Combinatorial optimization problems expressed as *Boolean constraint satisfaction problems* (BCSPs) arise in several contexts, ranging from the classical unate set cover/packing problems to the binate minimum cover problems, including the *Haplotype Inference by Pure Parsimony* (HIPP) problem. For years, publications on special purpose BCSP solvers have been comparing their performance to the more general-purpose LP-solver *cplex* that also takes instances of BCSPs. Not surprisingly, ‘experimental results’ of special purpose solvers would dominate the performance of *cplex* -- until a recent publication demonstrated the opposite trend on some of the instances.

In this talk we demonstrate how the scientific method can be applied to comparing the performance of not only BCSP solvers but also other solvers that address NP-hard problems. The approach is founded on two premises:

1. the introduction of instance isomorphs as families of equivalence classes, based on randomized replicas of a given reference instance, and
2. the use of isomorph classes for the design of reproducible experiments with BCSP solvers that includes performance testing hypotheses. We introduce a number of BCSP reference instances from different domains, generate isomorph classes and use *cplex* to characterize the solver performance and the isomorph classes themselves. Instances and results with supporting software utilities are posted on the Web.

By adopting the proposed methodology, solver developers may find it easier to
1. reliably improve the performance of their solvers, and,
2. report results of their experiments under the proposed schema.

This is a joint work with Jason Osborne and Matt Stallmann. 


About my curiosity about data sets ... 

... lighted some fires that predate the DIMACS series

1985 ... introduced the ISCAS’85 benchmark set, initially as ATPG instances (with Prof. Fujiwara)

1986 ... initiated bi-annual series of Layout Synthesis workshops (at MCNC) with benchmark sets

1987 ... initiated bi-annual series of Logic Synthesis workshops (at MCNC) with benchmark sets

1989 ... introduced the ISCAS’89 benchmark set, initially as FSM/ATPG instances

Still trying to correct mistakes after releasing the data sets without a sound strategy & support for experimental design ...
Experimental Computer Science: How About The Scientific Method?

Franc Brglez
NC STATE UNIVERSITY
Computer Science
Raleigh, NC, USA

Also, a question from a sponsor ...

Can one replicate these experiments?

{ Wow, what about the scientific method ???}
Yes, there are problems with experiments ...

1994 ... Needed: An Empirical Science of Algorithms
1995 ... Designing and reporting on computational experiments with heuristic methods
1996 ... Testing heuristics: We have it all wrong
1998 ... Design of Experiments to Evaluate Algorithms: Which Improvements Are Due to Improved Heuristic and Which Are Due to Chance?
2002 ... A Theoretician's Guide to the Experimental Analysis of Algorithms
2005 ... Experimental Computer Science: The Need for a Cultural Change

More “recommendations”…?

Four recommendations from WEA’2008 submission URL:

2002 ... A Theoretician's Guide to the Experimental Analysis of Algorithms (started at 7, now 36 pages)
2003 ... Algorithm Engineering (16 pages)
1999 ... How to Present a Paper on Experimental Work with Algorithms (6 pages)
20xx ... Presenting Data from Experiments in Algorithmics (16 pages)

A common thread: no hints about formalized experimental designs and the scientific method!
So what about “this scientific method”?

Should our colleagues in natural sciences have it so “easy”..

> RunMyExp RefFile ClassSize ClassID SolverIDs

```plaintext
proc GenericExperiment {RefFile, ClassSize, ClassID, SolverIDs} {
    set ClassFiles [ClassGen $RefFile $ClassSize $ClassID ]

    foreach SolverID $SolverIDs {
        foreach file $ClassFiles {
            set Results($ClassID,$SolverID,$file) \ 
                [Encap $SolverID $file]
        }
    }
    set SummaryTables [GenTables [array get Results]]
    return $SummaryTables
}
```

The key: isomorphs as sample population

- introduced for learning experiments by H. Simon (1969), and
  Tower of Hanoi isomorphs are still being published (ACCSS’2000)

- introduced for performance evaluation of algorithms:
  (in contrast to Simon, isomorph syntax is strictly invariant)

*Instance generation rules* for performance evaluation of
algorithms depend on the problem domain:

- almost trivial for many graph problems (permutation rules)
- “invented” for some problem domains (coordinate rotation)
- may be initiated within the scope of five generic rules

A single reference instance generates the entire sample population:

```plaintext
set ClassFiles [ClassGen $RefFile $ClassSize $ClassID]
```
Examples of ClassIDs \{LR, CLR\}

binate reference instance (7 vars, 6 constraints)

\[
\begin{align*}
\text{Max} & : 21x_1 + 22x_2 + 23x_3 + 25x_4 + 26x_5 + 27x_6 + 34x_7 \\
\text{st} & : \\
\ c1 & : +x_3, +x_4 \geq 1 \\
\ c2 & : -x_3, -x_4 \geq 0 \\
\ c3 & : +x_3, +x_4 \geq 0 \\
\ c4 & : -x_3, +x_7 \geq 0 \\
\ c5 & : -x_4 \geq 0 \\
\ c6 & : \geq 1 \\
\end{align*}
\]

class LR isomorph (local,row)  
class CLR isomorph (col, local,row)

Hypothesis and Design 3

Hypothesis:  
The branch-and-bound performance of any pair of solvers, formed from the table below, are equivalent.

