From VBA to High Performance Computing in Supply Chain Optimisation

Case Study

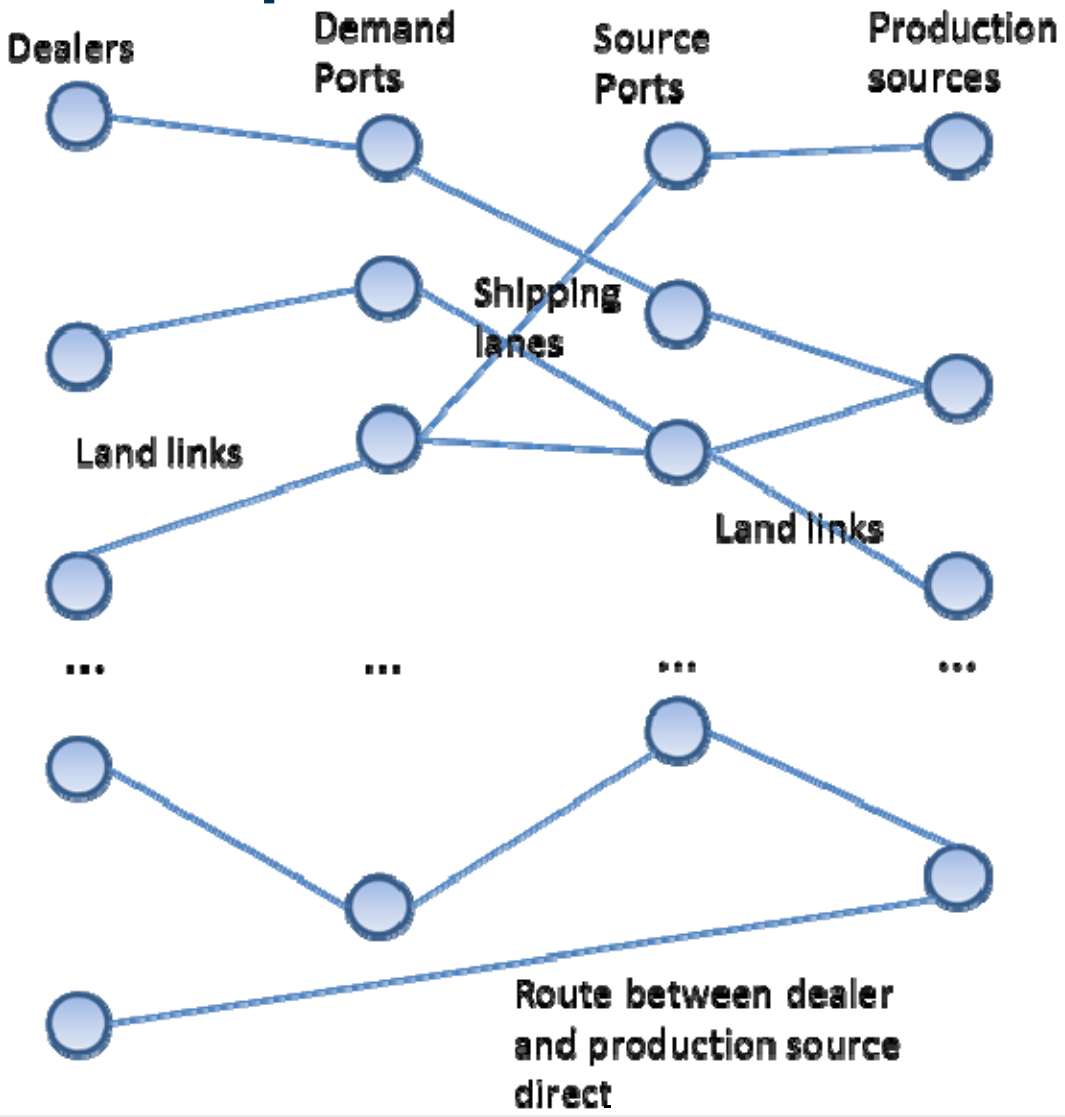
Tatiana Kalganova, Marco Veluscek

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

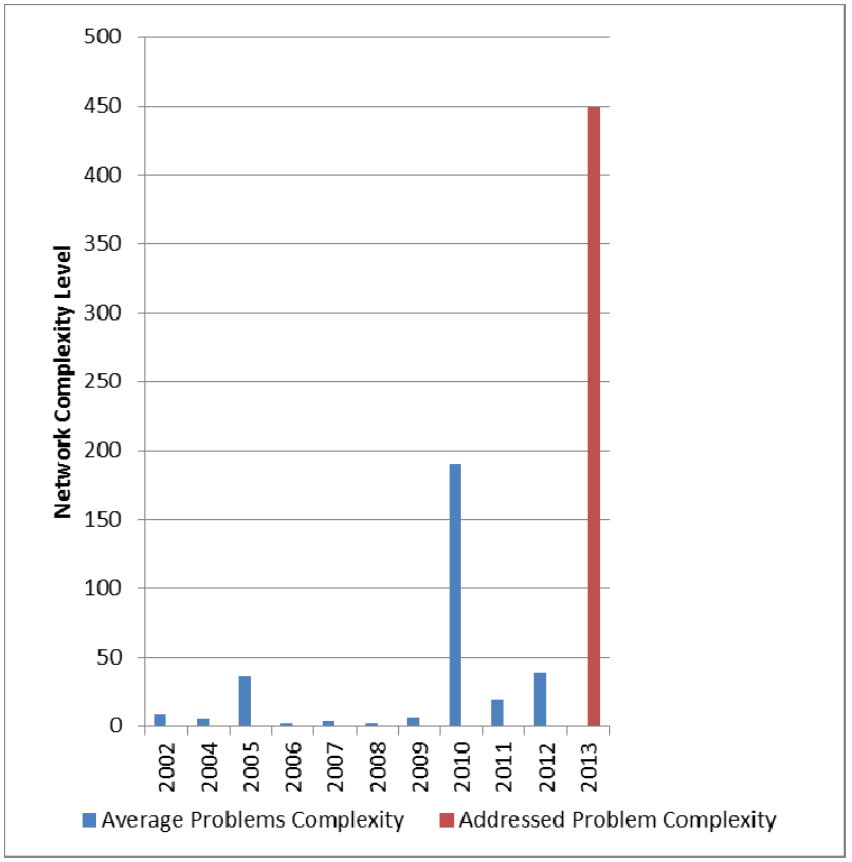
Introduction to Problem
Supply Chain Optimisation



