstract assignment* $t^{(\alpha)}$ *is an assignment to a subset of abstract variables* $Y^{(\alpha)} \subseteq X^{(\alpha)}$*. An abstract assignment is created by applying the appropriate domain abstraction functions* τ_i *to each value* $v \in D_i$ *in the concrete assignment. We denote this relationship by* $t^{(\alpha)} = \tau(t)$*, where* τ *represents the combined application of the domain abstraction functions* τ_i*.*

A domain abstraction function creates a partitioning of the domain. This means abstract values can be seen as sets which aggregate the original concrete values and, accordingly, abstract assignments as sets of concrete assignments. We express this fact by writing $t \in t^{(\alpha)}$ when referring to a concrete assignment t which is abstracted through the abstract assignment $t^{(\alpha)}$.

Definition 3. *Let* $\mathcal{R} = \langle X, D, F\rangle$ *be a COP and* τ_i *the domain abstraction functions for each domain* D_i*. Then a constraint abstraction function* $\Gamma : F \mapsto F^{(\alpha)}$ *is a function which maps soft constraints* $f \in F$ *to abstract soft constraints* $f^{(\alpha)} \in F^{(\alpha)}$*. The scope of* $f^{(\alpha)}$ *are the abstractions of the variables in the scope of* f*:* $\{X_i^{(\alpha)} | \, X_i \in scope(f)\}$*.* $f^{(\alpha)}$ *is defined through* f *and* τ *by*

$$\forall t \in D_{scope(f)} : f^{(\alpha)}(\tau(t)) = \min_{t' \in \tau(t)} f(t')$$

Applying given domain abstraction functions τ_i to all domains in COP $\mathcal{R} = \langle X, D, F\rangle$ and the constraint abstraction function Γ to all constraints we receive a new COP $\mathcal{R}^{(\alpha)} = \langle X^{(\alpha)}, D^{(\alpha)}, F^{(\alpha)}\rangle$. The new COP is hopefully easier to solve as its domains and therefore constraints can be smaller. The abstraction functions only modify the number of possible assignments of $\mathcal{R}$, but leave constraint network and the cost structure, i.e. the set $\{0, \dots, \top\}$ unchanged. Therefore, $\mathcal{R}^{(\alpha)}$ is just another WCSP and can be solved applying the same operations.

Proposition 1. *Let* $\mathcal{R} = \langle X, D, F\rangle$ *be a COP and* $\mathcal{R}^{(\alpha)} = \langle X^{(\alpha)}, D^{(\alpha)}, F^{(\alpha)}\rangle$ *its abstraction, induced by appropriate functions* τ *and* Γ*. Then for all constraints* $f \in F$ *and their abstractions* $f^{(\alpha)} \in F^{(\alpha)}$*,* $f^{(\alpha)}$ *is a lower bound for* f*:*

$$\forall t \in D_{scope(f)} : f^{(\alpha)}(\tau(t)) \leq f(t)$$

Furthermore, for all assignments t *to variables* X*, the cost of its corresponding abstract assignment* $t^{(\alpha)} = \tau(t)$ *to* $X^{(\alpha)}$ *is a lower bound on the cost of* t*.*

This means that in message-passing, we can approximate the concrete messages with abstract messages and, just as mini-cluster messages, use them as admissible heuristic in a subsequent search for an optimal solution.

2.3 BEDA Algorithm

The algorithms 1 and 2 show the greedy domain size adaptation and BEDA, respectively. The pseudo code in algorithm 2 is partly taken from the CTE code in [12]. The BEDA algorithm essentially consists of these three steps: (1) Find domain partitionings and generate the constraint abstraction function Γ, (2) perform message-passing with abstract messages computed from Γ-abstracted constraints, and (3) find the optimal solution(s) using the abstract messages as heuristic in an A* search.

The current implementation of the domain size adaptation finds partitionings through a rather simple greedy approach, which adapts message sizes locally for each cluster. An improvement could be to use linear optimization for this step instead, which will be explained in the next sub section. Also, only a single message is computed per cluster and only a single abstraction function Γ is formed for all constraints.

