Trading Off Solution Quality for Faster Computation in DCOP Search Algorithms*

William Yeoh Xiaoxun Sun Sven Koenig
Computer Science Department
University of Southern California
Los Angeles, CA 90089-0781, USA
{wyeoh, xiaoxuns, skoenig}@usc.edu

Abstract

Distributed Constraint Optimization (DCOP) is a key technique for solving agent coordination problems. Because finding cost-minimal DCOP solutions is NP-hard, it is important to develop mechanisms for DCOP search algorithms that trade off their solution costs for smaller runtimes. However, existing tradeoff mechanisms do not provide relative error bounds. In this paper, we introduce three tradeoff mechanisms that provide such bounds, namely the Relative Error Mechanism, the Uniformly Weighted Heuristics Mechanism and the Non-Uniformly Weighted Heuristics Mechanism, for two DCOP algorithms, namely ADOPT and BnB-ADOPT. Our experimental results show that the Relative Error Mechanism generally dominates the other two tradeoff mechanisms for ADOPT and the Uniformly Weighted Heuristics Mechanism generally dominates the other two tradeoff mechanisms for BnB-ADOPT.

Introduction

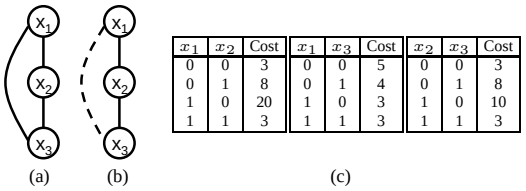
Many agent coordination problems can be modeled as Distributed Constraint Optimization (DCOP) problems, including the scheduling of meetings (Maheswaran et al. 2004), the allocation of targets to sensors in sensor networks (Ali, Koenig, and Tambe 2005) and the coordination of traffic lights (Junges and Bazzan 2008). Complete DCOP algorithms, such as ADOPT (Modi et al. 2005), find globally optimal DCOP solutions but have a large runtime, while incomplete DCOP algorithms, such as DBA (Zhang et al. 2005), find only locally optimal DCOP solutions but have a significantly smaller runtime. Because finding optimal DCOP solutions is NP-hard (Modi et al. 2005), it is important to develop mechanisms for DCOP algorithms that trade off their solution costs for smaller runtimes. Some complete DCOP algorithms, for example, allow users to specify an error bound on the solution cost. ADOPT is an example. Some incomplete DCOP algorithms allow users to specify the size k of the locally optimal groups. These DCOP algorithms partition the DCOP problem into groups of at most k agents and guarantee that their DCOP solution is optimal within these groups. The class of k-optimal algorithms (Pearce and Tambe 2007) is an example. However, efficient implementations for k-optimal algorithms are so far known only for $k \leq 3$ (Bowring et al. 2008).

We therefore seek to improve the tradeoff mechanisms of a subclass of complete DCOP algorithms, namely complete DCOP search algorithms. ADOPT is, to the best of our knowledge, the only complete DCOP search algorithm with such a tradeoff mechanism. Its Absolute Error Mechanism allows users to specify absolute error bounds on the solution costs, for example that the solution costs should be at most 10 larger than minimal. The downside of this tradeoff mechanism is that it is impossible to set relative error bounds, for example that the solution costs should be at most 10 percent larger than minimal, without knowing the optimal solution costs. In this paper, we therefore introduce three tradeoff mechanisms that provide such bounds, namely the Relative Error Mechanism, the Uniformly Weighted Heuristics Mechanism and the Non-Uniformly Weighted Heuristics Mechanism, for two complete DCOP algorithms, namely ADOPT and BnB-ADOPT (Yeoh, Felner, and Koenig 2008). BnB-ADOPT is a variant of ADOPT that uses a depth-first branch-and-bound search strategy instead of a best-first search strategy and has been shown to be faster than ADOPT on several DCOP problems (Yeoh, Felner, and Koenig 2008). Our experimental results on graph coloring, sensor network scheduling and meeting scheduling problems show that the Relative Error Mechanism generally dominates the other two tradeoff mechanisms for ADOPT and the Uniformly Weighted Heuristics Mechanism generally dominates the other two tradeoff mechanisms for BnB-ADOPT.

DCOP Problems

A DCOP problem is defined by a finite set of agents (or, synonymously, variables) $X = \{x_1, x_2, ..., x_n\}$; a set of finite domains $D = \{D_1, D_2, ..., D_n\}$, where domain D_i is the set of possible values of agent $x_i \in X$; and a set of

*This material is based upon work supported by, or in part by, the U.S. Army Research Laboratory and the U.S. Army Research Office under contract/grant number W911NF-08-1-0468 and by NSF under contract 0413196. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the sponsoring organizations, agencies, companies or the U.S. government. A version of this paper will appear in IJCAI 2009.



x_1	x_2	Cost
0	0	3
0	1	8
1	0	20
1	1	3

x_1	x_3	Cost
0	0	5
0	1	4
1	0	3
1	1	3

x_2	x_3	Cost
0	0	3
0	1	8
1	0	10
1	1	3

(c)

Figure 1: Example DCOP Problem

binary constraints $F = \{f_1, f_2, ..., f_m\}$, where constraint $f_i : D_{i_1} \times D_{i_2} \rightarrow \mathbb{R}^+ \cup \infty$ specifies its non-negative constraint cost as a function of the values of distinct agents $x_{i_1}, x_{i_2} \in X$ that share the constraint.[1] Each agent assigns itself repeatedly a value from its domain. The agents coordinate their value assignments via messages that they exchange with other agents. A complete solution is an agent-value assignment for all agents, while a partial solution is an agent-value assignment for a subset of agents. The cost of a complete solution is the sum of the constraint costs of all constraints, while the cost of a partial solution is the sum of the constraint costs of all constraints shared by agents with known values in the partial solution. Solving a DCOP problem optimally means to find its cost-minimal complete solution.

