From VBA to High Performance Computing in Supply Chain Optimisation

Case Study

Tatiana Kalganova, Marco Veluscek

Electronic and Computer Engineering
College of Engineering, Design and Physical Sciences
Brunel University
Kingston Lane, Uxbridge, UB8 3PH, United Kingdom
Outline

- Introduction
- Programming in Heterogeneous environment
- Chosen working framework
- C++11 for Multi-platform Implementation
- OpenMP for Multi-core Parallel Programming
- Standard Data Exchange Format – XML
- Conclusion
Introduction to Problem
Supply Chain Optimisation

Dealers

Demand

Ports

Source

Ports

Production

sources

Shipping

lanes

Land links

...  ...

Land links

...  ...

Route between dealer and production source direct
Problem complexity

Addressed problem:

> 2945 nodes and 135,111 edges.
> Data from a 24 months period.
> 30 products, each:
  > Made of 15 components, built from 5 sources.
  > 39 assembly point, 200 dealer locations and 48 shipping regions, partially interconnected.
Example of problem being solved
Experimental results

Round 1 Baseline       Round 1 ACO       Round 2A ACO  (Add Energy Cost)       Round 2C ACO  (Add Lane Commitments)       Round 3A ACO  (Add Inventory Costs)       Round 3B ACO  (Add Tariffs)

+4.6% to Marginal Profit
Introduction to Problem
Supply Chain Optimisation

- Complex problem
- Uses different sources of data
- Optimisation algorithm required to obtain the solution within reasonable time for integration purposes
- The developed algorithms have to be able to be integrated into existing system
- The algorithms can be run on different computational platforms
Heterogeneous development environment

- Team perspective
- Available resources perspective
Heterogeneous environment
Team skill sets

UG students
• Have no previous experience on programming of relatively complex systems
• Have experience on programming of individual projects only
• Most often have no experience on working on programming projects in teams
• In process of developing the programming skills
• Duration of Final Year project – 6 months part time

PG students
• In most cases have no previous experience on programming of relatively complex systems
• Duration of MSc project – 3 months full time (PGT students)
• Duration of PhD – 3 years full time (PGR students)

Company team
• Experience programmers, often used advanced programming techniques
## Heterogeneous environment

### Limitation of resources

<table>
<thead>
<tr>
<th>Company</th>
<th>Company Resources</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>HPC</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>University: PGR students</th>
<th>University Resources</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Linux servers</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>University: UG and PGT Students</th>
<th>University Resources</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Windows-based PCs</td>
</tr>
<tr>
<td></td>
<td>Macs</td>
</tr>
<tr>
<td></td>
<td>Multi-core computers</td>
</tr>
</tbody>
</table>
General approach
Architecture

Data handling

Optimiser
Data handling

- Spreadsheets
- XML
- Text
- Databases
Optimiser

- Ant colony algorithm
- Genetic programming
- Linear Programming
Ant colony optimisation

[Diagram of a network with nodes and edges labeled with numbers]

Customer 1

... Customer n

HB

NT

28
Working Framework
Working Framework

Main Programming Language:

- C++ standard version 11 (approved by ISO, 12 August 2012).
- Linux G++ compiler version GCC 4.7.2.

Style of programming

- Customised use of GUI depending on the programming environment used.
- For algorithmic implementation part, only use of common libraries to facilitate the stable algorithmic performance in heterogeneous environment.
Working Framework

Parallel Libraries

- OpenMP version 3.1 for multi-core programming.

Standard Data Exchange Format

- XML for data structures definitions and
- tinyXML as parsing library.
Many official standard functionalities are supported on all major platforms.

Macros may be used for platform-dependent requirements:

- `_WIN32, _WIN64` available on MS systems.
- `__GNUC__` available on GNU GCC for Linux systems.

Library for file system interaction:

- `direct.h` on MS systems.
- `sys/types.h` and `sys/stat.h` on GNU GCC.