Algorithm 1 greedy algorithm GreedyAF to compute abstraction function Γ

function GREEDYAF(tree decomposition $\langle T, \chi, \psi \rangle$, COP $\langle X, D, F \rangle$, size limit T_{max})
  $maxDoms \leftarrow$ a mapping from variables X to their respective domain sizes $|D|$
  for $sep(u, v) \in T$ **do**
    $t \leftarrow \prod_{x \in sep(u,v)} maxDoms(x)$ // The size of this cluster's message
    if $t > T_{max}$ **then**
      for $x \in sep(u, v)$ **do**
        $maxDoms(x) := min(maxDoms(x), \lfloor \sqrt[|sep(u,v)|]{T_{max}} \rfloor)$
      end for
    end if
  end for
  $\tau \leftarrow \emptyset$
  for $x \in X$ **do**
    $\tau_x \leftarrow \begin{cases} \tau^{maxDoms(x)} & |D_x| > maxDoms(x) \\ \tau^{ID} & |D_x| \leq maxDoms(x) \end{cases}$
    $\tau \leftarrow$ add τ_x to τ
    create Γ according to the set of domain abstraction functions τ
  end for
  return Γ
end function

2.4 Domain size adaptation as linear optimization problem

Our aim is not only to reduce domain sizes, but to reduce them to match a given size limit T_{max} as exact as possible, while staying strictly below it. The adaptation of domains to the limit can be formulated as an optimization problem with the objective to maximize the number of partition elements for each

Algorithm 2 Pseudo code for the BEDA algorithm.

function BEDA(tree decomposition $\langle T, \chi, \psi \rangle$, COP $\langle X, D, F \rangle$, size limit $Tmax$)
  Γ := GreedyAF($< T, \chi, \psi >$, $< X, D, F >$, T_{max}) // Compute the constraint abstraction function Γ
  for node $u \in T$ **do**
    if u has received messages from all adjacent nodes other than v **then**
      compute the abstraction of all constraints in $\psi(u)$ by applying Γ: $\Gamma(\psi(u))$
      $m^{(\alpha)}_{(u,v)} = \Downarrow_{sep(u,v)} \left(\bigotimes_{f^{(\alpha)} \in \Gamma(\psi(u)) \cup \{m^{(\alpha)}_{(i,u)} | (i,u) \in T, i \neq v\}} f^{(\alpha)} \right)$
      send $m^{(\alpha)}_{(u,v)}$
    end if
  end for
  generate optimal solution *sol* through A* search using the messages $m^{(\alpha)}_{(u,v)} \in T$ as heuristic
  return *sol*, T
end function

domain while not harming T_{max} for any of the messages. An according domain abstraction function $\tau^{d_i^{(\alpha)}}$ is then formed for each domain D_i. $d_i^{(\alpha)}$ is the desired size of the abstract domain, and $\tau^{d_i^{(\alpha)}}$ is a function which creates exactly $d_i^{(\alpha)}$ partition elements. An example for such a function is $\tau^4(x) := x \, mod \, 4$. The objective of the optimization is then accordingly to maximize $d_i^{(\alpha)}$ for all domains D_i. Together, the functions $\tau^{d_i^{(\alpha)}}$ then induce the function Γ which is applied to all constraints during message-passing.

The potential for improvement over the current implementation of domain size adaptation, GreedyAF, is illustrated in the following example:

Example 1. Consider a COP $\mathcal{R}$ with three variables A, B, C and a tree decomposition of $\mathcal{R}$ with three nodes $\{v_1, v_2, v_3\}$, where nodes v_1, v_2 and v_2, v_3 are connected. The according χ-labels are $\chi(v_1) = \{A, B, C\}$, $\chi(v_2) = \{A, B\}$, $\chi(v_3) = \{A\}$. For this example, we can ignore the constraints of $\mathcal{R}$. Let the domain sizes be $d_A = 12$, $d_B = 2$, $d_C = 8$. Then two message sizes are given through the two separators $sep(v_1, v_2) = \{A, B\}$ and $sep(v_2, v_3) = \{A\}$: $|m_{(v_1,v_2)}| = d_A \cdot d_B = 24$, $|m_{(v_2,v_3)}| = d_A = 12$. Let further $T_{max} = 15$. The first message violates the size limit, and the greedy adaptation yields $d_A^{(\alpha)} = d_B^{(\alpha)} = \left\lfloor \sqrt[|sep(v_1,v_2)|]{15} \right\rfloor = \left\lfloor \sqrt[2]{15} \right\rfloor = 3$ as sizes for $D_A^{(\alpha)}$ and $D_B^{(\alpha)}$.

Two disadvantages of the greedy approach become obvious: (1) the size of message $m^{(\alpha)}_{(v_1,v_2)}$ is $d_A^{(\alpha)} \cdot d_B = 3 * 2 = 6$ [2], where it would have sufficed to, e.g., have $d_A = 7$ in order to keep the size limit and have a bigger (and thus more informative) message $m^{(\alpha)}_{(v1,v2)}$, and (2) the greedy approach may compute abstract

[2] If the computed abstract domain size is actually bigger than the original size, the original size is retained because $\tau^k(x) = x \, mod \, k$ preserves all domains with size $\leq k$.

domain sizes which are *bigger* than the concrete ones. The latter case may occur if domains vary largely in size.

Due to these weaknesses it is worth looking at methods for adaptation which find a globally optimal or near optimal adaptation. We hope to achieve this by formulating the problem of domain size adaptation as a linear optimization problem (LOP) with relaxed variables, and solve it using standard solvers like lp_solve[3] or GLPK[4]. The relaxation occurs because the abstract domain sizes $d_i^{(\alpha)}$, which are the variables of the LOP, are represented as real values.

Let $sep(e)$ be the separator $sep(u, v)$ associated with edge $e \in E$ connecting nodes $u, v \in V$. Then the objective of having as big abstract domains as possible while not harming T_{max} is captured by the following optimization problem: Maximize, for each edge e_j, the product of all abstract domain sizes for all variables in $sep(e_j)$ while staying below the global message size limit T_{max}:

$$\begin{aligned} \forall\, e_j : max & \prod_{X_i \in sep(e_j)} d_i^{(\alpha)} \\ \forall\, e_j :\ T_{max} & \geq \prod_{X_i \in sep(e_j)} d_i^{(\alpha)} \\ \forall i = 1..n :\ d_i & \geq d_i^{(\alpha)} \end{aligned}$$

We assume that there are p edges and thus separators. The inequalities $d_i \geq d_i^{(\alpha)}$ represent the desire to have abstract sizes which are actually smaller or equal than their concrete counterpart. The problem can be reformulated into a linear optimization problem by applying log to the equations, i.e.

$$T_{max} \geq \prod_{X_i \in sep(e_j)} d_i^{(\alpha)} \Leftrightarrow log\, T_{max} \geq \sum_{X_i \in sep(e_j)} log\, d_i^{(\alpha)}$$

for all edges e_j. This requires the above mentioned relaxation of integer variables to real variables. For brevity we set $ld_i := log\, d_i^{(\alpha)}$. Now, since $max(\sum) = \sum(max)$ the p maximization criteria can be rewritten as a single one:

$$max \sum_{X_i \in sep(e_1)} ld_i + \sum_{X_i \in sep(e_2)} ld_i + \cdots + \sum_{X_i \in sep(e_p)} ld_i$$

A single domain size $d_i^{(\alpha)}$ may influence more than one message, since variable X_i can occur multiple times in various separators. This means, if variable X_i appears in the separators for edges e_1, e_2, e_3, then its according domain size $d_i^{(\alpha)}$ occurs three times in the above maximization criterion. With the factors λ_i denoting how often a variable X_i appears in all separators, we have the final

