Of Combinatorial Optimization, n-dimensional Dice, and Reproducible Research: An Overview

Franc Brglez

NC State University
Computer Science
Raleigh, NC, USA

Of Combinatorial Optimization ...

**goal:** find an optimal configuration from a finite set of configurations.

**B-class:** 2, 4, 8, 16, 32, 64, 128, ...

**T-class:** 3, 9, 27, 81, 243, 729, 2187, ...

**P-class:** 2, 6, 24, 120, 720, 5040, ...

**H2-class:** 2, 8, 48, 384, 3840, 46080, 645120, ...

Problem instances exist in many contexts:
automated reasoning, machine vision, databases, robotics, scheduling, IC design, computer architecture design, computer networking, OR, CAD, CAM, ....
n-Dimensional Dice

Example: encoding face labels in a regular octahedron

traditional: 0 1 2 3 4 5 6 7

Hyperhedron encodings: faceCoord; faceValue

B-class: 000;3 001;4 010;5 011;6 100;7 101;0 110;1 111;2
(coordinates are based on binary tuples)

H2-class: +1,+2;5 +1,-2;6 +2,+1;7 +2,-1;0 -1,+2;1 -1,-2;2 -2,+1;3 -2,-1;4
(coordinates are based on permutations with 2 orientations)

<table>
<thead>
<tr>
<th>class</th>
<th>faces \ n</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>B(f, n)</td>
<td>2^n</td>
<td>2</td>
<td>4</td>
<td>8</td>
<td>16</td>
<td>32</td>
</tr>
<tr>
<td>P(f, n)</td>
<td>n!</td>
<td>--</td>
<td>2</td>
<td>6</td>
<td>24</td>
<td>120</td>
</tr>
<tr>
<td>H2(f, n)</td>
<td>(2^n)(n!)</td>
<td>2</td>
<td>8</td>
<td>48</td>
<td>384</td>
<td>3840</td>
</tr>
</tbody>
</table>
### n-Dimensional Dice & Statistics

<table>
<thead>
<tr>
<th>class</th>
<th>faces \ n</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>B(f, n)</td>
<td>$2^n$</td>
<td>2</td>
<td>4</td>
<td>8</td>
<td>16</td>
</tr>
<tr>
<td>P(f, n)</td>
<td>n!</td>
<td>--</td>
<td>2</td>
<td>6</td>
<td>24</td>
</tr>
<tr>
<td>H2(f, n)</td>
<td>$(2^n)(n!)$</td>
<td>2</td>
<td>8</td>
<td>48</td>
<td>384</td>
</tr>
</tbody>
</table>

Given $t = 1, 2, 3, \ldots$ as the number of throws, the probability distribution of “first success” is a geometric distribution with $p = 1/|\text{faces}|$ and pmf = $p(1 - p)^{(t-1)}$

- mean = $1/p$, for $p \ll 1$, stdev = mean
- cumulative distribution function \( \text{cdf} = 1 - (1-p)^t \)

For $p=1/8$ and $t = 8$, cdf = 0.6564, for $t = 32$, cdf = 0.9861
n-Dimensional Dice & Practice

Einstein:  
*God does not play dice*

but ....
*combinatorial solvers do!*

solver: cplex090, B&B

instance: ILP, n = 350, class-B

runTime to reach optimum:
min = 3.56 CPU seconds
max = 2110 CPU seconds

on 13 of 46 instance isomorphs, this state-of-the art solver times out at 2112 CPU seconds

Reproducible Research??

A question from the sponsor ...

Can one replicate these experiments?

"Of course you can't replicate my experiments. That's the beauty of them."

paraphrased from American Scientist, July 2002
The Problem

There are five variables in the data set, the response \verb|y| and four predictors, \verb|x1|, \verb|x2|, \verb|x3|, and \verb|x4|.

A frequentist analysis for the problem is done by the following R statements

```r
library(mcmc)
data(logit)
out <- glm(y ~ x1 + x2 + x3 + x4, data = logit,
          family = binomial())
summary(out)
```

But this problem isn't about that frequentist analysis, we want a Bayesian analysis.