Design:

<table>
<thead>
<tr>
<th>classes</th>
<th>class size</th>
</tr>
</thead>
<tbody>
<tr>
<td>in401_sp_CLR</td>
<td>N = 32</td>
</tr>
<tr>
<td>f51mb_350_CLR</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>solver</th>
<th>none</th>
<th>dfs</th>
<th>feas2</th>
</tr>
</thead>
<tbody>
<tr>
<td>cplex090</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>cplex101</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
</tbody>
</table>
Design 3: Results (in401_sp_CLR)

RunTime statistics

<table>
<thead>
<tr>
<th>Solver</th>
<th>Class</th>
<th>RefV</th>
<th>MinV</th>
<th>MaxV</th>
<th>MedV</th>
<th>MeanV</th>
<th>StdV</th>
<th>N</th>
<th>Distribution</th>
</tr>
</thead>
<tbody>
<tr>
<td>cplex000</td>
<td>in401_sp_CLR</td>
<td>865</td>
<td>407</td>
<td>957</td>
<td>638</td>
<td>668</td>
<td>183</td>
<td>32</td>
<td>uniform</td>
</tr>
<tr>
<td>cplex101</td>
<td>in401_sp_CLR</td>
<td>841</td>
<td>625</td>
<td>1316</td>
<td>816</td>
<td>843</td>
<td>142</td>
<td>32</td>
<td>normal</td>
</tr>
<tr>
<td>cplex099-dfs</td>
<td>in401_sp_CLR</td>
<td>798</td>
<td>411</td>
<td>748</td>
<td>574</td>
<td>576</td>
<td>85.7</td>
<td>32</td>
<td>uniform</td>
</tr>
<tr>
<td>cplex101-dfs</td>
<td>in401_sp_CLR</td>
<td>676</td>
<td>592</td>
<td>1200</td>
<td>904</td>
<td>925</td>
<td>149</td>
<td>32</td>
<td>normal</td>
</tr>
<tr>
<td>cplex099-feas2</td>
<td>in401_sp_CLR</td>
<td>431</td>
<td>321</td>
<td>493</td>
<td>413</td>
<td>416</td>
<td>38.9</td>
<td>32</td>
<td>uniform</td>
</tr>
<tr>
<td>cplex101-feas2</td>
<td>in401_sp_CLR</td>
<td>1491</td>
<td>950</td>
<td>1987</td>
<td>1496</td>
<td>1510</td>
<td>247</td>
<td>22</td>
<td>normal</td>
</tr>
</tbody>
</table>

Design 3: Results (f51mb.._CLR)

RunTime statistics

<table>
<thead>
<tr>
<th>Solver</th>
<th>Class</th>
<th>RefV</th>
<th>MinV</th>
<th>MaxV</th>
<th>MedV</th>
<th>MeanV</th>
<th>StdV</th>
<th>N</th>
<th>Distribution</th>
</tr>
</thead>
<tbody>
<tr>
<td>cplex060</td>
<td>f51mb_350_CLR</td>
<td>115</td>
<td>71.3</td>
<td>2118</td>
<td>127</td>
<td>292</td>
<td>338</td>
<td>32</td>
<td>heavy-tail</td>
</tr>
<tr>
<td>cplex101</td>
<td>f51mb_350_CLR</td>
<td>87.3</td>
<td>53.5</td>
<td>2117</td>
<td>159</td>
<td>408</td>
<td>619</td>
<td>32</td>
<td>heavy-tail</td>
</tr>
<tr>
<td>cplex060-dfs</td>
<td>f51mb_350_CLR</td>
<td>102</td>
<td>60.1</td>
<td>2117</td>
<td>225</td>
<td>388</td>
<td>526</td>
<td>32</td>
<td>near-exponential</td>
</tr>
<tr>
<td>cplex101-dfs</td>
<td>f51mb_350_CLR</td>
<td>179</td>
<td>89.2</td>
<td>2116</td>
<td>227</td>
<td>441</td>
<td>585</td>
<td>32</td>
<td>exponential</td>
</tr>
<tr>
<td>cplex060-feas2</td>
<td>f51mb_350_CLR</td>
<td>115</td>
<td>49.2</td>
<td>446</td>
<td>94.2</td>
<td>113</td>
<td>69.8</td>
<td>32</td>
<td>near-exponential</td>
</tr>
<tr>
<td>cplex101-feas2</td>
<td>f51mb_350_CLR</td>
<td>69.1</td>
<td>58.1</td>
<td>2118</td>
<td>127</td>
<td>316</td>
<td>501</td>
<td>32</td>
<td>heavy-tail</td>
</tr>
</tbody>
</table>
Conclusions

• not about whether cplex090 dominates cplex101 on BCSP instances (it appears
• but about application of "the scientific method" to assessing the performance of combinatorial solvers

(1) by designing instance isomorph classes

(2) by reporting on reproducible solver experiments on such classes

• and about making more reliable decisions whether improvements of heuristics are due to the design or due to chance!