[3] `http://lpsolve.sourceforge.net/`
[4] `http://www.gnu.org/software/glpk/`

LOP:

$$max \quad \lambda_1 ld_1 + \lambda_2 ld_2 + \cdots + \lambda_n ld_n$$

$$\begin{aligned} e_1 :\ log\, T_{max} &\geq \sum_{X_i \in sep(e_1)} ld_i \\ &\vdots \\ e_p :\ log\, T_{max} &\geq \sum_{X_i \in sep(e_p)} ld_i \\ \forall i = 1..n :\ log\, d_i &\geq log\, d_i^{(\alpha)} \end{aligned}$$

After solving this LOP one can retrieve the abstract domain sizes via $d_i^{(\alpha)} = \left\lfloor b^{ld_i} \right\rfloor$, where b is the basis of the chosen logarithm. Note that due to the relaxation from integer values (domain sizes) to real values for the LOP the found solution is only near optimal, i.e. the found domains sizes may be smaller than the optimal solution. It's not clear yet how much overhead formulating and solving this LOP imposes. However, this computation step can also be taken off-line.

3 Experimental Results and Discussion

Preliminary results were obtained by empirically testing BEDA against MCTE on a diagnosis problem, two simple block-world planning problems and a radio link frequency assignment problem (RLFAP) generated with GRAPH [13]. The tests were run on a BEDA-Implementation employing greedy adaptation of abstract domains and using domain abstraction functions of the form $\tau^d(x) := x \bmod d$. The diagnosis problem is one derived from a diagnostic scenario of the EO-1 satellite [6], the other problems were taken from the benchmark library which comes with Toolbar.

The diagrams in figure 2 either show the approximation quality or the time consumption (Y-axis) versus the imposed message size limit (X-axis). The approximated messages where used as heuristic in a subsequent A* search for the optimal solution, except for RLFAP. The A* space consumption indicates the quality of the heuristic and therefore also the approximation quality. In case of RLFAP the lower bound found by BEDA/MCTE is used as quality indicator. Note that in the former case lower values mean better quality, while it's the opposite in the latter case. Finally, the solid and dotted lines show the performance of MCTE and BEDA, respectively.

Results from the test on the EO-1 diagnosis problem, depicted in figure 2 (1a), appear promising. Clearly, the larger the size limit, the better the heuristic approximation achieved during the message-passing phase, and thus the lower the searches space consumption. This tendency can be observed both for MCTE and BEDA. However, for MCTE, the message size proceeds only in relatively coarse steps, as it can only indirectly be controlled through the parameter i

