Saving Messages in ADOPT-based Algorithms

Patricia Gutierrez, Pedro Meseguer
IIIA-CSIC
Bellaterra, Spain
patricia|pedro@iiia.csic.es
Overview

Introduction: BnB-ADOPT
messages
operation
Key Idea: Redundancy of
VALUEs
COSTs
New algorithm BnB-ADOPT+
Extending to ADOPT
Experimental Results
Conclusion
**DCOP**

**COP**: \((X, D, C)\)
- \(X = \{x_1, x_2, \ldots, x_n\}\) \hspace{1cm} variables
- \(D = \{d_1, d_2, \ldots, d_n\}\) \hspace{1cm} domains (finite)
- \(C = \{c_1, c_2, \ldots, c_r\}\) \hspace{1cm} cost functions
- \(c: d_i \times d_j \times \ldots \times d_k \rightarrow \text{NU}\{0, \infty\}\)

**DCOP**: \((X, D, C, A, \alpha)\)
- \(A = \{a_1, a_2, \ldots, a_k\}\) \hspace{1cm} agents
- \(\alpha : X \rightarrow A\) \hspace{1cm} maps variables into agents

**Solution**: total assignment with acceptable cost
\[\sum c_i(S)\]

**Optimal Sol**: solution with minimum cost
BnB-ADOPT
[Yeoh, Felner, Koenig, 08]

BnB-ADOPT:
- solves DCOPs to optimality
- correct, complete, terminates

Depth-first version of best-first ADOPT [Modi et al 05]:
- same messages
- same communication structure

At the end:
- each agent knows its value in the optimal solution
- no agent knows the whole solution
BnB-ADOPT Messages

Agents: arranged in a DFS pseudotree total order at each branch

VALUE \((i,j,val,th)\): \(i\) informs child or pseudochild \(j\) that \(i\) takes value \(val\) with threshold \(th\) (=upper bound of the cost for \(j\))

COST \((k,j,context,lb,ub)\): \(k\) informs parent \(j\) that with \(context\) its bound are \(lb\) and \(ub\)

TERMINATE \((i,j)\): \(i\) informs child \(j\) that \(i\) terminates
BnB-ADOPT: Values and Timestamps

Agent values contain a timestamp \(<val, t>\)

Timestamp:
• counter that increases when the agent selects value
• detects more recent values
• allows to update context not only by VALUE messages but also by COST messages

Example: agent \(i\)

first value \(<a,1>\)
second value \(<b,2>\)
third value \(<a,3>\) ............
**BnB-ADOPT Messages (II)**

**VALUE** \((i,j,val,th)\): \(i\) informs child or pseudochild \(j\) that \(i\) takes value \(val\) with threshold \(th\)

\(val, \text{timestamp}\)

**COST** \((k,j, context, lb, ub)\): \(k\) informs parent \(j\) that with context its bound are \(lb\) and \(ub\)

...\(i, val, \text{timestamp}, ..., j, val, \text{timestamp}, ...\)
BnB-ADOPT operation

Loop \((at \ each \ agent)\)
read and process input queue until empty
if \(lb(\text{current\_value}) \geq \text{ub}\) then
    select new_value
send a VALUE to each child
send a VALUE to each pseudochild
send a COST to parent

For each iteration, an agent sends:
- several VALUES (except leaves)
- one COST (except the root)
Redundant Messages

Intuition: a message is redundant if it can be removed without any harm for the algorithm properties (correctness, completeness, termination)

Definition: A message $msg$ from $i$ to $j$ is redundant if at some future time $t$ the collective effect of other messages that necessarily arrive to $j$ between $msg$ and $t$ would cause the same effect, so $msg$ could have been avoided.
Lemma 1

Lemma 1: If agent $i$ takes value $v_1$ with timestamp $t_1$, and the next value it takes is $v_2$ with timestamp $t_2$, there is no message with timestamp for agent $i$ between $t_1$ and $t_2$.

VALUE: since $v_1$ and $v_2$ are consecutive, there is no VALUE with timestamp between $t_1$ and $t_2$ for agent $i$.