Reproducible Research: here via xBed

(xBed is a test environment based on the dice metaphor)

% dice_analysis_exh diceType B
diceFunction index
instanceFile ".\xData\index\set1\v-005-B.index" coordBottom
11110 seedBottom 51307 plotType
walk solverMethod xRW2

....... cmdLine =
R CMD BATCH --vanilla
R_encap_tmp.R R_encap_tmp.Rout
.. making a copy of Rplots.pdf
as fg_v-005-B-
index_11110_51307_walk_xRW2.pdf
.. created file Figures/fg_v-005-B-
index_11110_51307_walk_xRW2
appending it to file
xTex_index/Figures.tex
%
Acknowledgements

• for help with Tcl
  -- Hemang Lavana, Clif Flynt

• for beginner’s help with R
  -- Brian Ripley, Tony Plate,
    Duncan Murdoch, David Hiebeler

• for giving us all Tex and LaTex
  -- Donald Knuth, Leslie Lamport

• for invitation of sharing the early ideas in this talk
  -- Baldomir Zajc, Andrej Žemva
Outline

• Problem instance selection
  -- index family of test problems
  -- npp (number partitioning problem) family

• Hasse graphs

• Random walks in hyperhedrons

• A stochastic search strategy: an example

• Experimental results

• Summary and conclusions
Index function family of test problems

All instances have a face with unique minimum value

Examples from instance family index:
(values piecewise linear wrt coordinates in lex order)

B: 000;3 001;4 010;5 011;6 100;7 101;0 110;1 111;2
(coordinates are based on binary tuples)

H2: +1,+2;5 +1,-2;6 +2,+1;7 +2,-1;0 -1,+2;1 -1,-2;2 -2,+1;3 -2,-1;4
(coordinates are based on permutations with 2 orientations)

Examples from instance family index-perm:
(values randomly permuted wrt coordinates in lex order)

B: 000;4 001;3 010;7 011;6 100;0 101;5 110;2 111;1
(coordinates are based on binary tuples)

H2: +1,+2;6 +1,-2;5 +2,+1;1 +2,-1;2 -1,+2;7 -1,-2;4 -2,+1;0 -2,-1;3
(coordinates are based on permutations with 2 orientations)
**npp (number partitioning prob.) family**

All instances have *a face with unique minimum value*

Additional properties include:

- number of bits in each integer = number of integers (a prerequisite for “hardest instance”)
- under *npp0*, each instance has one and only one perfect partition with discrepancy = 0
- when solved by **LDM** (due to Karmarkar-Karp), each instance reports discrepancy > 0, i.e. **LDM** (**Largest Differencing Method**) heuristic fails to find a perfect partition

Example for *n = 6*:

- iList: 4, 5, 7, 8, 35, 41
- perfect partition: 001110 ... (4 + 5 + 41) - (7 + 8 + 35) = 0
- **LDM** partition: 000101 ... (4 + 5 + 7 + 35) - (8 + 41) = 2
Hasse Graph: a generalization of Hasse diagram

- connected, sparse, polar graph
- an edge for face adjacencies
- a labeled vertex for each face
  \( \text{faceCoord}; \text{faceValue} \)
- \text{faceCoord} a binary, ternary, ..., or oriented permutation ...
- \text{faceValue} a function of interest
- function symmetry ...
- Hasse graph width, height ...
- Hasse rank distance measured from \text{bottomFace} ...
- coordinate complement based on max. Hasse rank distance ...

plot_hasse_graph of 'fg_v−003−B−index_011'
(diceType=B, diceFunction=index, bottomFace=011;6)

vertices and labels are ordered L \rightarrow R by function values
(for diceType=B, vertex distribution at each rank is binomial)
Hasse Graph: with 32 and 24 faces

32-face instance of npp
(number partitioning problem)

plot_hasse_graph of 'fg_v−006−06−npp0_00101'
(diceType=B, diceFunction=npp0, bottomFace=00101;2)

Hasse rank distance from the bottom face of the dice
vertices and labels are ordered L -> R by function values
(for diceType=B, vertex distribution at each rank is binomial)

24-face instance of jss
(job-shop scheduling prob.)

plot_hasse_graph of 'fg_v−04−05−wismer−jss_3,4,1,2'
(diceType=P, diceFunction=jss, bottomFace=3,4,1,2;51)

Hasse rank distance from the bottom face of the dice
vertices and labels are ordered L -> R by function values
(for diceType=P, vertex distribution at each rank is Mahonian)
Hasse Graph: 48 faces and H. walk

48-face instance of $H2(index, 3)$: function = index, n = 3

plot_hasse_graph of 'fg_v-003-H2-index_+1,+2,+3'
(diceType=H2, diceFunction=index, bottomFace=+1,+2,+3;28)

vertices and labels are ordered L -> R by function values
(The vertex distribution at each Hasse rank distance is a product of binomial and Mahonian distributions)
Random walks in hyperhedrons

32-face instance of $B(\text{index}, 5)$: function = index, n = 5
Throws/walks terminate upon reaching face with value = 0!