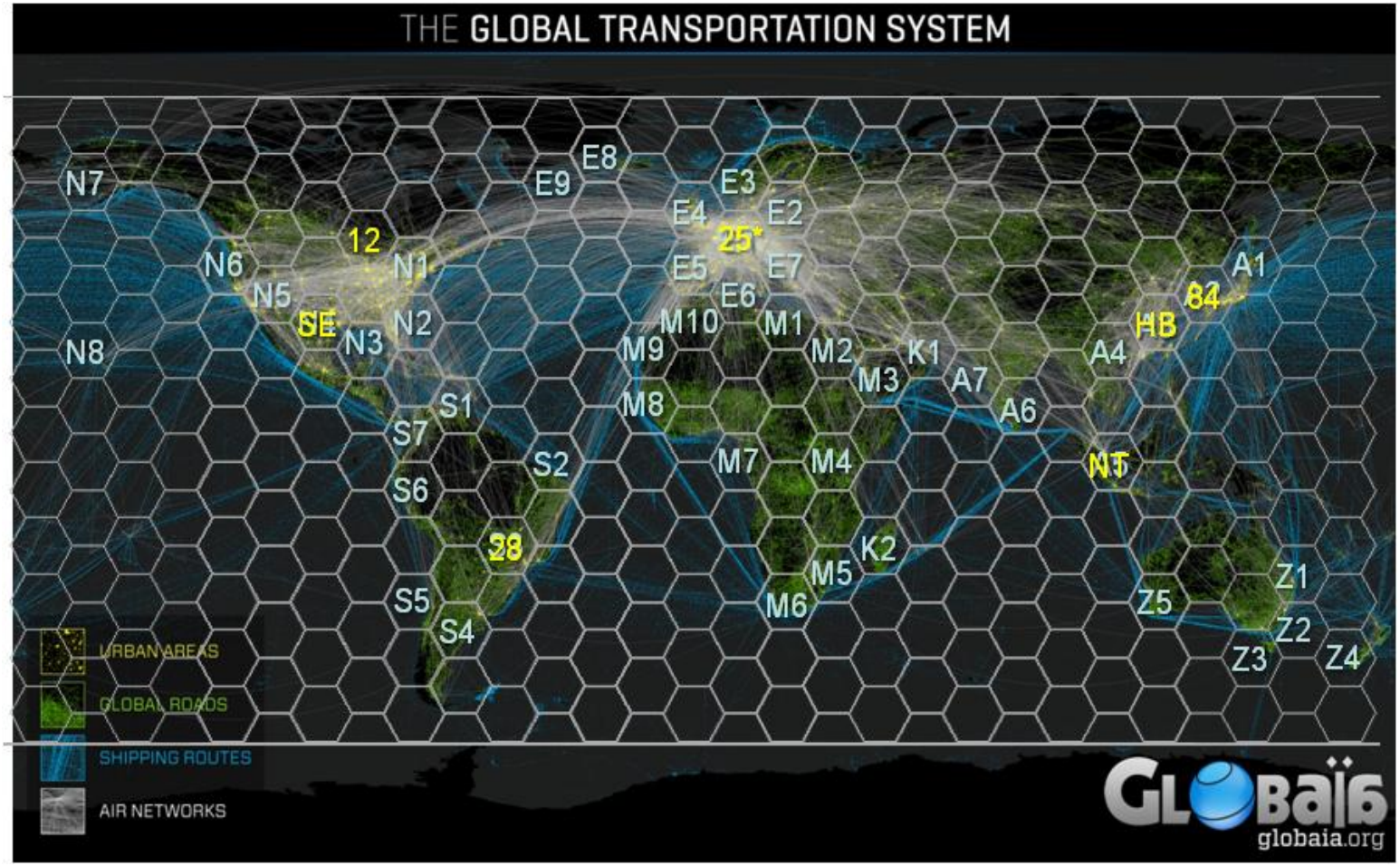
Problem complexity

Addressed problem:

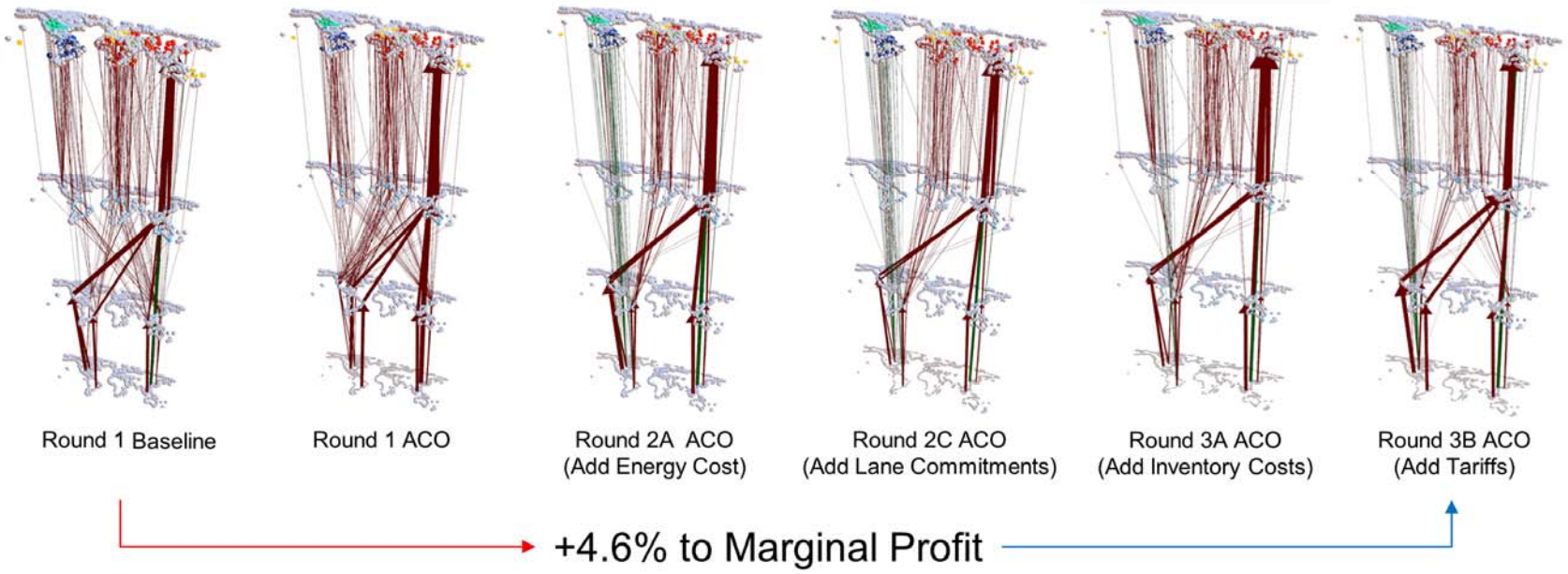
- 2945 nodes and 135,111 edges.
- Data from a 24 months period.
- 30 products, each:
  - Made of 15 components, built from 5 sources.
  - 39 assembly point, 200 dealer locations and 48 shipping regions, partially interconnected.



