On the Scalability and the Performance Optimization of Combinatorial Auction Solvers Using Isomorphs

Andrej Kosir

Electrical Engineering Department, University of Ljubljana, Slovenia

September 5, 2006

A joint research with Dr. Franc Brglez and Dr. Matt Stallmann.
Outline

- Problem Formulation
- Related Work
- Opcost algorithm
- Methodology and Benchmark Sets
- Experimental results
- Conclusions
Related work

2006: Cplex 9.0 package

2006: SAG2 vs Cplex 8.0

(Guo et al) Heuristics for a Bidding Problem, 'Computers and Operations Research 33, Elsevier Ltd.'

2005: CABOB vs Cplex 8.0


2002: Theoretical foundations

Combinatorial Auction / Set Packing Problem

ILP formulation (of CAP/SPP): \( a_{ij} \) and \( x_j \in \{0, 1\}; \ w_j > 0 \):

Graph-based formulation: an intersection (or conflict) graph, – a weighted MIS problem.
Combinatorial Auction / Set Packing Problem

ILP formulation (of CAP/SPP):
\[a_{ij} \text{ and } x_j \in \{0, 1\}; \ w_j > 0:\]
\[
\max \sum_{j \in \{1, \ldots, n\}} w_j x_j \quad s.t.
\]
\[
\sum_{j \in \{1, \ldots, n\}} a_{ij} x_j \leq 1, \ i \in \{1, \ldots, m\}
\]

Graph-based formulation: an intersection (or conflict) graph, – a weighted MIS problem.

\[
\begin{bmatrix}
w_1 & w_2 & w_3 & w_4 & w_5 & w_6 \\
5.0 & 4.0 & 2.0 & 6.0 & 3.0 & 1.0
\end{bmatrix}
\]

\[
\begin{bmatrix}
1 & 1 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 1 & 0 & 0 \\
0 & 1 & 1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 1 & 0 \\
0 & 0 & 1 & 0 & 1 & 0 \\
0 & 0 & 1 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 & 1 & 1
\end{bmatrix}
\]

goods (\(m = 7\))

bids (\(n = 6\))
Combinatorial Auction / Set Packing Problem

ILP formulation (of CAP/SPP):

\[ a_{ij} \text{ and } x_j \in \{0, 1\}; \ w_j > 0 : \]

\[ \max \sum_{j \in \{1, \ldots, n\}} w_j x_j \quad s.t. \]

\[ \sum_{j \in \{1, \ldots, n\}} a_{ij} x_j \leq 1, \ i \in \{1, \ldots, m\} \]

Graph-based formulation: an intersection (or conflict) graph,
– a weighted MIS problem.

graph node order = 1,2,3,4,5,6

opcost = 9

\[
\begin{matrix}
\text{w1} & \text{w2} & \text{w3} & \text{w4} & \text{w5} & \text{w6} \\
5.0 & 4.0 & 2.0 & 6.0 & 3.0 & 1.0
\end{matrix}
\]

goods 
\[(m = 7)\]

\[
\begin{bmatrix}
1 & 1 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 1 & 0 & 0 \\
0 & 1 & 1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 1 & 0 \\
0 & 0 & 1 & 0 & 1 & 0 \\
0 & 0 & 1 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 & 1 & 1
\end{bmatrix}
\]

bids 
\[(n = 6)\]

\[
\begin{matrix}
1 & 2 & 3 & 4 & 5 & 6 \\
r1: 5.0 & r2: 4.0 & r3: 2.0 & r4: 6.0 & r5: 3.0 & r6: 1.0
\end{matrix}
\]
Combinatorial Auction / Set Packing Problem

ILP formulation (of CAP/SPP):

\[ a_{ij} \text{ and } x_j \in \{0, 1\}; w_j > 0 : \]

\[
\max \sum_{j \in \{1, \ldots, n\}} w_j x_j \quad \text{s.t.} \\
\sum_{j \in \{1, \ldots, n\}} a_{ij} x_j \leq 1, \ i \in \{1, \ldots, m\}
\]

Graph-based formulation: an intersection (or conflict) graph, – a weighted MIS problem.