`GNU make` may be used for compilation on all major platforms.
C++11 for Multi-platform Implementation

GNU make example.

```bash
CC = g++
LDFLAGS = -$lm
SRC_LOC = ../src/
ALG = aco/
CORE_LOC = $(SRC_LOC)/$(ALG)/core/
DATA_LOC = $(SRC_LOC)/$(ALG)/data/
IO_LOC = $(SRC_LOC)/$(ALG)/io/
MODEL_LOC = $(SRC_LOC)/$(ALG)/model/
UTILITY = $(SRC_LOC)/$(ALG)/util/
CFLAGS = -std=gnu++0x -Os -fopenmp -fipa -fp -pg -Wall -g -ggdb

all : rebuild
     ./acs
run : aco
     ./acs
build : aco
     rebuild :
          $(MAKE) clean
          $(MAKE) aco
lib :
     $(MAKE) build
     ar crf ../lib/aco.a $(wildcard *.o)
aco: Colony.o OptimizationProblem.o DistributionPlan.o Fitness.o TransportationNetwork.o任通.o RandGen.o ticpp.o tinyxml2.o tinyxml.o tinyxmlerror.o tinyxmlparser.o
$(CC) $(CFLAGS) main.o DistributionPlan.o Fitness.o Colony.o OptimizationProblem.o Utilities.o MonteCarlo.o RandGen.o ticpp.o tinyxml2.o tinyxml.o tinyxmlerror.o
```

> High performance configuration.

> Static library compilation.
OpenMP for Multi-core Parallel Programming

> Parallel directive

```c
// Split problem

#pragma omp parallel for
for (int core=0; core<nCores; core++) {
    Fitness* fitnessFunction = _cores[fitnessFunction[core];
    RandGen rand = randGens[core];
```

> Parallel loop scheduling

```c
// Load port to port
_portToPort.init(sourcePorts->size(), dealerPorts->size(), -1);

#pragma omp parallel for schedule(static) firstprivate(randomGenerator)
for (int i=0; i<sourcePorts->size(); i++) {
```

> Data scope attribute

> Execution on Intel i7 CPU 3.40GHz of 100 iterations on 8 cores.

> Reduced runtime requirements.

> Improved search as the visited portion of the solution space is increased.
Standard Data Exchange Format - XML

> Extensible Markup Language (XML):
  > Encoding for all input and output files.

> tinyXML:
  > Simple, small, and efficient C++ static library for XML documents parsing.

> Example for document parsing:

> Example for document navigation:
Standard Data Exchange Format - XML

**XML Table Representation:**

```xml
<data>
  <header_row>
    <col_name/>
    ...
  </header_row>
  <content_row>
    <cell/>
  </content_row>
  ...
</data>
```

**XML Matrix Representation:**

```xml
<data>
  <row_data>
    <row_name/>
    <cell>
      <col_name/>
      <cell_data/>
      ...
    </cell>
    ...
  </row_data>
  ...
</data>
```
Experimental results
1000 iterations for 12 months

<table>
<thead>
<tr>
<th>Operating system</th>
<th>Programming approach</th>
<th># cores</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Windows</td>
<td>VBA</td>
<td>single / multiple core</td>
<td>N/A</td>
</tr>
<tr>
<td>HPC</td>
<td>VBA</td>
<td></td>
<td>1.5 months</td>
</tr>
<tr>
<td>Windows</td>
<td>C++</td>
<td>single core</td>
<td>500s</td>
</tr>
<tr>
<td>Linux</td>
<td>C++</td>
<td>single core</td>
<td>800s</td>
</tr>
<tr>
<td>Windows</td>
<td>C++</td>
<td>8 cores</td>
<td>60s</td>
</tr>
<tr>
<td>Linux</td>
<td>C++</td>
<td>16 cores</td>
<td>50s</td>
</tr>
<tr>
<td>HPC</td>
<td>C++</td>
<td></td>
<td>1.5s</td>
</tr>
</tbody>
</table>
Conclusion

- OpenMP for Multi-core Parallel Programming
- C++11 for Multi-platform Implementation
- Standard Data Exchange Format – XML
- The use of XML allows possibilities of exploring difference methods of visualization

References