Constraint Graphs and Pseudo-Trees

DCOP problems can be represented with constraint graphs whose vertices are the agents and whose edges are the constraints. ADOPT and BnB-ADOPT transform constraint graphs in a preprocessing step into pseudo-trees. Pseudo-trees are spanning trees of constraint graphs with the property that edges of the constraint graphs connect vertices only with their ancestors or descendants in the pseudo-trees. For example, Figure 1(a) shows the constraint graph of an example DCOP problem with three agents that can each assign itself the values zero or one, and Figure 1(c) shows the constraint costs. Figure 1(b) shows one possible pseudo-tree. The dotted line is part of the constraint graph but not the pseudo-tree.

Search Trees and Heuristics

The operation of ADOPT and BnB-ADOPT can be visualized with AND/OR search trees (Marinescu and Dechter 2005). We use regular search trees and terminology from A* (Hart, Nilsson, and Raphael 1968) for our example DCOP problem since its pseudo-tree is a chain. We refer to its nodes with the identifiers shown in Figure 2(a). Its levels correspond to the agents. A left branch that enters a level means that the corresponding agent assigns itself the value zero, and a right branch means that the corresponding agent assigns itself the value one. For our example DCOP problem, the partial solution of node e is $(x_1 = 0, x_2 = 1)$. The

[1]Formulations of DCOP problems where agents are responsible for several variables each can be reduced to our formulation (Burke and Brown 2006). Similarly, formulations of DCOP problems where constraints are shared by more than two agents can be reduced to our formulation (Bacchus et al. 2002).