graph node order = 1, 2, 3, 4, 5, 6

goals (m = 7)

\[
\begin{bmatrix}
w_1 & w_2 & w_3 & w_4 & w_5 & w_6 \\
5.0 & 4.0 & 2.0 & 6.0 & 3.0 & 1.0
\end{bmatrix}
\]

bids (n = 6)

\[
\begin{bmatrix}
1 & 1 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 1 & 0 & 0 \\
0 & 1 & 1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 1 & 0 \\
0 & 0 & 1 & 0 & 1 & 0 \\
0 & 0 & 1 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 & 1 & 1
\end{bmatrix}
\]

opcost = 9

graph node order = 2, 1, 6, 4, 5, 3

opcost = 11
The opportunity cost algorithm

Notation:
– a node order $V_0$ in $G = (V, E)$ induces $G_0 = (V_0, E_0)$ (a DAG)
– write $u \rightarrow v$ if $uv \in E$
– set of predecessors of $u$: $\delta_u^-$
– set of successors of $u$: $\delta_u^+$
The opportunity cost algorithm

Notation:
- a node order \( V_0 \) in \( G = (V, E) \) induces \( G_0 = (V_0, E_0) \) (a DAG)
- write \( u \to v \) if \( uv \in E \)
- set of predecessors of \( u \): \( \delta^-_u \)
- set of successors of \( u \): \( \delta^+_u \)

Forward traversal (in order of \( V_0 \)) estimates 'node gain':
\[
f_u = w_u - \sum_{v \to u} \max(0, f_v)
\]

Backward traversal (reverse of \( V_0 \) order) finds nodes in IS:
\[
x_u = (f_u \geq 0) \forall v \in \delta^+_u : \neg x_v
\]
The opportunity cost algorithm

Notation:
- a node order $V_0$ in $G = (V, E)$ induces $G_0 = (V_0, E_0)$ (a DAG)
- write $u \rightarrow v$ if $uv \in E$
- set of predecessors of $u$: $\delta_u^-$
- set of successors of $u$: $\delta_u^+$

Forward traversal (in order of $V_0$) estimates 'node gain':

$$f_u = w_u - \sum_{v \rightarrow u} \max(0, f_v)$$

Backward traversal (reverse of $V_0$ order) finds nodes in IS:

$$x_u = (f_u \geq 0) \forall v \in \delta_u^+: \neg x_v$$

Examples to illustrate dependence on node order:
- a reference graph with natural node order
- a reference graph with permuted node order
- an isomorph of the reference graph with natural node order

... isomorphs are essential for reliable performance testing of "black-box programs" such as cplex!
Reference graph with natural node order

Opcost = 9
Reference graph with natural node order

Opcost = 9

\[ f_u = w_u - \sum_{v \rightarrow u} \max(0, f_v) \]

\[
\begin{align*}
  f_1 &= w_1 &= 5 \\
  f_2 &= w_2 - f_1 &= -1 \\
  f_3 &= w_3 &= 2 \quad (*) \\
  f_4 &= w_4 - f_1 &= 1 \\
  f_5 &= w_5 - f_3 &= 1 \quad (*) \\
  f_6 &= w_6 - f_3 - f_5 &= -2 \\
\end{align*}
\]

(*) no substractions when \( f_u < 0 \)
Reference graph with natural node order

Opcost = 9

\[ f_u = w_u - \sum_{v \rightarrow u} \max(0, f_v) \]

\[ x_u = (f_u \geq 0) \forall v \in \delta^+_u : \neg x_v \]

\begin{align*}
  f_1 &= w_1 &= 5 \\
  f_2 &= w_2 - f_1 &= -1 \\
  f_3 &= w_3 &= 2 \quad (*) \\
  f_4 &= w_4 - f_1 &= 1 \\
  f_5 &= w_5 - f_3 &= 1 \quad (*) \\
  f_6 &= w_6 - f_3 - f_5 &= -2 \\
\end{align*}

\((*)\) no substractions when \(f_u < 0\)

\begin{align*}
  b_6 &= \{\} & x_6 &= 0, f_6 < 0 \\
  b_5 &= \{5\} & x_5 &= 1, f_5 > 0 \quad (*) \\
  b_4 &= \{5, 4\} & x_4 &= 1, f_4 > 0 \quad (*) \\
  b_3 &= \{5, 4\} & x_3 &= 0, f_3 > 0 \quad (**) \\
  b_2 &= \{5, 4\} & x_2 &= 0, f_2 < 0 \\
  b_1 &= \{5, 4\} & x_1 &= 0, f_1 > 0 \quad (**) \\
\end{align*}