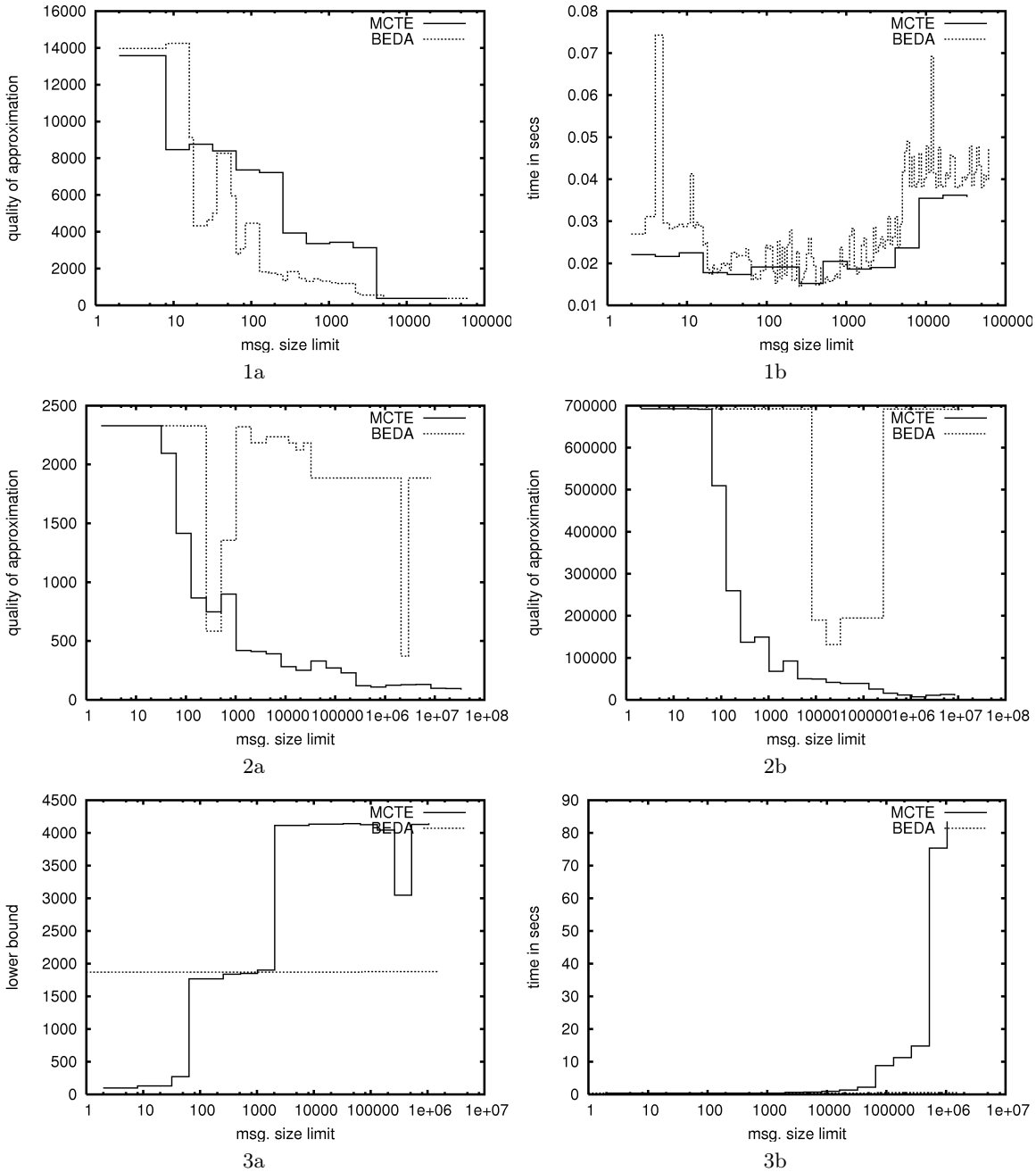


Fig. 2. (1a,b) Result of running MCTE and BEDA on a diagnosis problem from NASA satellite EO1. Size: 73 variables, domain sizes 2-9, 45 constraints with arity 2-6. The image to the right shows the time consumed by MCTE and BEDA and the subsequent A* search. (2a,b) Result of running MCTE and BEDA on bwt3c.wcsp, left hand side, and bwt4cc.wcsp, right hand side. The bwt3c problem has 45 Variables with domain sizes 2-11 and 685 binary constraints, the bwt4cc problem 179 variables with domain sizes 2-18 and 7109 binary constraints. (3a,b) Result of running MCTE and BEDA on graph07.wcsp from the CELAR benchmark suite. Size: 141 variables with domain sizes 6-44 and 2213 binary constraints.

(maximum number of variables in a mini-cluster). In comparison, BEDA enables finer control of the message size, potentially better adapting to the size limit imposed by, e.g., an embedded system. Note also that for this example, at a certain minimum size of the messages, BEDA yields a better heuristic than the mini-cluster approximation, given the same message size limit. This is intriguing considering that BEDA sends *fewer* messages per cluster than MCTE. We attribute this to the fact that at least in this real-world example, the approximation of a cluster message through local combinations of its soft-constraints as in MCTE is less informative than a global (though coarse) approximation as used in BEDA. Moreover, the positive results are even more interesting considering the relative weak greedy adaptation method.