Example of problem being solved



Experimental results



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

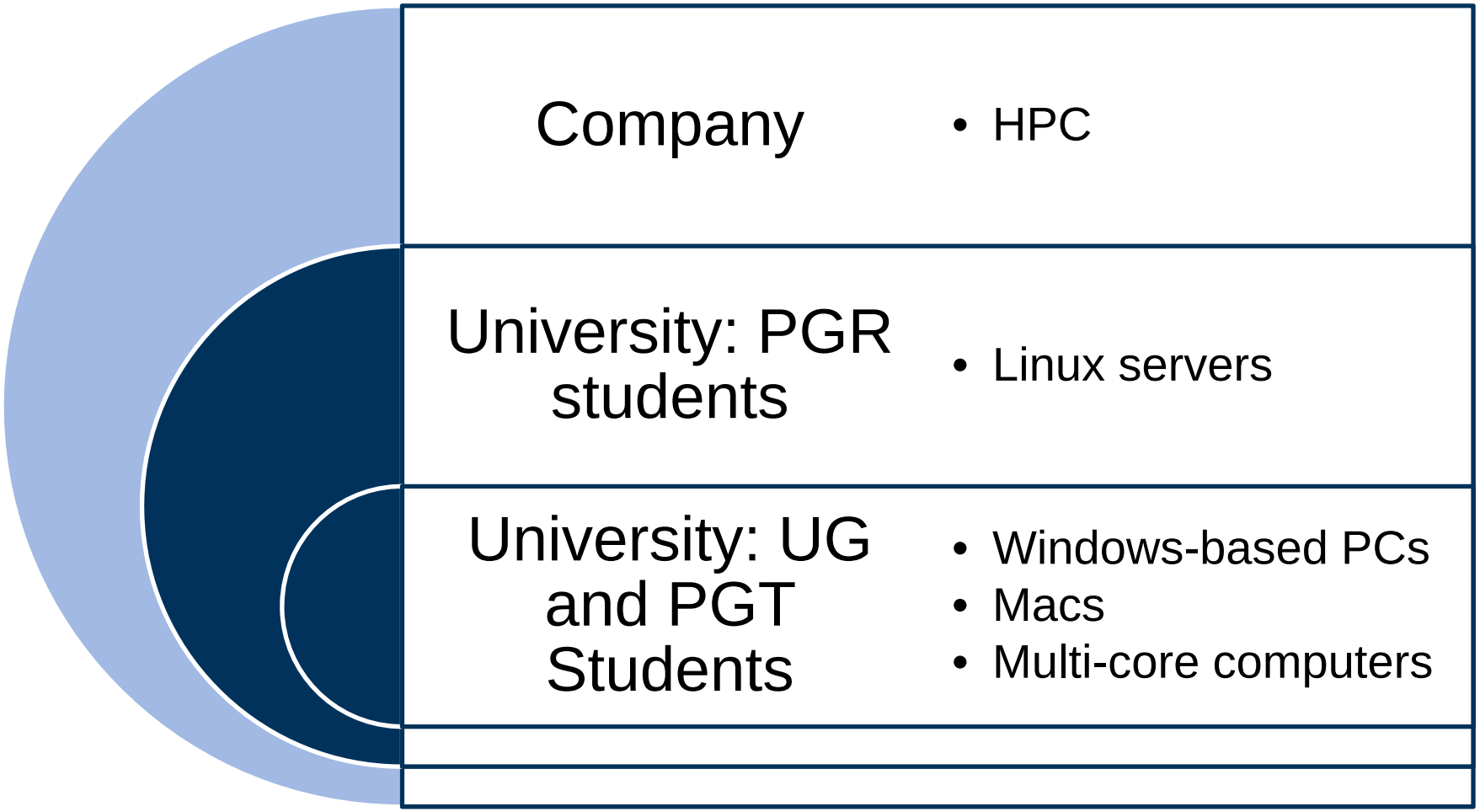
- In most cases have no previous experience on programming of relatively complex systems

- Duration of MSc project – 3 months full time (PGT students)
- Duration of PhD – 3 years full time (PGR students)