\((*)\) no conflict with successor in IS

\((**\) conflict with successor in IS
Reference graph with permuted node order

Opcost = 10
Reference graph with permuted node order

Opcost = 10

\[ f_u = w_u - \sum_{v \rightarrow u} \max(0, f_v) \]

\[ f_5 = w_5 \quad = 3 \]
\[ f_4 = w_4 \quad = 6 \]
\[ f_2 = w_2 - f_5 \quad = 1 \]
\[ f_6 = w_6 - f_6 \quad = -2 \]
\[ f_3 = w_3 - f_2 - f_5 \quad = -2 \quad (*) \]
\[ f_1 = w_1 - f_2 - f_4 \quad = -2 \]

(*) no substractions when \( f_u < 0 \)
Reference graph with permuted node order

Opcost = 10

\[ f_u = w_u - \sum_{v \rightarrow u} \max(0, f_v) \]

\[ x_u = (f_u \geq 0) \forall v \in \delta^+_u : \neg x_v \]

\[
\begin{align*}
    f_5 &= w_5 &= 3 \\
    f_4 &= w_4 &= 6 \\
    f_2 &= w_2 - f_5 &= 1 \\
    f_6 &= w_6 - f_6 &= -2 \\
    f_3 &= w_3 - f_2 - f_5 &= -2 (*) \\
    f_1 &= w_1 - f_2 - f_4 &= -2 \\
    b_1 &= \{\} & x_1=0, f_1 < 0 \\
    b_3 &= \{\} & x_3=0, f_3 < 0 \\
    b_6 &= \{\} & x_6=0, f_6 < 0 \\
    b_2 &= \{2\} & x_2=1, f_2 > 0 (*) \\
    b_4 &= \{2, 4\} & x_4=1, f_4 < 0 \\
    b_5 &= \{2, 4\} & x_5=0, f_5 > 0 (*)
\end{align*}
\]

(*) no substractions when \( f_u < 0 \)

(*) no conflict with successor in IS

(**) conflict with successor in IS
Isomorph graph with natural node order

Opcost = 11

... a permutation relabels node names in the reference graph:

\[ r_1 \rightarrow p_2; \quad r_2 \rightarrow p_1; \quad r_3 \rightarrow p_6; \quad r_4 \rightarrow p_4; \quad r_5 \rightarrow p_5; \quad r_6 \rightarrow p_3 \]
Isomorph graph with natural node order

Opcost = 11

... a permutation relabels node names in the reference graph:

\[ r_1 \rightarrow p_2; \ r_2 \rightarrow p_1; \ r_3 \rightarrow p_6; \ r_4 \rightarrow p_4; \ r_5 \rightarrow p_5; \ r_6 \rightarrow p_3 \]

\[ f_u = w_u - \sum_{v \rightarrow u} \max(0, f_v) \]

\[
\begin{align*}
  f_1 &= w_1 = 4 \\
  f_2 &= w_2 - f_1 = 1 \\
  f_3 &= w_3 = 1 \\
  f_4 &= w_4 - f_2 = 5 \\
  f_5 &= w_5 - f_3 - f_3 = -2 \\
  f_6 &= w_6 - f_1 - f_3 = -3 \quad (*)
\end{align*}
\]

(*) no substractions when \( f_u < 0 \)
Isomorph graph with natural node order

Opcost = 11

... a permutation relabels node names in the reference graph:

\[ r_1 \rightarrow p_2; \ r_2 \rightarrow p_1; \ r_3 \rightarrow p_6; \ r_4 \rightarrow p_4; \ r_5 \rightarrow p_5; \ r_6 \rightarrow p_3 \]

\[ f_u = w_u - \sum_{v \rightarrow u} \max(0, f_v) \]
\[ x_u = (f_u \geq 0) \forall v \in \delta^+_u : \neg x_v \]

\[
\begin{align*}
  f_1 &= w_1 &= 4 \\
  f_2 &= w_2 - f_1 &= 1 \\
  f_3 &= w_3 &= 1 \\
  f_4 &= w_4 - f_2 &= 5 \\
  f_5 &= w_5 - f_3 - f_3 &= -2 \\
  f_6 &= w_6 - f_1 - f_3 &= -3 (*)
\end{align*}
\]

\[
\begin{align*}
  b_6 &= \{\} & x_6 &= 0, \ f_6 &< 0 \\
  b_5 &= \{\} & x_5 &= 0, \ f_5 &< 0 \\
  b_4 &= \{4\} & x_4 &= 1, \ f_4 &> 0 (*) \\
  b_3 &= \{4, 3\} & x_3 &= 1, \ f_3 &> 0 (*) \\
  b_2 &= \{4, 3\} & x_2 &= 0, \ f_2 &> 0 (**) \\
  b_1 &= \{4, 3, 1\} & x_1 &= 1, \ f_1 &> 0 (*)
\end{align*}
\]