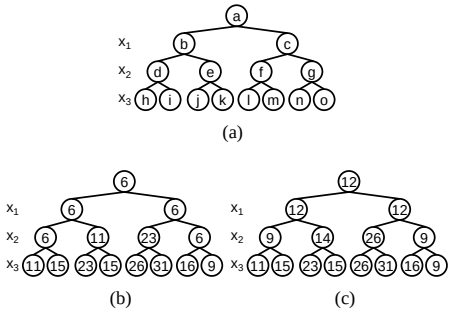


Figure 2: Search Trees for the Example

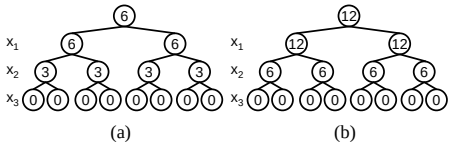


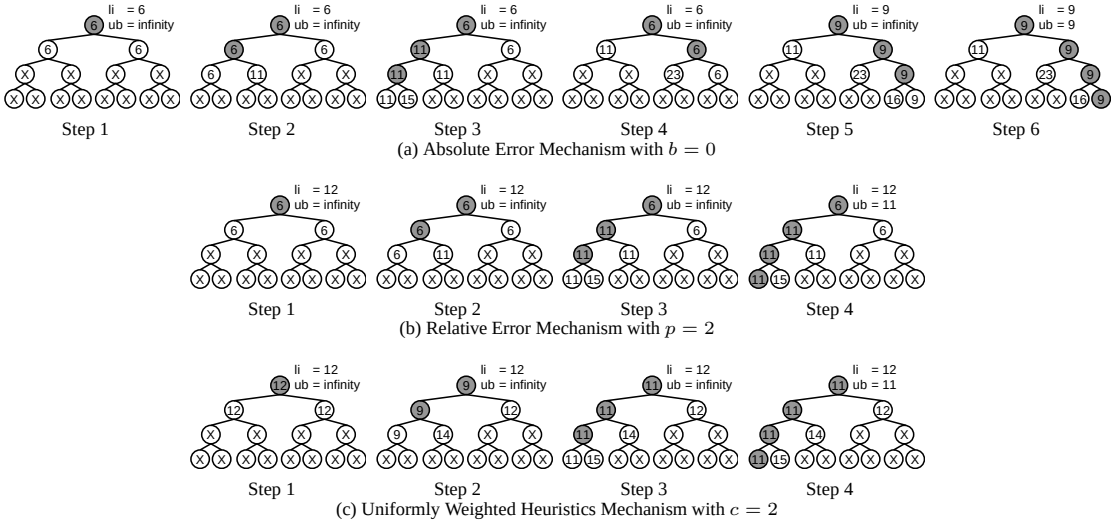
Figure 3: h-Values for the Example

f^*-value of a node is the minimal cost of any complete solution that completes the partial solution of the node. For our example DCOP problem, the f^*-value of node e is the minimum of the cost of solution $(x_1 = 0, x_2 = 1, x_3 = 0)$ [=23] and the cost of solution $(x_1 = 0, x_2 = 1, x_3 = 1)$ [=15]. Thus, the f^*-value of node e is 15. The f^*-value of the root node is the minimal solution cost. Since the f^*-values are unknown, ADOPT and BnB-ADOPT use estimated f^*-values, called f-values, during their searches. They calculate the f-value of a node by summing the costs of all constraints that involve two agents with known values and adding a user-specified h-value (heuristic) that estimates the sum of the unknown costs of the remaining constraints, similarly to how A* calculates the f-values of its nodes. For our example DCOP problem, assume that the h-value of node e is 3. Then, its f-value is 11, namely the sum of the cost of the constraint between agents x_1 and x_2 [=8] and its h-value. The ideal h-values result in f-values that are equal to the f^*-values. For our example DCOP problem, the ideal h-value of node e is $15 - 8 = 7$. Consistent h-values do not overestimate the ideal h-values. ADOPT originally used zero h-values but was later extended to use consistent h-values (Ali, Koenig, and Tambe 2005), while BnB-ADOPT was designed to use consistent h-values. We thus assume for now that the h-values are consistent.

ADOPT and BnB-ADOPT

We now give an *extremely simplistic* description of the operation of ADOPT and BnB-ADOPT to explain their search principles. For example, we assume that agents operate sequentially and information propagation is instantaneous. Complete descriptions of ADOPT and BnB-ADOPT can be found in (Modi et al. 2005; Yeoh, Felner, and Koenig 2008).

We visualize the operation of ADOPT and BnB-ADOPT on our example DCOP problem with the search trees shown in Figures 4 and 5. Unless mentioned otherwise, we use the consistent h-values from Figure 3(a), which result in the f-



(a) Absolute Error Mechanism with $b = 0$

(b) Relative Error Mechanism with $p = 2$

(c) Uniformly Weighted Heuristics Mechanism with $c = 2$

Figure 4: Simplified Execution Traces of ADOPT

values from Figure 2(b). The nodes that are being expanded and their ancestors are shaded grey.

ADOPT and BnB-ADOPT maintain lower bounds for all grey nodes and their children, shown as the numbers in the nodes. ADOPT and BnB-ADOPT initialize the lower bounds with the f-values and then always set them to the minimum of the lower bounds of the children of the nodes. Memory limitations prevent them from maintaining the lower bounds of the other nodes, shown with crosses in the nodes. ADOPT and BnB-ADOPT also maintain upper bounds, shown as ub. They always set them to the smallest costs of any complete solutions found so far. Finally, ADOPT maintains limits (usually expressed as the thresholds of the root nodes), shown as li. It always set them to b plus the maximum of the lower bounds $lb(r)$ and the f-values $f(r)$ of the root nodes r [$li := b + \max(lb(r), f(r))$], where $b \geq 0$ is a user-specified absolute error bound. For consistency, we extend BnB-ADOPT to maintain these limits as well.

ADOPT expands nodes in a depth-first search order. It always expands the child of the current node with the smallest lower bound and backtracks when the lower bounds of all unexpanded children of the current node are larger than the limits. This search order is identical to a best-first search order if one considers only nodes that ADOPT expands for the first time. BnB-ADOPT expands nodes in a depth-first branch-and-bound order. It expands the children of a node in order of their f-values and prunes those nodes whose f-values are no smaller than the upper bounds.

ADOPT and BnB-ADOPT terminate once the limits (that are equal to b plus the tightest lower bounds on the minimal solution costs) are no smaller than the upper bounds [$li \geq ub$].[2] Thus, ADOPT and BnB-ADOPT terminate with solution costs that should be at most b larger than minimal, which is why we refer to this tradeoff mechanism as the Absolute Error Mechanism. Figures 4(a) and 5(a) show execution traces of ADOPT and BnB-ADOPT, respectively, with the Absolute Error Mechanism with absolute error bound $b = 0$ for our example DCOP problem. Thus, they find the cost-minimal solution.

[2]The unextended BnB-ADOPT terminates when $lb(r) = ub$.

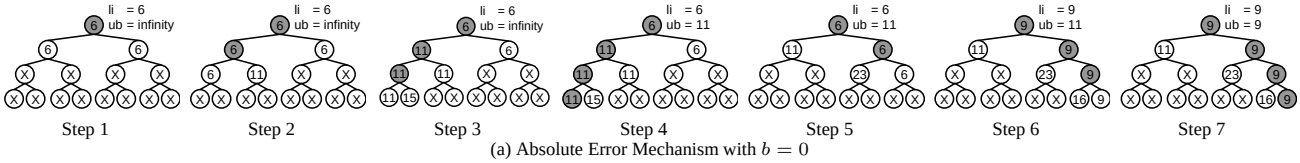
Proposed Tradeoff Mechanisms

We argued that it is often much more meaningful to specify the relative error on the solution costs than the absolute error, which cannot be done with the Absolute Error Mechanism without knowing the minimal solution costs. In this section, we introduce three new tradeoff mechanisms with this property, namely the Relative Error Mechanism, the Uniformly Weighted Heuristics Mechanism and the Non-Uniformly Weighted Heuristics Mechanism.