COST: timestamps are taken from VALUE messages, so no COST may have a timestamp between $t_1$ and $t_2$ for agent $i$. 

Redundant VALUEs

Theorem 1: if \( i \) sends to \( j \) two consecutive VALUE with the same \( val \), the second is redundant.

- \( V_1 \) discarded or not:
- \( V_2 \) discarded: \( V_2 \) is redundant
- \( V_2 \) not discarded: then \((val, t_1)\) must be in \( j \) (because Lemma 1) so \( V_2 \) only changes timestamp. But any future message accepted with \( t_2 \), would also had been accepted with \( t_1 \), so \( V_2 \) is redundant.
Redundant COSTs

Theorem 2: if $k$ sends to $j$ two consecutive COST with the same content and $k$ has not detected a context change, the second is redundant.

Process of $C_1$ and $C_2$ messages:
1. Copy most recent values in the receiver context
2. If contexts (message, receiver) are compatible, update bounds

Step 1 is not essential for $C_2$: Updates timestamps only (because there is no context change in $k$)
If not done, VALUE would be accepted by $j$
So we concentrate on step 2
Redundant COSTs (II)

No matter $C_1$ is discarded or not, if $C_2$ discarded then $C_2$ is redundant

$C_1$ discarded and $C_2$ accepted (arrival to $j$ $V_1$ $C_1$ $V_2$ $C_2$)

Values $V_1$ and $V_2$ have not yet arrived to $k$

When they will arrive, more updated COST messages will be generated

So $C_2$ is redundant
Redundant COSTs (III)

$C_1$ accepted and $C_2$ accepted: two possible arrivals to $j$

- $C_1 C_2 V_1 V_2$: $C_2$ is redundant
- $C_1 V_1 V_2 C_2$: $C_1$ and $C_2$ accepted

Values $V_1$ and $V_2$ have not yet arrived to $k$

When they will arrive, more updated COST messages will be generated

So $C_2$ is redundant
New Algorithm

Applying theorems 1 and 2, we obtain a new algorithm that is correct, complete and terminates…. *but it is not efficient***!

Why? Because we have ignored the role of thresholds: upper bound of cost of each child

Solution: consider those cases where threshold improves efficiency (reinitialization of bounds, after context change)
Modification

Context change: $i$ changes value, $k$ threshold becomes $\infty$ but parent $j$ does not realize it

Threshold travels in $\text{VALUE}$: Parent $j$ has to resend $\text{VALUE}$, even if redundant

To know when redundant $\text{VALUES}$ have to be sent, a modification on $\text{COST}$ is proposed

A $\text{COST}$ from $k$ to $j$ includes a boolean $\text{ThReq}$, set to $true$ when $k$ threshold was reinitialized (otherwise $false$)

$\text{COST} (k,j,\text{context},lb,ub,\text{ThReq})$
New Algorithm: BnB-ADOPT+

BnB-ADOPT plus:

1 Agent $i$ remembers the last message sent to each neighbor

2 A $\text{COST}$ from $j$ to $i$ includes a boolean $ThReq$, set to $true$ when $j$ threshold was reinitialized

3 If $j$ has to send $i$ a $\text{COST}$ equal to the last $\text{COST}$ sent, it is send iff $j$ has detected a context change

4 If $i$ has to send $j$ a $\text{VALUE}$ equal to the last $\text{VALUE}$ sent, it is send iff the last $\text{COST}$ received from $j$ has $ThReq = true$
Extending to ADOPT

These ideas can be applied to ADOPT as well

Three changes wrt original ADOPT:
• THRESHOLD messages are included in VALUE messages
• ADOPT processes messages until input queue is empty, and then takes value
• ADOPT is modified to include timestamps, allowing COST messages to update contextes
**Experiments: Random DCOPs**

