Robustness of Combinatorial Algorithms vs Component Reliability: Two Faces of the Same Coin?

Franc Brglez   Matthias F. Stallmann

Dept. of Computer Science, NC State University
Raleigh, North Carolina, USA
{brglez, stallmann}@cs.ncsu.edu

A presentation to NASA Langley Core Competency Directors
during the visit to NCSU on September 4, 2003

Summary

Robustness of Combinatorial Algorithms vs Component Reliability: Two Faces of the Same Coin?

Franc Brglez and Matt Stallmann
Computer Science Department, NCSU

The recent euphoria of finding, at a great computational expense, the branch-and-bound solution to a Quadratic Assignment Problem posed in the 60's (http://www-unix.mcs.anl.gov/metaneos/nug30/) may well be eclipsed by an alternative search algorithm that finds solutions of the same quality for the same problem faster (on the average) than one can finish drinking a cup of coffee (a manuscript under review). In quite a different context, a critical system component performing without fail for years, may unexpectedly fail due to an unanticipated fatigue phenomenon. While the two contexts for these phenomena are clearly different, there are layers of underlying mathematical framework that are remarkably similar for both phenomena.

In this talk we review our recently demonstrated methodology of reliable performance testing of combinatorial solvers, leading to introduction of new, significantly improved solvers. Next, we argue for the generalization of the testbed devised for experimental flows with combinatorial solvers to formalizing a hierarchy of computational flows that support cyclic plasticity modeling (CPM), finite element analysis with embedded CPM (FEA), material fatigue modeling (MFM), and structural fatigue modeling (SFM).

Most importantly, each flow is terminated by a model-specific verification and report generation program. We also outline a current software prototype (OmniFlow) that allows user to describe any hierarchy of (distributed) programs and (distributed) data and renders a graphical user face (GUI) that allows the user to interact with controlling the flow of execution and access to all data. A real-time demo of OmniFlow may be scheduled, following this presentation.

Issues we would like to discuss with NASA visitors and participants include:

--- support for a program on systematic prototyping of state-of-the-art combinatorial problem solvers of particular interest to NASA projects such as optimal placement of sensors, subject to noise constraints. Each of these prototypes would be subject to the highly reliable testbed deployment and comparative performance analysis with current state-of-the-art combinatorial solvers.

--- support for a program on generalizing the current OmniFlow prototype to a universal client with a GUI that will embed not only distributed programs and data but also provide collaborative peer-to-peer control of computational tasks in the flow. The driver to prototype such generalization may well be the embedding of CPM, FEA, MFM, and SFM software programs and data supplied by the distributed and collaborating participants in a specific NASA project itself.
Outline

- Context and Motivation
- Example: Reliability vs Solvability
- Current problems: SAT, maxSAT, LABS
- OmniFlow (GUI) client for distrib. computing
- OmniFlow real-time demo
- peer2peer contexts
  - distributed computing fatigue research
  - motivation for p2p computing environment
- Items for discussion

Context and Motivation

*Side 1 of the coin:* Reliability of a critical component

<table>
<thead>
<tr>
<th>Environment 1</th>
<th>Environment 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>30 mins</td>
<td>maximize lifetime</td>
</tr>
</tbody>
</table>

*Side 2 of the coin:* Solvability of a combinatorial problem (nug30)

<table>
<thead>
<tr>
<th>Local Search</th>
<th>Branch-and-bound</th>
</tr>
</thead>
<tbody>
<tr>
<td>1--60 seconds (1 processor)</td>
<td>minimize runtime</td>
</tr>
<tr>
<td>RELIABLY!!</td>
<td>(2510 processors)</td>
</tr>
</tbody>
</table>
Example: Reliability vs Solvability

Component (A/C unit of Boeing 720)

How long will it last?

(213 total)

MTTF = 93.3 hours
Std = 107 hours

(Prochan, 1963)

Component replicas

problem replicas

Hot. 15
deg

Hot. 30
deg

LA-x*y problem

How long to optimally place x warehouses to minimize travel to y locations?

(24 total)

This LA-solver should be improved (reliably!)

Current problems: SAT, maxSAT, LABS

SATbed: www.cbl.ncsu.edu/OpenExperiments/SAT/

Tasks:

refGen \rightarrow classGen \rightarrow solver1 \rightarrow statsGen \rightarrow rptGen

Data:

12 SAT solvers, contributed by 7 peers, have been encapsulated …

LABSbed: www.cbl.ncsu.edu/OpenExperiments/LABS/

... similar architecture of LABS problem solvers

maxSATbed: ... in progress
OmniFlow (GUI) client for distrib. computing

**INPUT (in XML):**
(hierarchical) graph of task nodes, data nodes  
task-task (control) edges, data-task (data) edges

**OUTPUT:** a task tree and a taskflow

Tasks: T1 → T2 → T4 → T3

Data: T4a → T4b → T4c

**User interactions:**
- data node click invokes viewer or editor
- control edge click opens/closes task-task edge
- task node click invokes scheduler and task execution sequence(s)
- task node H-icon click expands taskflow hierarchy, etc.

A brief OmniFlow real-time demo follows (next 2 slides screenshots)
Taskflow Inst. (re-configured in GUI)

Enabled concurrent execution of tasks 
({(InitB), (InitC)}, then ((B), (C))

green" state shows task completion

disabled repeatInvEdge

disabled invocationEdge

peer2peer distributed fatigue research

(a) cyclic plasticity modeling flow

CPM: cyclic plasticity modeling

(b) finite-element analysis flow

FEA: finite element analysis engine with embedded CPM

(c) material fatigue modeling flow

MFM: material fatigue modeling

(d) structural fatigue modeling flow

SFM: structural fatigue modeling
Peers outside p2p environment

Alice, Bob, Cindy, David are programming tasks A, B, C, D

Assumptions:
• four programmers have no awareness of each other
• data formats between tasks are incompatible

Question:
• what will it take to merge four tasks into single program to evaluate alg1 and alg2?

Answer:
• time-consuming, error-prone, hard-to-test, hard-coded programming (with file translators), possibly by the programmer no. 5

Peers within p2p environment

Alice, Bob, Cindy, David are programming tasks A, B, C, D -- each using dynamically created GUI

In this taskflow, each task recognizes local and global invocation context (L/G) and waiting and ready state (W/R).

The atomic states (W/R) allow to schedule tasks (A) and (B) for concurrent execution.

Such control is essential to render this GUI client into a tool not only for peer2peer programming but also for peer2peer scientific computing.

Created by David as the project leader, GUI captures task-task and data-task dependencies
Items for discussion

• a program on reliable prototyping of novel combinatorial solvers of interest to NASA and MAE/ECE/other participants (e.g. optimal placement of sensors, subject to noise constraints) …

• a driver program in a distributed computational fatigue research that demonstrates merits of a collaborative peer2peer control within OmniFlow - e.g. the embedding of CPM, FEA, MFM, and FSM software programs and data shared by the distributed and collaborating participants in a specific NASA project …

A project site review scenario …

… paraphrased from American Scientist, July 2002

Can one replicate these experiments?

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