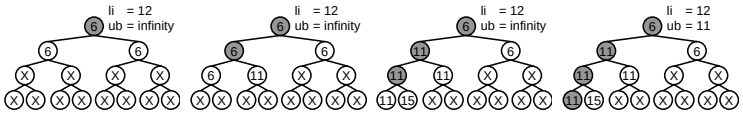
Relative Error Mechanism

We can easily change the Absolute Error Mechanism of ADOPT and BnB-ADOPT to a Relative Error Mechanism. ADOPT and BnB-ADOPT now set the limits to p times the maximum of the lower bounds $lb(r)$ and the f-values $f(r)$ of the root nodes r [$li := p \times \max(lb(r), f(r))$], where $p \geq 1$ is a user-specified relative error bound. ADOPT and BnB-ADOPT still terminate once the limits (that are now equal to p times the tightest lower bounds on the minimal solution costs) are no smaller than the upper bounds. Thus, although currently unproven, they should terminate with solution costs that are at most p times larger than minimal or, equivalently, at most $(p-1) \times 100$ percent larger than minimal, which is why we refer to this tradeoff mechanism as the Relative Error Mechanism. The guarantee of the Relative Error Mechanism with relative error bound p is thus similar to the guarantee of the Absolute Error Mechanism with an absolute error bound b that is equal to $p - 1$ times the minimal solution cost, except that the user does not need to know the minimal solution cost.

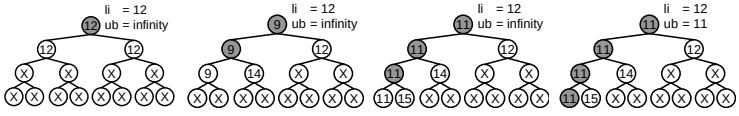
Figures 4(b) and 5(b) show execution traces of ADOPT and BnB-ADOPT, respectively, with the Relative Error



(a) Absolute Error Mechanism with $b = 0$



(b) Relative Error Mechanism with $p = 2$



(c) Uniformly Weighted Heuristics Mechanism with $c = 2$

Figure 5: Simplified Execution Traces of BnB-ADOPT

Mechanism with $p = 2$ for our example DCOP problem. For example, after ADOPT expands node d in Step 3, the lower bound [=11] of unexpanded child h of node e is no larger than the limit [=12]. ADOPT thus expands the child [=h] with the smallest lower bound in Step 4. The limit is now no smaller than the upper bound and ADOPT terminates. However, after ADOPT in Figure 4(a) expands node d in Step 3, the lower bounds of all unexpanded children of node d are larger than the limit. ADOPT backtracks repeatedly, expands node c next and terminates eventually in Step 6. Thus, ADOPT with the Relative Error Mechanism with relative error bound $p = 2$ terminates two steps earlier than in Figure 4(a) but with a solution cost that is 2 larger.

Uniformly Weighted Heuristics Mechanism

The h-values should be as close as possible to the ideal h-values to minimize the runtimes of ADOPT and BnB-ADOPT. We therefore multiply consistent h-values with a user-specified constant weight $c \geq 1$, which can result in them no longer being consistent, similar to what others have done in the context of A* where they could prove that A* is then no longer guaranteed to find cost-minimal solutions but is still guaranteed to find solutions whose costs are at most c times larger than minimal (Pohl 1970). ADOPT and BnB-ADOPT use no error bounds, that is, either the Absolute Error Mechanism with absolute error bound $b = 0$ or the Relative Error Mechanism with relative error bound $p = 1$. They terminate once the lower bounds of the root nodes (that can now be at most c times larger than the minimal solution costs and thus, despite their name, are no longer lower bounds on the minimal solution costs) are no smaller than the upper bounds. Thus, although currently unproven, ADOPT and BnB-ADOPT should terminate with solution costs that are at most c times larger than minimal. Therefore, the Uniformly Weighted Heuristics Mechanism has similar advantages as the Relative Error Mechanism but achieves them differently. The Uniformly Weighted Heuristics Mechanism inflates the lower bounds of branches of the search trees that

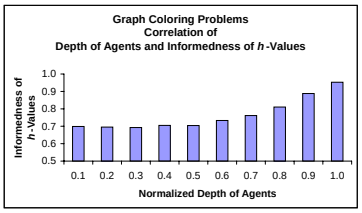


Figure 6: Depth of Agents vs. Informedness of h-Values

are yet to be explored and thus makes them appear to be less promising, while the Relative Error Mechanism prunes all remaining branches once the early termination condition is satisfied.

Figures 4(c) and 5(c) show execution traces of ADOPT and BnB-ADOPT, respectively, with the Uniformly Weighted Heuristics Mechanism with constant weight $c = 2$ for our example DCOP problem. Figure 3(b) shows the corresponding h-values, and Figure 2(c) shows the corresponding f-values. ADOPT terminates two steps earlier than in Figure 4(a) but with a solution cost that is 2 larger.