Ongoing Work

• updates on home page of xBed: http://www.cbl.ncsu.edu/xBed/
  (Datasets, Open Experiments, and Utilities)

• complete the work on block-dominant compositions of BCSP datasets with hidden solutions, including the refinement for finer control of variable size increments ....

• create new data sets for robust designs of new and scalable heuristics ....
  (an invitation for a collaborative project)
Preconditioning for Combinatorial Solvers?

- large sparse linear systems could not be solved without instance preconditioning ...

“structured instance?” “yes, good ordering exists”

less storage; faster, more stable solution

\[ \rho = 0.277 \]
\[ \rho = 0.913 \]

Structured-vs-Unstructured instances

“structured instance” ==> “∃ good ordering”
“unstructured instance” ==> “∀ good ordering”

\[ \rho = 0.26 \]
\[ \rho = 0.22 \]
Two hypotheses, resolved via isomorphs

(1) Instances of MIS (Max-Indep-Set) problems:

An instance ordered by Fiedler permutation is “structured” if Pearson correlation coefficient $\rho > 0.5$ and “unstructured” (random) otherwise.

**Resolution:** TRUE (so far ...)

(2) MIS problem solvers:

For the same reference instance and the same solver, the isomorph class CLR is equivalent to the isomorph class CLR_Fiedler.

**Resolution:** FALSE for “better” solvers, i.e. preconditioning for MIS problem solvers matters!!

### Stats for classes of frb30-15-1 (450 vars)

<table>
<thead>
<tr>
<th>Solver</th>
<th>Precond</th>
<th>Observed</th>
<th>RefV</th>
<th>MinV</th>
<th>MaxV</th>
<th>MedV</th>
<th>MeanV</th>
<th>StdV</th>
<th>p stat</th>
</tr>
</thead>
<tbody>
<tr>
<td>MaxIS_101</td>
<td>no</td>
<td>Trials</td>
<td>6596</td>
<td>4493</td>
<td>12888</td>
<td>5728</td>
<td>6301</td>
<td>2003</td>
<td></td>
</tr>
<tr>
<td></td>
<td>yes</td>
<td>Trials</td>
<td>5971</td>
<td>5971</td>
<td>5971</td>
<td>5971</td>
<td>5971</td>
<td>0.00</td>
<td>$p = 0.000$</td>
</tr>
<tr>
<td></td>
<td>no</td>
<td>ObjBest</td>
<td>27.0</td>
<td>24.0</td>
<td>28.0</td>
<td>25.0</td>
<td>25.5</td>
<td>0.95</td>
<td></td>
</tr>
<tr>
<td></td>
<td>yes</td>
<td>ObjBest</td>
<td>26.0</td>
<td>26.0</td>
<td>26.0</td>
<td>26.0</td>
<td>26.0</td>
<td>0.00</td>
<td>$p = 0.000$</td>
</tr>
<tr>
<td>MaxIS_OpC</td>
<td>no</td>
<td>Trials</td>
<td>4785</td>
<td>4027</td>
<td>8891</td>
<td>5693</td>
<td>6172</td>
<td>1801</td>
<td></td>
</tr>
<tr>
<td></td>
<td>yes</td>
<td>Trials</td>
<td>4175</td>
<td>4175</td>
<td>4175</td>
<td>4175</td>
<td>4175</td>
<td>0.00</td>
<td>$p = 0.000$</td>
</tr>
<tr>
<td></td>
<td>no</td>
<td>ObjBest</td>
<td>27.0</td>
<td>25.0</td>
<td>28.0</td>
<td>26.5</td>
<td>26.4</td>
<td>0.80</td>
<td></td>
</tr>
<tr>
<td></td>
<td>yes</td>
<td>ObjBest</td>
<td>27.0</td>
<td>27.0</td>
<td>27.0</td>
<td>27.0</td>
<td>27.0</td>
<td>0.00</td>
<td>$p = 0.000$</td>
</tr>
<tr>
<td>cplex090</td>
<td>no</td>
<td>ObjBest</td>
<td>22.0</td>
<td>20.0</td>
<td>27.0</td>
<td>24.5</td>
<td>24.3</td>
<td>1.51</td>
<td></td>
</tr>
<tr>
<td></td>
<td>yes</td>
<td>ObjBest</td>
<td>25.0</td>
<td>24.0</td>
<td>26.0</td>
<td>24.0</td>
<td>24.3</td>
<td>0.58</td>
<td>$p = 0.752$</td>
</tr>
</tbody>
</table>

Since $p > 0.05$ only for cplex090, Fiedler-preconditioning makes a difference for all solvers except cplex090!