(*) no substractions when \( f_u < 0 \)

(*) no conflict with successor in IS

(**) conflict with successor in IS

– p. 8/20
Comparing opcost strategies

Strategies are "nodes order only" and "isomorphs"

Data sets: most difficult instances from each set

Hypothesis: "Objective means are equal" (strategies are equivalent)

Data and method:

- 1000 runs of each strategy on the reference and 32 isomorphs (a total of 33,000 runs for each strategy)
- t-test at the significance level $\alpha = 0.05$

<table>
<thead>
<tr>
<th>Instance</th>
<th>$p$</th>
<th>$p &gt; \alpha$</th>
</tr>
</thead>
<tbody>
<tr>
<td>in601</td>
<td>0.20</td>
<td>accepted</td>
</tr>
<tr>
<td>frb30-15-5</td>
<td>0.07</td>
<td>accepted</td>
</tr>
<tr>
<td>DSJC125.1_2000_050_Blocks</td>
<td>0.63</td>
<td>accepted</td>
</tr>
</tbody>
</table>

However, isomorphs ARE needed to test cplex!!
Methodology

- Apply ’any-time’ concepts on a reference and 32 isomorphs:
  - run cplex and opcost each instance for 16 seconds (a total of 528 seconds)
  - run cplex and opcost each instance for 32 seconds (a total of 1056 seconds)
  - run cplex and opcost each instance for 64 seconds (a total of 2112 seconds)
- For each instance and each solver
  - tabulate reported objective values and solutions
  - verify solution for feasibility and reported objective value
  - study asymptotic behavior, statistical testing and visualization strategies
Benchmark sets (1)

Results with most published benchmarks were shown to be "easy", even with cplex versions predating 9.0. Benchmarks below are "hard" for cplex 9.0.

Unit-weighted random, with optimum hidden solution

<table>
<thead>
<tr>
<th>instance</th>
<th>nodes</th>
<th>edges</th>
<th>optimum</th>
<th>MIS</th>
</tr>
</thead>
<tbody>
<tr>
<td>frb30-15-1</td>
<td>450</td>
<td>17827</td>
<td>30</td>
<td></td>
</tr>
<tr>
<td>frb30-15-2</td>
<td>450</td>
<td>17874</td>
<td>30</td>
<td></td>
</tr>
<tr>
<td>frb30-15-3</td>
<td>450</td>
<td>17809</td>
<td>30</td>
<td></td>
</tr>
<tr>
<td>frb30-15-4</td>
<td>450</td>
<td>17831</td>
<td>30</td>
<td></td>
</tr>
<tr>
<td>frb30-15-5</td>
<td>450</td>
<td>17794</td>
<td>30</td>
<td></td>
</tr>
</tbody>
</table>
Benchmark sets (1)

Results with most published benchmarks were shown to be "easy", even with cplex versions predating 9.0. Benchmarks below are "hard" for cplex 9.0.

Unit-weighted random, with optimum hidden solution

<table>
<thead>
<tr>
<th>instance</th>
<th>nodes</th>
<th>edges</th>
<th>optimum</th>
<th>MIS</th>
</tr>
</thead>
<tbody>
<tr>
<td>frb30-15-1</td>
<td>450</td>
<td>17827</td>
<td>30</td>
<td></td>
</tr>
<tr>
<td>frb30-15-2</td>
<td>450</td>
<td>17874</td>
<td>30</td>
<td></td>
</tr>
<tr>
<td>frb30-15-3</td>
<td>450</td>
<td>17809</td>
<td>30</td>
<td></td>
</tr>
<tr>
<td>frb30-15-4</td>
<td>450</td>
<td>17831</td>
<td>30</td>
<td></td>
</tr>
<tr>
<td>frb30-15-5</td>
<td>450</td>
<td>17794</td>
<td>30</td>
<td></td>
</tr>
</tbody>
</table>

Unit-weighted structured, block-diagonal, with optimum hidden solution

<table>
<thead>
<tr>
<th>instance</th>
<th>nodes</th>
<th>edges</th>
<th>optimum</th>
<th>MIS</th>
</tr>
</thead>
<tbody>
<tr>
<td>DSJC125.1_0125</td>
<td>125</td>
<td>736</td>
<td>34</td>
<td></td>
</tr>
<tr>
<td>DSJC125.1_0250_050</td>
<td>250</td>
<td>1840</td>
<td>68</td>
<td></td>
</tr>
<tr>
<td>DSJC125.1_0500_050</td>
<td>500</td>
<td>4600</td>
<td>136</td>
<td></td>
</tr>
<tr>
<td>DSJC125.1_1000_050</td>
<td>1000</td>
<td>11500</td>
<td>272</td>
<td></td>
</tr>
<tr>
<td>DSJC125.1_2000_050</td>
<td>2000</td>
<td>28750</td>
<td>544</td>
<td></td>
</tr>
</tbody>
</table>
Benchmark sets (2)