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>all uniq</td>
<td>throws length</td>
<td>throws length</td>
</tr>
<tr>
<td>0:1776</td>
<td>11 10</td>
<td>1 79</td>
<td>1 19</td>
</tr>
<tr>
<td>1:61110</td>
<td>74 29</td>
<td>1 46</td>
<td>1 4</td>
</tr>
<tr>
<td>127:68095</td>
<td>49 27</td>
<td>1 72</td>
<td>3 25</td>
</tr>
<tr>
<td>128:6568</td>
<td>11 8</td>
<td>1 71</td>
<td>1 3</td>
</tr>
<tr>
<td>median</td>
<td>21.5 17.0</td>
<td>1 22.5</td>
<td>1 14.5</td>
</tr>
<tr>
<td>mean</td>
<td>32.9 16.8</td>
<td>1 33.1</td>
<td>1.3 14.1</td>
</tr>
<tr>
<td>stdDev</td>
<td>31.7 9.3</td>
<td>0 31.2</td>
<td>1 9.0</td>
</tr>
<tr>
<td>stderr</td>
<td>2.8 0.8</td>
<td>0 2.8</td>
<td>0.1 0.8</td>
</tr>
</tbody>
</table>

32-face instance of $B(\text{index}, 5)$: function = index, n = 5
Throws/walks terminate upon reaching face with value = 0!
Random walks $xRW1$ and $xRW2$

32-face instance of $B(index, 5)$: function = index, $n = 5$

$xRW1$: with repetition from one throw

$xRW1$ walk in 'fg_v−005−B−index_11110', seedBottom=1215
(diceType=B, diceFunction=index, bottomFace=11110;11)

$xRW2$: without repetition from two throws

$xRW2$ walk in 'fg_v−005−B−index_11110', seedBottom=1248
(diceType=B, diceFunction=index, bottomFace=11110;11)
Random walks: asymptotic performance

Instances of $B(index, n)$: function $= index$, faces $= 2^n$

128 instances derived from random seed $= 1066$:

**random throws:**
- $xRT \approx 2^n$
- boost factor $\approx 1$

**walks with repetitions:**
- $xRW1 \approx 2^n$
- boost factor $\approx 1$

**walks w/o repetitions:**
- $xRW2 \approx 0.5 \cdot 2^n$
- boost factor $\approx 2$

Reduction of function evaluations makes boost factor $> 1$
A stochastic search strategy: xKL

A spreadsheet view of xKL

<table>
<thead>
<tr>
<th>probes</th>
<th>probes</th>
<th>pivotPair</th>
<th>neighbor 1</th>
<th>neighbor 2</th>
<th>neighbor 3</th>
<th>neighbor 4</th>
<th>neighbor 5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cum</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>11110;11</td>
<td>10110;3</td>
<td>11010;7</td>
<td>11100;9</td>
<td>11111;12</td>
<td>01110;27</td>
</tr>
<tr>
<td>4</td>
<td>9</td>
<td>10110;3</td>
<td>10100;1</td>
<td>10111;4</td>
<td>00110;19</td>
<td>10010;31</td>
<td>NA</td>
</tr>
<tr>
<td>3</td>
<td>12</td>
<td>10100;1</td>
<td>10101;2</td>
<td>00100;17</td>
<td>10000;29</td>
<td>NA</td>
<td>NA</td>
</tr>
<tr>
<td>2</td>
<td>14</td>
<td>10101;2</td>
<td>00101;18</td>
<td>10001;30</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
</tr>
<tr>
<td>1</td>
<td>15</td>
<td>00101;18</td>
<td>00001;14</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
</tr>
<tr>
<td>0</td>
<td>15</td>
<td>00001;14</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
</tr>
</tbody>
</table>

initial walk

A * attached to the value of the function denotes that the coordinate associated with this value has been visited already, for example 10101;2 is marked as 10101;2* -- however, only vertices in neighbor columns are so marked, not vertices found in the pivotPair column!!