Non-Uniformly Weighted Heuristics Mechanism

The h-values of agents higher up in the pseudo-tree are often less informed than the h-values of agents lower in the pseudo-tree. The informedness of h-values is defined as the ratio of the h-values and the ideal h-values. We run experiments using the same experimental formulation and setup as (Maheswaran et al. 2004; Yeoh, Felner, and Koenig 2008) on graph coloring problems with 10 agents/vertices, density 2 and domain cardinality 3 to confirm this correlation. We use the preprocessing framework DP2 (Ali, Koenig, and Tambe 2005), that calculates the h-values by solving relaxed DCOP problems (that result from ignoring backedges) with a dynamic programming approach. DP2 was developed in the context of ADOPT but applies unchanged to BnB-ADOPT as well. Figure 6 shows the re-

	ADOPT											BnB-ADOPT										
Relative Error Bound	1.0	1.2	1.4	1.6	1.8	2.0	2.2	2.4	2.6	2.8	3.0	1.0	1.2	1.4	1.6	1.8	2.0	2.2	2.4	2.6	2.8	3.0
AE Mechanism	508	515	547	568	571	577	577	577	577	577	577	508	518	545	569	573	579	579	579	579	579	579
RE Mechanism	508	513	543	558	571	572	577	577	577	577	577	508	515	544	559	572	573	579	579	579	579	579
UWH Mechanism	508	514	535	555	593	607	622	644	654	675	704	508	515	533	558	594	609	630	663	705	724	727
NUWH Mechanism	508	513	540	559	596	605	618	651	660	663	663	508	514	541	559	596	605	620	648	660	669	669

Sensor Network Scheduling – 9 Agents

	ADOPT											BnB-ADOPT										
Relative Error Bound	1.0	1.2	1.4	1.6	1.8	2.0	2.2	2.4	2.6	2.8	3.0	1.0	1.2	1.4	1.6	1.8	2.0	2.2	2.4	2.6	2.8	3.0
AE Mechanism	116	119	124	130	133	134	133	135	135	136	136	116	118	124	130	133	136	138	138	138	139	139
RE Mechanism	116	118	122	127	131	133	133	133	133	135	135	116	117	122	127	131	133	135	135	137	138	138
UWH Mechanism	116	119	124	130	139	144	148	154	153	155	160	116	118	126	134	142	148	153	158	156	160	165
NUWH Mechanism	116	118	125	133	141	144	148	152	159	162	165	116	118	126	135	143	148	151	156	162	163	166

Meeting Scheduling – 10 Agents

	ADOPT											BnB-ADOPT										
Relative Error Bound	1.0	1.2	1.4	1.6	1.8	2.0	2.2	2.4	2.6	2.8	3.0	1.0	1.2	1.4	1.6	1.8	2.0	2.2	2.4	2.6	2.8	3.0
AE Mechanism	54207	54256	56819	61326	64204	64380	64539	64539	64539	64539	64539	54207	54221	56018	63149	67681	69326	69732	69863	69863	69863	69863
RE Mechanism	54207	54284	54771	57381	60146	62754	63998	64525	64539	64539	64539	54207	54207	54454	56088	61277	64515	67271	68891	69231	69601	69863
UWH Mechanism	54207	54207	54944	57423	62344	64391	64792	66488	67411	67913	68473	54207	54207	54733	58410	62636	66160	66812	68253	69541	70389	70840
NUWH Mechanism	54207	54207	54697	58071	62022	64342	66065	66987	68216	68010	68481	54207	54207	54639	58443	63105	66156	67878	69483	70120	70143	70942

Graph Coloring – 10 Agents

	ADOPT											BnB-ADOPT										
Relative Error Bound	1.0	1.2	1.4	1.6	1.8	2.0	2.2	2.4	2.6	2.8	3.0	1.0	1.2	1.4	1.6	1.8	2.0	2.2	2.4	2.6	2.8	3.0
AE Mechanism	67675	67795	71744	78149	79322	79591	79591	79591	79591	79591	79591	67675	67705	71566	78770	82645	83439	83768	83768	83768	83768	83768
RE Mechanism	67675	67700	68894	73059	77387	78556	79197	79591	79591	79591	79591	67675	67691	68705	72027	77020	80160	82223	82845	83439	83768	83768
UWH Mechanism	67675	67795	68868	73084	76433	77808	79632	80747	80889	82046	83452	67675	67675	68543	71864	76812	80605	81947	82578	82824	82509	82509
NUWH Mechanism	67675	67689	69055	72684	75716	77863	78658	79431	80787	82109	82580	67675	67675	67675	67683	67878	69296	69613	70676	72036	72983	73926

Graph Coloring – 12 Agents

	ADOPT											BnB-ADOPT										
Relative Error Bound	1.0	1.2	1.4	1.6	1.8	2.0	2.2	2.4	2.6	2.8	3.0	1.0	1.2	1.4	1.6	1.8	2.0	2.2	2.4	2.6	2.8	3.0
AE Mechanism	N/A	76669	80699	89978	92415	93095	93095	93095	93095	93095	93095	76465	76465	80231	91334	96922	99717	99717	99717	99717	99717	99717
RE Mechanism	N/A	N/A	77411	82443	87841	91373	92907	93095	93095	93095	93095	76465	76465	77401	80865	87210	93726	97048	98925	99166	99717	99717
UWH Mechanism	N/A	N/A	77285	81422	86407	91124	91881	92713	94694	95195	96683	76465	76465	77064	80848	87589	92584	95716	97479	97701	97992	97958
NUWH Mechanism	N/A	N/A	77648	81513	86780	89816	91509	93413	94263	94455	95422	76465	76465	76465	76479	76837	77334	78424	79773	81234	82661	84266