The diagram in (1b) shows the time consumed by both approaches and the subsequent A* search. As can be seen here the time for computing and passing along the BEDA messages was comparable to or even smaller than for MCTE, that is, computing the domain abstractions did not incur significant overhead over forming the mini-clusters.

BEDA shows worse performance than MCTE in the results for planning problems bwt3c and bwt4cc (diagrams (2a,b) in figure 2, respectively) and for the RLFAP problem (diagrams (3a,b) for approximation quality and time consumption). Two points, however, might indicate additional potential: First, in bwt3c, with size limit roughly 200 BEDA seems to perform slightly better than MCTE. And second, for very strict size limits (even as low as one tuple) BEDA finds a better lower bound for the RLFAP problem than MCTE. An explanation for the latter point might be that when imposing very strict limits as in this case, MCTE must omit nearly all constraints, while BEDA at least keeps a very coarse one-tuple abstraction. This seems to save more information in this case.

Limits of the method are indicated through the overall poor performance compared to MCTE on the block-world and RLFAP problems. Three points may explain the poor performance: First, the implementation uses the greedy approach for domain size adaptation (which was shown to be weak) and it employs domain abstraction functions ($\tau^d(x) := x \, mod \, d$) which do not take into account the cost structure. This may greatly harm performance since, e.g., tuples with very low and very high costs might be combined. Second, the block-world and RLFAP consist exclusively of binary constraints. As mentioned we see BEDA's potential in problems with big constraints (> 4 variables), and these results indicate that MCTE is the better choice for problems with binary constraints. And third, in case of BEDA only a single (size limited) message was computed per cluster, while with MCTE the number of (size limited) messages was not restricted.

4 Related Work and Conclusion

In this paper we presented an approach to constraint optimization in the domain of embedded diagnosis and planning, based on the message-passing soft constraint algorithm MCTE. The novel method, called Bucket Elimination with

Domain Abstraction, addresses the problem of adapting approximation techniques used in constraint optimization to reduce complexity. Current approaches like MCTE provide such an approximation, but are however not flexible enough. This can lead to big jumps in the adaptation of the approximation to given resource limits. The new method extends MCTE by combining the message-passing scheme with adaptive domain abstraction. Test results show the methods potential especially for diagnosis problems, but indicate limitations on problems with binary constraints.

The idea of constraint abstraction is not new. Two frameworks for abstracting COPs have been introduced by [14] and [15]. The authors of [14] aim to find optimal solutions more quickly via the abstraction of a semi-ring constraint network [11], while the objective of [15] is to find upper and lower bounds for the optimal solution of a valued CSP by simplifying its valuation structure [11]. Essentially, both frameworks abstract COPs by changing/simplifying the set of preference values, e.g. from $[0, 1]$ to $\{false, true\}$. Our work differs from [14] and [15] in that our abstraction changes the domains rather than the set of preference values. Specifically this means that the same operations can be applied to the concrete problem and its abstraction. The works [16] and [17] formulate abstraction schemes for aggregating variable values. However, the authors of [16] formulate a framework for classical CSPs[5], while we focus on constraint optimization. [17] is very close to our approach, the main difference is that we focus on adapting the abstraction to given resource limits within the context of embedded systems.

For future work it remains to conduct further tests on other problems and do a rigorous complexity analysis to reveal the potential of the approach. Furthermore, we hope to improve BEDA performance by employing linear optimization for the domain size adaptation step. This direction seems promising since LP solving is a big improvement over the current very simple greedy approach. Also, we are working on more refined strategies for the automatic adaptation of the abstraction. A particular promising direction is iterative refinement of the abstractions, based on ideas presented in [17,8,9]. Our main goal here is to achieve better utilization of embedded resources, by enabling more fine-grained control over the message size in message-passing algorithms. Another goal in embedded diagnosis and planning is to focus the computation on the best solutions only, as in this application context the controller typically needs to know only a few best solutions [18]. Our abstraction-based approach can be helpful in this respect also, as we can easily bias the computation of the abstraction: the idea is to use a more fine-grained resolution for values with high utility, and a more coarse-grained resolution for values with lower utility. We are currently experimenting with such biased abstraction strategies for embedded diagnosis and planning applications.