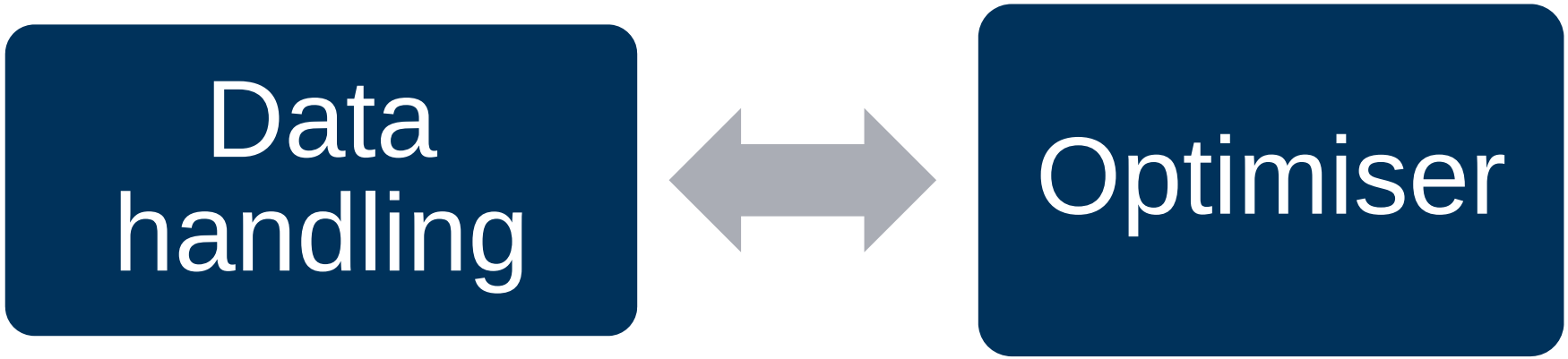
Company team

- Experience programmers, often used advanced programming techniques

Heterogeneous environment
Limitation of resources



General approach
Architecture



Data handling

Spreadsheets

XML

Text

Databases

Optimiser

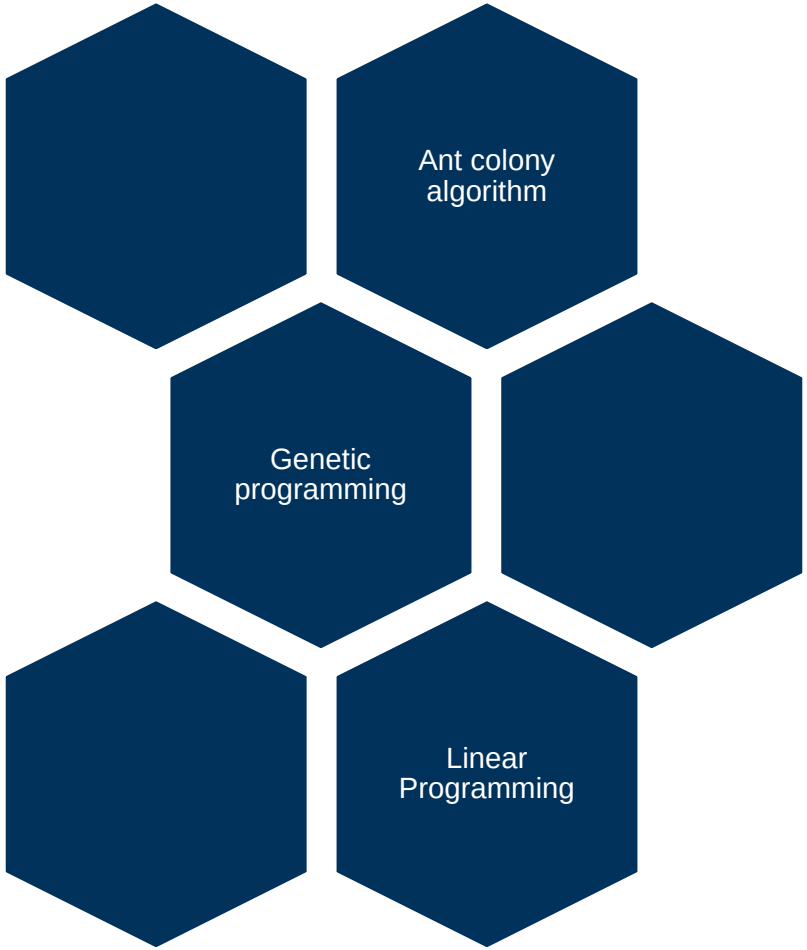
Single core computers

Multiple core computers

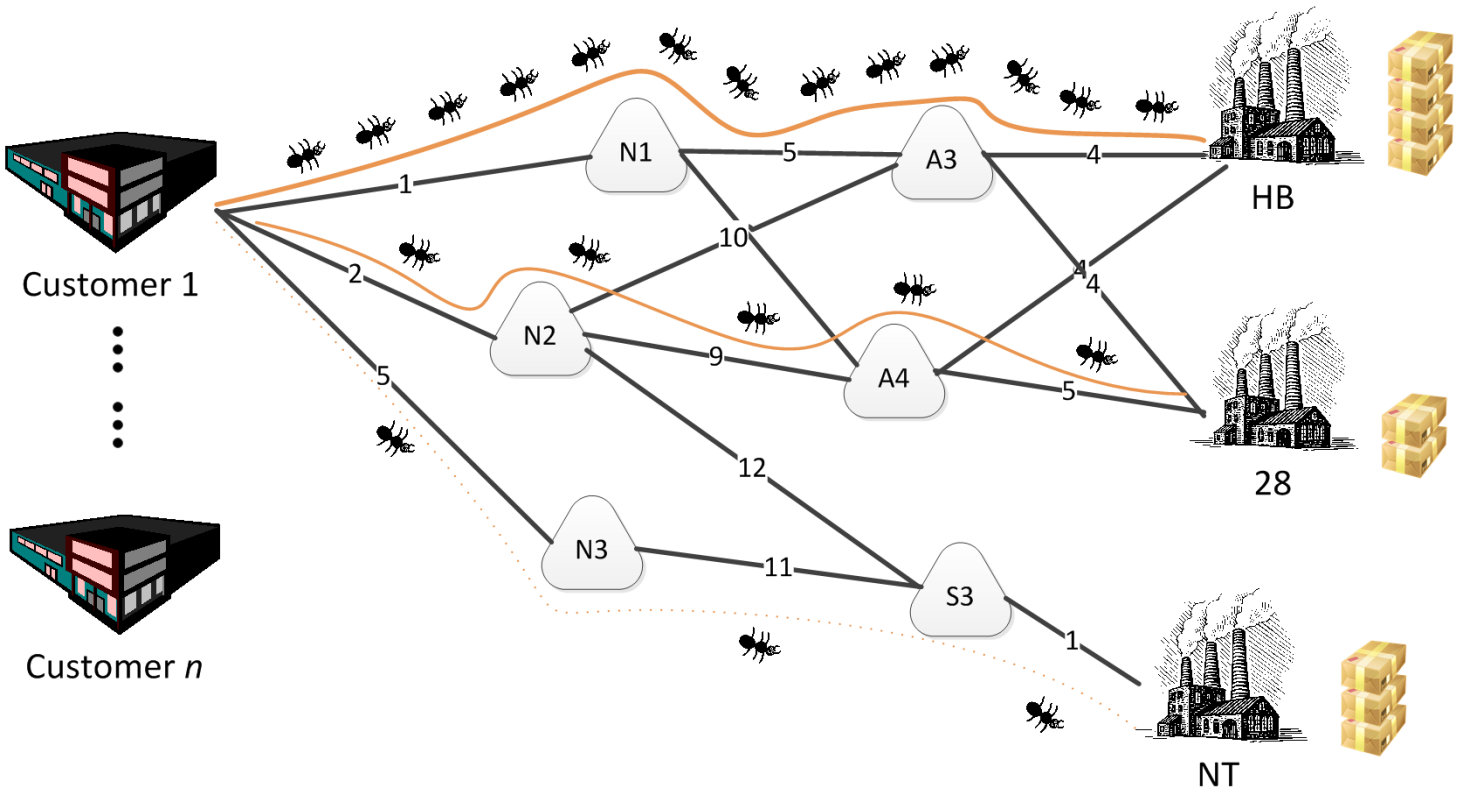
Linux servers

HPC

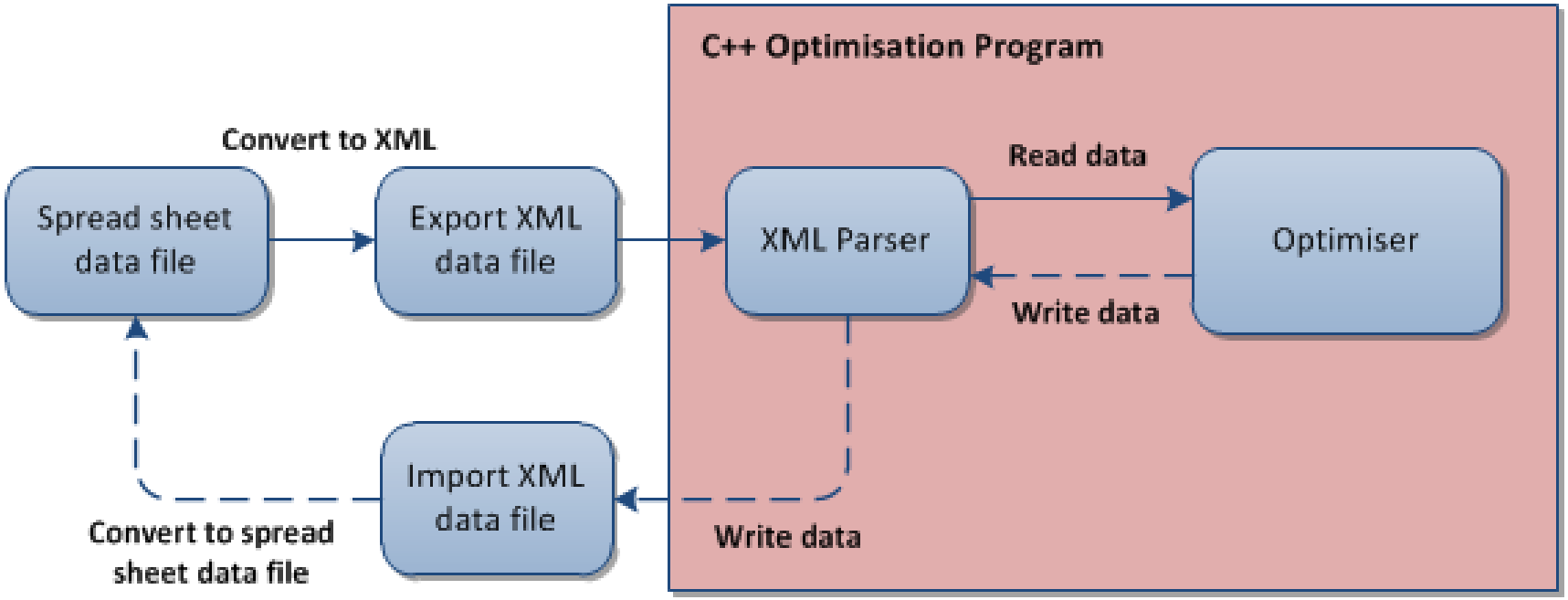
Optimiser


Ant colony algorithm
Genetic programming
Linear Programming

Ant colony optimisation



Working Framework



Working Framework

Main Programming Language:

- *C++ standard version 11* (approved by ISO, 12 August 2012).
- *Microsoft Visual C++* for Visual Studio Professional 2013.
- *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.

C++11 for Multi-platform Implementation

- 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.