Graph Coloring – 14 Agents

Table 1: Experimental Results on the Solution Costs

sults. The y-axis shows the informedness of the h-values, and the x-axis shows the normalized depth of the agents in the pseudo-tree. The informedness of the h-values indeed increases as the normalized depth of the agents increases. Pearson's correlation coefficient shows a large correlation with $\rho > 0.85$. Motivated by this insight, we multiply consistent h-values with weights that vary according to the depths of the agents, similar to what others have done in the context of A* (Pohl 1973). We set the weight of agent x_i to $1+(c-1)\times(1-d(x_i)/N)$, where c is a user-specified maximum weight, $d(x_i)$ is the depth of agent x_i in the pseudo-tree and N is the depth of the pseudo-tree. This way, the weights decrease with the depth of the agents. Everything else is the same as for the Uniformly Weighted Heuristics Mechanism. The resulting weights are no larger than the weights used by the Uniformly Weighted Heuristics Mechanism with constant weight c. Thus, although currently unproven, ADOPT and BnB-ADOPT should terminate with solution costs that are at most c times larger than minimal.

Experimental Results

We compare ADOPT and BnB-ADOPT with the Absolute Error Mechanism, the Relative Error Mechanism, the Uniformly Weighted Heuristics Mechanism and the Non-Uniformly Weighted Heuristics Mechanism. We use the DP2 preprocessing framework to generate the h-values. We run experiments using the same experimental formulation and setup as (Maheswaran et al. 2004; Yeoh, Felner, and Koenig 2008) on graph coloring problems with 10, 12 and 14 agents/vertices, density 2 and domain cardinality 3; sensor network scheduling problems with 9 agents/sensors and domain cardinality 9; and meeting scheduling problems with 10 agents/meetings and domain cardinality 9. We average the experimental results over 50 DCOP problem instances each. We measure the runtimes in cycles (Modi et al. 2005) and normalize them by dividing them by the runtimes of the same DCOP algorithm with no error bounds. We normalize the solution costs by dividing them by the minimal solution costs. We vary the relative error bounds from 1.0 to 4.0. We use the relative error bounds both as the relative error bounds for the Relative Error Mechanism, the constant weights for the Uniformly Weighted Heuristics Mechanism and the maximum weights for the Non-Uniformly Weighted Heuristics Mechanism. We pre-calculate the minimal solution costs and use them to calculate the absolute error bounds for the Absolute Error Mechanism from the relative error bounds.

Tables 1 and 2 tabulate the solution costs and runtimes of ADOPT and BnB-ADOPT with the different tradeoff mechanisms. We set the runtime limit to be 5 hours for each DCOP algorithm. Data points for DCOP algorithms that failed to terminate within this limit are labeled 'N/A' in the tables. We did not tabulate the data for all data points due to

	ADOPT											BnB-ADOPT										
Relative Error Bound	1.0	1.2	1.4	1.6	1.8	2.0	2.2	2.4	2.6	2.8	3.0	1.0	1.2	1.4	1.6	1.8	2.0	2.2	2.4	2.6	2.8	3.0
AE Mechanism	5069	74	37	14	13	13	13	13	13	13	13	431	102	38	14	13	13	13	13	13	13	13
RE Mechanism	5069	96	42	15	14	13	13	13	13	13	13	431	123	49	15	14	13	13	13	13	13	13
UWH Mechanism	5069	255	39	18	14	14	14	13	12	12	12	431	95	38	19	14	14	14	13	12	12	12
NUWH Mechanism	5069	444	70	19	15	15	14	14	14	12	12	431	118	49	20	17	16	14	14	14	12	12

Sensor Network Scheduling – 9 Agents

	ADOPT											BnB-ADOPT										
Relative Error Bound	1.0	1.2	1.4	1.6	1.8	2.0	2.2	2.4	2.6	2.8	3.0	1.0	1.2	1.4	1.6	1.8	2.0	2.2	2.4	2.6	2.8	3.0
AE Mechanism	8347	525	66	28	18	17	18	17	18	17	17	1180	578	94	39	24	19	18	17	17	17	17
RE Mechanism	8347	1022	350	58	26	20	21	20	18	17	18	1180	644	348	100	41	28	24	24	21	19	19
UWH Mechanism	8347	1482	160	29	25	20	19	19	18	18	18	1180	344	133	41	28	24	23	21	19	18	18
NUWH Mechanism	8347	2573	522	50	26	20	22	20	19	19	18	1180	485	265	68	34	26	27	22	20	20	17

Meeting Scheduling – 10 Agents

	ADOPT											BnB-ADOPT										
Relative Error Bound	1.0	1.2	1.4	1.6	1.8	2.0	2.2	2.4	2.6	2.8	3.0	1.0	1.2	1.4	1.6	1.8	2.0	2.2	2.4	2.6	2.8	3.0
AE Mechanism	17566	2606	152	31	26	21	18	18	18	18	18	703	665	269	47	21	19	19	19	19	19	19
RE Mechanism	17566	3819	1496	291	51	26	22	19	19	18	18	703	677	578	304	67	44	23	19	19	19	19
UWH Mechanism	17566	8625	2284	87	30	20	18	17	17	17	17	703	523	318	102	38	21	18	18	17	17	17
NUWH Mechanism	17566	13804	5665	808	44	22	18	18	17	17	17	703	636	487	177	48	23	18	18	18	18	17