[5] The authors claim, however, that their framework is quite easily extendable to constraint optimization. It could be that such a hypothetical extension would than also account for our special case of domain abstraction.

References

1. Beetz, M., Buss, M., Wollherr, D.: Cognitive technical systems - what is the role of artificial intelligence? In: KI 2007: Advances in Artificial Intelligence. (2007) 19–42
2. Sachenbacher, M., Williams, B.: Diagnosis as semiring-based constraint optimization. In: Proceedings of the European Conference on Artificial Intelligence (ECAI-04), Valencia, Spain (2004)
3. Dechter, R.: Constraint Processing. Morgan Kaufmann Publishers, San Francisco, CA 94104-3205 (2003)
4. Fattah, Y.E., Dechter, R.: Diagnosing tree-decomposable circuits. In: IJCAI. (1995) 1742–1749
5. Kask, K., Dechter, R.: Mini-bucket heuristics for improved search. In: Proceedings of the 15th Annual Conference on Uncertainty in Artificial Intelligence (UAI-99), San Francisco, CA, Morgan Kaufmann (1999) 314–32
6. Hayden, S.C., Sweet, A.J., Christa, S.E.: Livingstone Model-Based Diagnosis of Earth Observing One. In: AIAA 1st Intelligent Systems Technical Conference. (September 2004)
7. Maier, P., Sachenbacher, M.: Constraint optimization and abstraction for embedded intelligent systems. In Perron, L., Trick, M.A., eds.: CPAIOR. Volume 5015 of LNCS., Springer (2008) 338–342
8. Sturtevant, N.R., Jansen, R.: An analysis of map-based abstraction and refinement. In: Symposium on Abstraction, Reformulation and Approximation. (2007) 344–358
9. Hernadvolgyi, I., Holte, R.: Steps towards the automatic creation of search heuristics. technical report tr04-02, Computing Science Department, University of Alberta (2004)
10. Bouveret, S., Heras, F., Givry, S., Larrosa, J., Sanchez, M., Schiex, T.: Toolbar: a state-of-the-art platform for wcsp. `http://www.inra.fr/mia/T/degivry/ToolBar.pdf` (2004)
11. Francesca Rossi, Peter van Beek, Toby Walsh, eds.: Handbook of Constraint Programming. Foundations of Artificial Intelligence. Elsevier (2006)
12. Kask, K., Dechter, R., Larrosa, J., Cozman, F.: Bucket-Tree Elimination for Automated Reasoning. Artifical Intelligence **125** (2001) 91–131
13. van Benthem, H.: GRAPH: Generating Radiolink frequency Assignment Problems Heuristically. Master's thesis, Delft University of Technology, Faculty of Technical Mathematics and Informatics, Delft, The Netherlands. (1995)
14. Bistarelli, S., Codognet, P., Rossi, F.: An abstraction framework for soft constraints and its relationship with constraint propagation. In: Proceedings of the 4th International Symposium on Abstraction, Reformulation, and Approximation. Volume 1864 of LNCS., Springer-Verlag (2000) 71–86
15. de Givry, S., Verfaillie, G., Schiex, T.: Bounding the optimum of constraint optimization problems. In: Principles and Practice of Constraint Programming. (1997) 405–419
16. Lecoutre, C., Merchez, S., Boussemart, F., Grégoire, E.: A CSP abstraction framework. In: Proceedings of the 4th International Symposium on Abstraction, Reformulation, and Approximation. Volume 1864 of LNCS., Springer-Verlag (2000) 164–184
17. Koster, A.M.C.A.: Frequency Assignment—Models and Algorithms. PhD thesis, Universiteit Maastricht, Maastricht, The Netherlands (1999)
18. O'Sullivan, B., Provan, G.M.: Approximate compilation for embedded model-based reasoning. In: AAAI. (2006)