10 variables, 10 values, random costs from \{0,\ldots,100\}

UP: BnB-ADOPT  
DOWN: BnB-ADOPT+

<table>
<thead>
<tr>
<th>p1</th>
<th>messages x 10^3</th>
<th>ncccs x 10^3</th>
<th>cycles x 10^3</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1,393</td>
<td>11,003</td>
<td>53</td>
</tr>
<tr>
<td>.4</td>
<td>658</td>
<td>10,828</td>
<td>53</td>
</tr>
<tr>
<td></td>
<td>11,055</td>
<td>85,306</td>
<td>360</td>
</tr>
<tr>
<td>.5</td>
<td>4,570</td>
<td>84,383</td>
<td>360</td>
</tr>
<tr>
<td></td>
<td>68,116</td>
<td>508,186</td>
<td>1,988</td>
</tr>
<tr>
<td>.6</td>
<td>24,809</td>
<td>499,214</td>
<td>1,988</td>
</tr>
<tr>
<td></td>
<td>184,735</td>
<td>1,366,404</td>
<td>4,740</td>
</tr>
<tr>
<td>.7</td>
<td>59,900</td>
<td>1,339,303</td>
<td>4,740</td>
</tr>
<tr>
<td></td>
<td>293,923</td>
<td>2,153,777</td>
<td>6,874</td>
</tr>
<tr>
<td>.8</td>
<td>86,233</td>
<td>2,112,858</td>
<td>6,874</td>
</tr>
</tbody>
</table>
Experiments: Meeting Scheduling

UP: BnB-ADOPT / DOWN: BnB-ADOPT+

<table>
<thead>
<tr>
<th></th>
<th>#messages</th>
<th>#ncccs</th>
<th>#cycles</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>96,493</td>
<td>697,774</td>
<td>4,427</td>
</tr>
<tr>
<td>A</td>
<td>35,767</td>
<td>690,786</td>
<td>4,427</td>
</tr>
<tr>
<td></td>
<td>182,652</td>
<td>879,417</td>
<td>7,150</td>
</tr>
<tr>
<td>B</td>
<td>69,453</td>
<td>801,384</td>
<td>7,150</td>
</tr>
<tr>
<td></td>
<td>34,374</td>
<td>167,058</td>
<td>1,278</td>
</tr>
<tr>
<td>C</td>
<td>13,862</td>
<td>157,995</td>
<td>1,278</td>
</tr>
<tr>
<td></td>
<td>47,729</td>
<td>155,833</td>
<td>1,733</td>
</tr>
<tr>
<td>D</td>
<td>20,386</td>
<td>141,816</td>
<td>1,733</td>
</tr>
</tbody>
</table>
## Experiments: Sensor Networks

UP: BnB-ADOPT / DOWN: BnB-ADOPT+

<table>
<thead>
<tr>
<th></th>
<th>#messages</th>
<th>#ncccs</th>
<th>#cycles</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>7,040</td>
<td>15,097</td>
<td>226</td>
</tr>
<tr>
<td>B</td>
<td>1,074</td>
<td>14,514</td>
<td>226</td>
</tr>
<tr>
<td></td>
<td>10,258</td>
<td>23,597</td>
<td>320</td>
</tr>
<tr>
<td>B</td>
<td>1,859</td>
<td>22,659</td>
<td>320</td>
</tr>
<tr>
<td>C</td>
<td>19,563</td>
<td>118,795</td>
<td>981</td>
</tr>
<tr>
<td></td>
<td>6,236</td>
<td>116,434</td>
<td>981</td>
</tr>
<tr>
<td>D</td>
<td>56,398</td>
<td>169,784</td>
<td>1,660</td>
</tr>
<tr>
<td>D</td>
<td>17,484</td>
<td>167,658</td>
<td>1,660</td>
</tr>
</tbody>
</table>
Conclusions

• We have characterized some redundant messages in BnB-ADOPT (and ADOPT): exact algorithms for DCOP solving

• Just removing redundant messages does not extract all the potential of the approach

• Efficiency requires take into account thresholds

• Experimental results confirm a drastic decrement in the number of exchanged messages, slightly decreasing #NCCCs and keeping almost constant #cycles
Thanks for your attention!