- High performance configuration.
- Static library compilation.

```
1  CC = g++
2  LDFLAGS = #-lm
3  SRC_LOC = ../src/
4  ALG = aco/
5  CORE_LOC = ${SRC_LOC}${ALG}core/
6  DATA_LOC = ${SRC_LOC}${ALG}data/
7  IO_LOC = ${SRC_LOC}${ALG}io/
8  MODEL_LOC = ${SRC_LOC}${ALG}model/
9  UTIL_LOC = ${SRC_LOC}${ALG}util/
10 CFLAGS = -std=gnu++0x -O3 -fopenmp #-O3 #-pg #-Wall -g -ggdb
11
12 all : rebuild
13     ./acs
14
15 run : aco
16     ./acs
17
18 build : aco
19
20 rebuild :
21     $(MAKE) clean
22     $(MAKE) aco
23
24 lib :
25     $(MAKE) build
26     ar crf ../lib/aco.a $(wildcard *.o)
27
28 aco: Colony.o OptimizationProblem.o DistributionPlan.o Fitness.o TransportationNetw
   main.o RandGen.o ticpp.o tinyxml2.o tinyxml.o tinyxmlerror.o tinyxmlparser.o
29     ${CC} ${CFLAGS} main.o DistributionPlan.o Fitness.o Colony.o OptimizationProble