Weighted random combinatorial auctions, based on prices/preferences/fairness factors

<table>
<thead>
<tr>
<th>instance</th>
<th>bids</th>
<th>goods</th>
<th>Sum of weights</th>
</tr>
</thead>
<tbody>
<tr>
<td>in101</td>
<td>1000</td>
<td>500</td>
<td>6,988,485</td>
</tr>
<tr>
<td>in201</td>
<td>1000</td>
<td>1000</td>
<td>7,200,075</td>
</tr>
<tr>
<td>in401</td>
<td>500</td>
<td>1000</td>
<td>7,367,997</td>
</tr>
<tr>
<td>in501</td>
<td>1500</td>
<td>1000</td>
<td>29,938,788</td>
</tr>
<tr>
<td>in601</td>
<td>1500</td>
<td>1500</td>
<td>32,171,508</td>
</tr>
</tbody>
</table>
in201 at 512, 1056, 2112 seconds

Best Objective Values

Runtimes (528, 1056, 2112 seconds) for in201

- cplex-9.0
- opcost-40
objective value distributions: in201

cplex (in201, for 2112 seconds)

opcost (in201, for 2112 seconds)
in601 at 512, 1056, 2112 seconds

Best Objective Values

Runtimes (528, 1056, 2112 seconds) for in601

- cplex-9.0
- opcost-40
Hidden optima: frb30-15-5

Best Objective Values

Run times (528, 1056, 2112 seconds) for frb30-15-5

- hidden optima
- cplex-9.0
- opcost-40
Asymptotic performance: cplex vs opcost

Open challenge: reduce the performance gap!
Effectiveness of opcost

Number of calls to opcost in $33 \times 64 = 2112$ seconds:

<table>
<thead>
<tr>
<th>instance class</th>
<th>nodes</th>
<th>Calls to opcost per instance</th>
</tr>
</thead>
<tbody>
<tr>
<td>frb30-15-1</td>
<td>450</td>
<td>85,000</td>
</tr>
<tr>
<td>in201</td>
<td>1000</td>
<td>6,000</td>
</tr>
<tr>
<td>in601</td>
<td>1500</td>
<td>2,300</td>
</tr>
<tr>
<td>DSJC125.1_1000_050</td>
<td>1000</td>
<td>26,000</td>
</tr>
<tr>
<td>DSJC125.1_2000_050</td>
<td>2000</td>
<td>7,500</td>
</tr>
</tbody>
</table>
Effectiveness of opcost

Number of calls to opcost in $33 \times 64 = 2112$ seconds:

<table>
<thead>
<tr>
<th>instance class</th>
<th>nodes</th>
<th>Calls to opcost per instance</th>
</tr>
</thead>
<tbody>
<tr>
<td>frb30-15-1</td>
<td>450</td>
<td>85,000</td>
</tr>
<tr>
<td>in201</td>
<td>1000</td>
<td>6,000</td>
</tr>
<tr>
<td>in601</td>
<td>1500</td>
<td>2,300</td>
</tr>
<tr>
<td>DSJC125.1_1000_050</td>
<td>1000</td>
<td>26,000</td>
</tr>
<tr>
<td>DSJC125.1_2000_050</td>
<td>2000</td>
<td>7,500</td>
</tr>
</tbody>
</table>

With relatively low number of random permutations, we find opcost performance on par or better than cplex (... for the benchmark sets we examined).
Summary and Future Work

Summary:

- opcost algorithm performs significantly better than simpler heuristics such as mis101 (our implementation): taking advantage of an ordered graph makes a difference.

- compared to performance of cplex, special purpose solvers such as opcost may have a larger role in the future, in particular on very large instances.
Summary and Future Work

Summary:
- opcost algorithm performs significantly better than simpler heuristics such as mis101 (our implementation): taking advantage of an ordered graph makes a difference.
- compared to performance of cplex, special purpose solvers such as opcost may have a larger role in the future, in particular on very large instances.

Future Work:
- investigate node ordering strategies other than random (e.g., a version of KL-like node swapping strategy)
- randomized bit-flipping strategy to improve solutions returned by opcost.