Load balancing while solving large linear integer problems for enumeration purposes

José Núñez Ares
Jeff Linderoth
introduction

disciplines:

experimental design (statistics)
integer programming (mathematical programming)
high-throughput computing

goal:

complete enumeration of MARS designs
what is a MARS design?

top of one of the buttes in Murray Buttes. Image processing by Paul Hammond. Photo Credit: NASA/JPL-Caltech/MSSS/Paul Hammond
what is a MARS design?

\[
\begin{pmatrix}
1 & 0 & 0 & -1 & 1 & -1 & 0 \\
0 & 1 & 0 & 0 & -1 & -1 & -1 \\
1 & 0 & 1 & 0 & 0 & 1 & -1 \\
1 & 1 & 0 & 1 & 0 & 0 & 1 \\
-1 & 1 & 1 & 0 & 1 & 0 & 0 \\
0 & 1 & -1 & -1 & 0 & 1 & 0 \\
0 & 0 & 1 & -1 & -1 & 0 & 1 \\
-1 & 0 & 0 & 1 & -1 & 1 & 0 \\
0 & -1 & 0 & 0 & 1 & 1 & 1 \\
-1 & 0 & -1 & 0 & 0 & -1 & 1 \\
-1 & -1 & 0 & -1 & 0 & 0 & -1 \\
1 & -1 & -1 & 0 & -1 & 0 & 0 \\
0 & -1 & 1 & 1 & 0 & -1 & 0 \\
0 & 0 & -1 & 1 & 1 & 0 & -1 \\
\end{pmatrix}
\]

minimally aliased response surface designs
what is a MARS design?

A $m \times n \{−1, 0, 1\}$ matrix has the following desirable statistical properties:

$i)$ columns sum up to zero

$ii)$ columns are orthogonal

$iii)$ component-wise multiplication of any 3 columns produce a column that sums up to zero
why is this important?

there is a small set of MARS designs and they have became standard in response surface methodology designs with less runs which give the same amount of information of bigger ones designs which perform well under conflicting criteria
how do we find them?

\[ \sum_{p \in \Omega} y^p = n \]

\[ \sum_{p \in \Omega_{0i}} y^p = n_0^{ME} \quad 1 \leq i \leq m \]

\[ \sum_{p \in \Omega_{0ij}} y^p = n_0^{IE} \quad 1 \leq i < j \leq m \]

\[ \sum_{p \in \Omega} \alpha_{ij}^p y^p = 0 \quad 1 \leq i < j \leq m \]

\[ \sum_{p \in \Omega} \alpha_{ijk}^p y^p = 0 \quad 1 \leq i \leq j \leq k \leq m \]

\[ y^p \in \{0, 1\} \quad p \in \Omega \]

|\Omega| = 3^m - 1

\[ S \subset \Omega := \text{basic design} \]

\[ G := \text{group of permutations of levels and factors} \]

\[ |G| = 2^m m! \]

iteratively add isomorphism inequalities:

\[ \sum_{p \in g(s)} y^p \leq n - 1, \forall g \in G \]

discrete enumeration tree exploration
what are the problems?

MARS designs have huge isomorphic groups

mathematical programming techniques help with this

Andy Warhol's Marilyn Monroe Series, 1967
what are the problems?

$$\sum_{p \in \Omega} y^p = n$$

$$\sum_{p \in \Omega_{0i}} y^p = n_0^{ME} \quad 1 \leq i \leq m$$

$$\sum_{p \in \Omega_{0ij}} y^p = n_0^{IE} \quad 1 \leq i < j \leq m$$

$$\sum_{p \in \Omega} \alpha_{ij}^p y^p = 0 \quad 1 \leq i < j \leq m$$

$$\sum_{p \in \Omega} \alpha_{ijk}^p y^p = 0 \quad 1 \leq i \leq j \leq k \leq m$$

$$y^p \in \{0, 1\} \quad p \in \Omega$$

tree of exponential size

$$|\Omega| = 3^m - 1$$

$$S \subset \Omega := \text{basic design}$$

$$G := \text{group of permutations of levels and factors}$$

$$|G| = 2^m m!$$

iteratively add isomorphism inequalities:

$$\sum_{p \in g(s)} y^p \leq n - 1, \forall g \in G$$
what does the enumeration tree look like?

![Diagram of a B&B tree (4_22_6_10_minInd 0s)]

skinny deep
what does the enumeration tree look like?

B&B tree (tree_5_24_6_10_minInd 43s )

depth

skiny

deep
what does the enumeration tree look like?

B&B tree (5_26_8_14_minInd 1159s )

skinny
depth
what does the enumeration tree look like?
why htcondor?

unknown number of processed nodes (potentially huge)

long processing time

“pleasantly parallel”, little communication and synchronization needed
our load balancing scheme

element 1: Knuth estimation
done ntimes, if predicted size > threshold
then we do BFS, otherwise DFS

element 2: breadth-first-search (BFS)
until a certain depth determined dynamically by a max processing time
OR a max number of open nodes
our load balancing scheme

**element 3: depth-first-search (DFS)**

faster and more memory efficient than BFS, creates less open nodes while evaluating more nodes, max processing time

**element 4: trimming**

after BFS and DFS (if not solved) we solve every open node if the solution time < max processing time of a trivial node, otherwise we store the open node data
our load balancing scheme

Knuth dives from root node

this is diving 1...

with predicted size < threshold
our load balancing scheme

Knuth dives from root node
this is diving 2...
with predicted size < threshold
our load balancing scheme

Knuth dives from root node
this is diving 3...
with predicted size \(>\) threshold
our load balancing scheme

we then do BFS from root node

dynamical depth, which depends on time/number of open nodes
our load balancing scheme

now we repeat the process for each one of the open nodes ...

let's do some Knuth dives ...
our load balancing scheme

do DFS from this node

called this time

predicted size < threshold
our load balancing scheme

this time

predicted size < threshold

do DFS from this node
our load balancing scheme

we repeat the process on the open nodes
HTCondor DAGMan files

marsd.dag
SUBDAG EXTERNAL workers workers.dag
SCRIPT POST workers marsdOneIter.sh 7
RETRY workers 1000

workers.dag
JOB main submit-solve.cmd

input data: 30-36MB

output data < 1MB
executables

marsdOneIter.sh

identifies the open nodes
writes the htcondor submit file
dynamically tune the parameters

marsd

does the load balance
<table>
<thead>
<tr>
<th>size</th>
<th>6</th>
<th>7</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>#nodes</td>
<td>(2,276 \times 10^6)</td>
<td>(704 \times 10^6)</td>
<td>(166 \times 10^6)</td>
</tr>
<tr>
<td>CPU time (years)</td>
<td>3.32</td>
<td>5.57</td>
<td>20.52</td>
</tr>
<tr>
<td>#solutions</td>
<td>296,193</td>
<td>20,184</td>
<td>521</td>
</tr>
</tbody>
</table>
acknowledgement

Peter Goos, my promotor at KU Leuven

fonds wetenschappelijk onderzoek, grant V402917N
thank you!

any questions?