Graph Coloring – 10 Agents

	ADOPT											BnB-ADOPT										
Relative Error Bound	1.0	1.2	1.4	1.6	1.8	2.0	2.2	2.4	2.6	2.8	3.0	1.0	1.2	1.4	1.6	1.8	2.0	2.2	2.4	2.6	2.8	3.0
AE Mechanism	42256	6499	820	36	21	21	21	21	21	21	21	1007	959	424	51	22	21	21	21	21	21	21
RE Mechanism	42256	7857	3557	1255	201	36	32	21	21	21	21	1007	983	793	476	173	64	40	27	21	21	21
UWH Mechanism	42256	18507	4556	831	84	30	21	20	19	19	19	1007	745	436	206	68	35	24	22	21	21	21
NUWH Mechanism	42256	34009	13226	3222	558	29	28	20	19	19	19	1007	834	692	536	412	315	244	185	150	117	93

Graph Coloring – 12 Agents

	ADOPT											BnB-ADOPT										
Relative Error Bound	1.0	1.2	1.4	1.6	1.8	2.0	2.2	2.4	2.6	2.8	3.0	1.0	1.2	1.4	1.6	1.8	2.0	2.2	2.4	2.6	2.8	3.0
AE Mechanism	N/A	29983	712	53	29	24	24	24	24	24	24	2048	1956	861	85	28	24	24	24	24	24	24
RE Mechanism	N/A	N/A	16234	2687	102	30	25	24	24	24	24	2048	1994	1678	883	204	41	25	24	24	24	24
UWH Mechanism	N/A	N/A	8710	956	54	27	23	22	22	22	22	2048	1355	683	254	73	30	24	23	23	23	23
NUWH Mechanism	N/A	N/A	49484	6712	79	32	22	22	22	21	21	2048	1581	1197	879	618	442	332	230	187	151	124

Graph Coloring – 14 Agents

Table 2: Experimental Results on the Runtimes

space constraints.

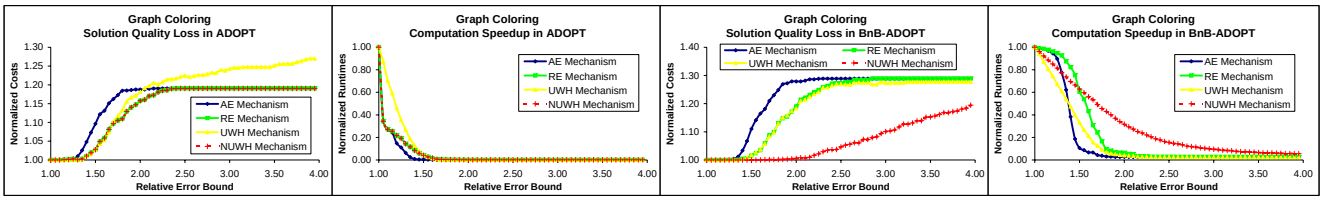
Figure 7 shows the results on the graph coloring problems with 10 agents. We do not show the results on the graph coloring problems with 12 and 14 agents, sensor network scheduling problems and meeting scheduling problems since they are similar. Figures 7(a1) and 7(b1) show that the normalized solution cost increases as the relative error bound increases, indicating that the solution quality of ADOPT and BnB-ADOPT decreases. The solution quality remains significantly better than predicted by the error bounds. For example, the normalized solution cost is less than 1.4 (rather than 3.0) when the relative error bound is 3.0.

Figures 7(a2) and 7(b2) show that the normalized runtime decreases as the relative error bound increases, indicating that ADOPT and BnB-ADOPT terminate earlier. In fact, their normalized runtime is almost zero when the relative error bound reaches about 1.5 for ADOPT and 2.0 for BnB-ADOPT.

Figure 8 plots the normalized runtime needed to achieve a given normalized solution cost. It compares ADOPT (top) and BnB-ADOPT (bottom) with the different tradeoff mechanisms on the graph coloring problems with 10 agents (left), sensor network scheduling problems (center) and meeting scheduling problems (right). For ADOPT, the Absolute Error Mechanism and the Relative Error Mechanism perform better than the other two mechanisms. However, the Relative Error Mechanism has the advantage over the Absolute Error Mechanism that relative error bounds are often more desirable than absolute error bounds. For BnB-ADOPT, on the other hand, the Uniformly Weighted Heuristics Mechanism performs better than the other three mechanisms. For example, on graph coloring problems with 10 agents, the normalized runtime needed to achieve a normalized solution cost of 1.05 is about 0.25 for the Uniformly Weighted Heuristics Mechanism, about 0.30 for the Absolute Error Mechanism, about 0.35 for the Relative Error Mechanism and about 0.40 for the Non-Uniformly Weighted Heuristics Mechanism. This trend is consistent across the three DCOP problem classes. Thus, the Uniformly Weighted Heuristics Mechanism generally dominates the other proposed or existing tradeoff mechanisms in performance and is thus the preferred choice. This is a significant result since BnB-ADOPT has been shown to be faster than ADOPT by an order of magnitude on several DCOP problems (Yeoh, Felner, and Koenig 2008) and our results allow one to speed it up even further.