       Utilities.o MonteCarlo.o RandGen.o ticpp.o tinyxml2.o tinyxml.o tinyxmlerror.o
```

OpenMP for Multi-core Parallel Programming

- Parallel directive

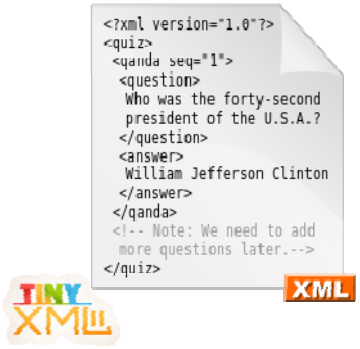
```
248
249     /// Split problem
250 #ifdef _OPENMP
251     omp_set_num_threads(nCores);
252 #endif
    #pragma omp parallel for
254     for (int core=0; core<nCores; core++) {
255
256         Fitness* fitnessFunction = _coresFitnessFunct[core];
257         RandGen rand = randGens[core];
```

- Parallel loop scheduling
- Data scope attribute

```
249     RandGen randomGenerator;
250     if (_simulateMC)
251         randomGenerator = _monteCarlo->getRandGenerator();
252
253     /// Load port to port
254     _portToPort.init(sourcePorts->size(), dealerPorts->size(), -1);
255 #pragma omp parallel for schedule(static) firstprivate(randomGenerator)
256     for (int i=0; i<sourcePorts->size(); i++) {
257
```