first chained walk

<p>| | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>probes</td>
<td>probes</td>
<td>pivotPair</td>
<td>neighbor 1</td>
<td>neighbor 2</td>
<td>neighbor 3</td>
<td>neighbor 4</td>
<td>neighbor 5</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>10100;1</td>
<td>10101;2*</td>
<td>10110;3*</td>
<td>11100;9*</td>
<td>00100;17*</td>
<td>10000;29*</td>
<td>10000;29*</td>
</tr>
<tr>
<td>1</td>
<td>10101;2</td>
<td>10111;4*</td>
<td>11101;10</td>
<td>00101;18*</td>
<td>10001;30*</td>
<td>NA</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>10111;4</td>
<td>10011;0</td>
<td>11111;12*</td>
<td>00111;20</td>
<td>NA</td>
<td>NA</td>
</tr>
<tr>
<td>2</td>
<td>5</td>
<td>10011;0</td>
<td>11011;8</td>
<td>00011;16</td>
<td>NA</td>
<td>NA</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>6</td>
<td>11011;8</td>
<td>01011;24</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
</tr>
<tr>
<td>0</td>
<td>6</td>
<td>01011;24</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
</tr>
</tbody>
</table>

termination of the walk

<p>| | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>probes</td>
<td>probes</td>
<td>pivotPair</td>
<td>neighbor 1</td>
<td>neighbor 2</td>
<td>neighbor 3</td>
<td>neighbor 4</td>
<td>neighbor 5</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>10011;0</td>
<td>10111;4*</td>
<td>11011;8*</td>
<td>00011;16*</td>
<td>10001;30*</td>
<td>10010;31*</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>10111;4</td>
<td>10101;2*</td>
<td>10110;3*</td>
<td>11111;12*</td>
<td>00111;20*</td>
<td>NA</td>
<td></td>
</tr>
</tbody>
</table>

A Hasse graph view of xKL

initial walk -- blue lines
chained walk -- dotted lines

xKL walk in 'fg_v−005–B–index_11110'
(diceType=B, diceFunction=index, bottomFace=11110;11)

A Hasse graph view of xKL

Hasse rank distance from the bottom face of the dice
vertices and labels are ordered L → R by function values
(for diceType=B, vertex distribution at each rank is binomial)
Experimental results: index & index-perm

Values piecewise linear wrt to coordinates in lex order

Values randomly permuted wrt to coordinates in lex order
Experimental results: npp0 & index-perm

Values from npp0 wrt to coordinates in lex order

plot_scatter of 'fg_v−006–06–npp0_lex_scatter'
(diceType = B, diceFunction = npp0 (number partitioning prob.))

Values from index-perm wrt to coordinates in lex order

plot_scatter of 'fg_v−005–B–perm–index_lex_scatter'
(diceType = B, diceFunction = index–perm (permuted index))
Experimental asymptotics with xKL solver

Instance families npp0, index-perm, index

for each family, 128 instances were tested from random seed = 1492.

**npp0:**

cntProbes \(\approx 2^n\)

**index-perm:**

cntProbes \(\approx 2^n\)

**index:**

cntProbes \(\approx n^{2.251}\)

Order of function values does matter!
Summary

• First attempt to argue for merits of hyperhedron as
  -- a model for instances arising as combinatorial problems
  -- a metaphor for classes of combinatorial search algorithms
    that remains invariant under choice of instance
    coordinates, from an n-tuple to an oriented permutation
    of length n.

• Illustration of ideas conveys a depth of understanding
  that symbolic description of them cannot readily match
  -- hence the dice model and analogies to dice solids
  -- hence the Hasse graph model of dice with well-defined
    height and width, whose labeled vertices can be readily
    evaluated and traversed

• More musings on these topics are in progress ...
Conclusions

There may be no better way to conclude this talk than to paraphrase a famous quote from the founding editor of the Manchester Guardian (now The Guardian)**:

Comments are free but the reproducibility of experiments is sacred.

You are invited to contribute both.
Thank you for listening.