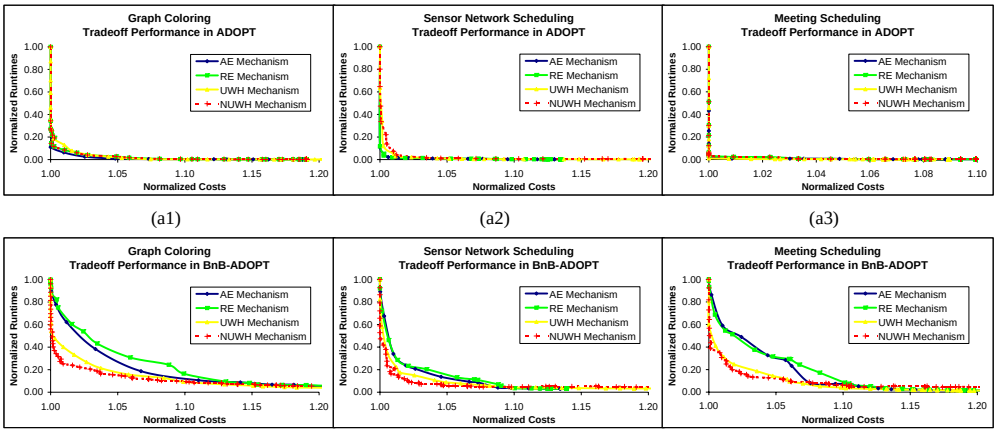
Conclusions

In this paper, we introduced three mechanisms that trade off the solution costs of DCOP algorithms for smaller runtimes, namely the Relative Error Mechanism, the Uniformly Weighted Heuristics Mechanism and the Non-Uniformly



(a1) (a2) (b1) (b2)

Figure 7: Experimental Results of ADOPT and BnB-ADOPT



(a1) (a2) (a3)

(b1) (b2) (b3)

Figure 8: Experimental Results on the Tradeoff Performance

Weighted Heuristics Mechanism. These tradeoff mechanisms provide relative error bounds and thus complement the existing Absolute Error Mechanism, that provides only absolute error bounds. For ADOPT, the Relative Error Mechanism is similar in performance to the existing tradeoff mechanism but has the advantage that relative error bounds are often more desirable than absolute error bounds. For BnB-ADOPT, the Uniformly Weighted Heuristics Mechanism generally dominates the other proposed or existing tradeoff mechanisms in performance and is thus the preferred choice. In general, we expect our tradeoff mechanisms to apply to other DCOP search algorithms as well since all of them perform search and thus benefit from using h-values to focus their searches.

References

Ali, S.; Koenig, S.; and Tambe, M. 2005. Preprocessing techniques for accelerating the DCOP algorithm ADOPT. In *Proceedings of AAMAS*, 1041–1048.

Bacchus, F.; Chen, X.; van Beek, P.; and Walsh, T. 2002. Binary vs. non-binary constraints. *Artificial Intelligence* 140(1-2):1–37.

Bowring, E.; Pearce, J.; Portway, C.; Jain, M.; and Tambe, M. 2008. On k-optimal distributed constraint optimization algorithms: New bounds and algorithms. In *Proceedings of AAMAS*, 607–614.

Burke, D., and Brown, K. 2006. Efficiently handling complex local problems in distributed constraint optimisation. In *Proceedings of ECAI*, 701–702.

Hart, P.; Nilsson, N.; and Raphael, B. 1968. A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions on Systems Science and Cybernetics* SSC4(2):100–107.

Junges, R., and Bazzan, A. 2008. Evaluating the performance of DCOP algorithms in a real world, dynamic problem. In *Proceedings of AAMAS*, 599–606.

Maheswaran, R.; Tambe, M.; Bowring, E.; Pearce, J.; and Varakantham, P. 2004. Taking DCOP to the real world: Efficient complete solutions for distributed event scheduling. In *Proceedings of AAMAS*, 310–317.

Marinescu, R., and Dechter, R. 2005. AND/OR branch-and-bound for graphical models. In *Proceedings of IJCAI*, 224–229.

Modi, P.; Shen, W.; Tambe, M.; and Yokoo, M. 2005. ADOPT: Asynchronous distributed constraint optimization with quality guarantees. *Artificial Intelligence* 161(1-2):149–180.

Pearce, J., and Tambe, M. 2007. Quality guarantees on k-optimal solutions for distributed constraint optimization problems. In *Proceedings of IJCAI*, 1446–1451.

Pohl, I. 1970. First results on the effect of error in heuristic search. *Machine Intelligence* 5:219–236.

Pohl, I. 1973. The avoidance of (relative) catastrophe, heuristic competence, genuine dynamic weighting and computational issues in heuristic problem solving. In *Proceedings of IJCAI*, 12–17.

Yeoh, W.; Felner, A.; and Koenig, S. 2008. BnB-ADOPT: An asynchronous branch-and-bound DCOP algorithm. In *Proceedings of AAMAS*, 591–598.

Zhang, W.; Wang, G.; Xing, Z.; and Wittenberg, L. 2005. Distributed stochastic search and distributed breakout: Properties, comparison and applications to constraint optimization problems in sensor networks. *Artificial Intelligence* 161(1-2):55–87.