- 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.
  - http://www.grinninglizard.com/tinyxml2/index.html

  - Example for document parsing:

```
272
273     XMLDocument doc;
274     doc.LoadFile( "input.xml" );
275
```

  - Example for document navigation:

```
292
293     XMLElement* root = doc.FirstChildElement(matrixName);
294     XMLElement* row = root->FirstChildElement(rowName);
295     for (; row; row=row->NextSiblingElement()) {
296
297         XMLElement* cell = row->FirstChildElement(colName);
298
```

Standard Data Exchange Format - XML

XML Table Representation:

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

XML Matrix Representation:

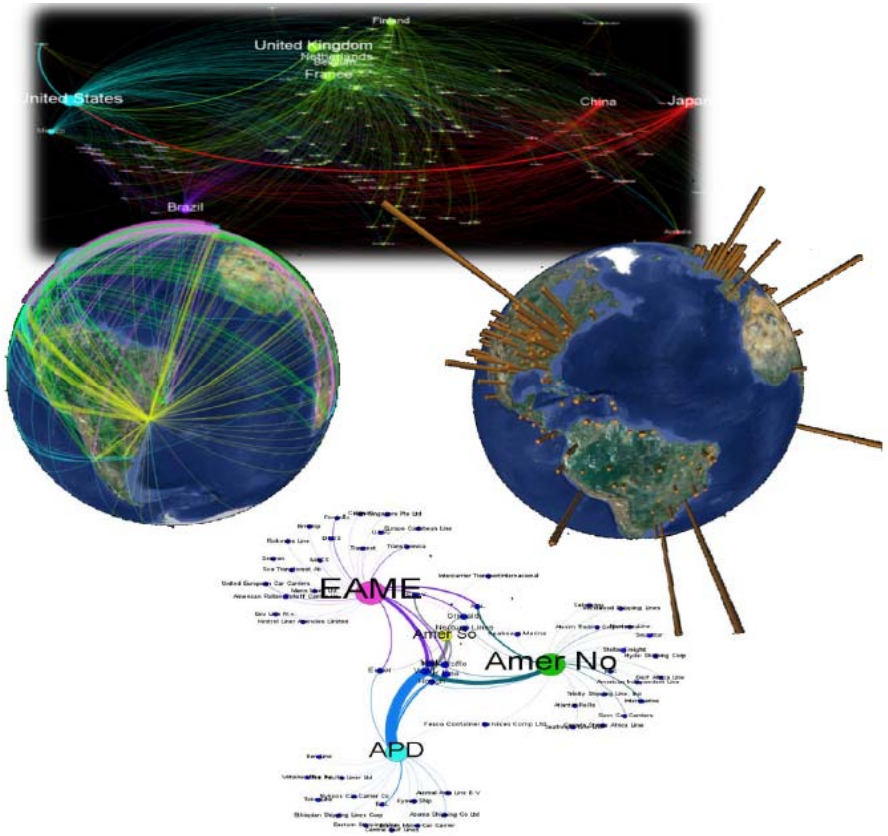
```
<data>
        <row_data>
            <row_name/>
            <cell>
                    <col_name/>
                    <cell_data/>
            </cell>
            ...
        </row_data>
        …
</data>
```

Experimental results
1000 iterations for 12 months

Operating system	Programming approach	# cores	Time
Windows	VBA	single / multiple core	N/A
HPC	VBA		1.5 months
Windows	C++	single core	500s
Linux	C++	single core	800s
Windows	C++	8 cores	60s
Linux	C++	16 cores	50s
HPC	C++		1.5s

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

  - Veluscek, M.; Kalganova, T.; Broomhead, P, "Improving Ant Colony Optimization Performance through Prediction of Best Termination Condition" IEEE International Conference on Industrial Technology, IEEE Computer Society. 2015.
  - Veluscek, Marco; Kalganova, Tatiana; Ogunbanwo, Adebowale; Williamson, Alina; Broomhead, Peter; Grichnik, Anthony, "Benchmarking of Meta-heuristic Algorithms for Real-World Transportation Network Optimization", International Journal of Production Research. Submitted 9/10/2014.
  - Ogunbanwo A., A. Williamson, M. Veluscek, R. Izsak, T. Kalganova and P. Broomhead (2014) Transportation Network Optimization. Encyclopaedia of Business Analytics and Optimization, February 2014, Ed. John Wang. pp. 2570-2583. DOI: 10.4018/978-1-4666-5202-6.ch229.