**G52GRP** Interim Group Report

29th November 2009

***A Visualisation Tool for a Choice Hyper-Heuristic***

*Group***gp09-exo**

Lao Jingqi jxl29u  
Zhang Chao cxz09u  
Ben Jenkinson bxj08u  
Thomas Barton txb18u  
Alexander Jermstad asj08u

*Supervisor*  
**Dr. Ender Ozcan** exo

Contents

[Problem Description 3](#_Toc247273717)

[Background Information & Research 3](#_Toc247273718)

[Heuristics 3](#_Toc247273719)

[Hyper-Heuristics 4](#_Toc247273720)

[What is VCH? 4](#_Toc247273721)

[Requirements Specification 5](#_Toc247273722)

[Functional Requirements: 5](#_Toc247273723)

[Non-functional Requirements: 5](#_Toc247273724)

[Initial Design 6](#_Toc247273725)

[Key implementation 6](#_Toc247273726)

[Discussion of Problems 7](#_Toc247273727)

[Technical Issues 7](#_Toc247273728)

[Non-Technical Issues 7](#_Toc247273729)

[Timeplan 8](#_Toc247273730)

[Appendix 8](#_Toc247273731)

# Problem Description

# Background Information & Research

## Heuristics

Finding a general solving algorithm is not always efficient: for example, some problems have a very completed methodology which would take a lot of time to implement but not widely-used. On the other hand, although there are many problem-solving computer algorithms, they are too knowledge-intensive to implement for most potential users. In such a case, people use heuristics as a substitution.

One single heuristic is an algorithm with a candidate solution and some constraints. A heuristic can be described with the following pseudo code:

|  |
| --- |
| BEGIN  INITIALIZE Candidate-solution  WHILE (Candidate-solution !satisfy Constraints)  Temp-solution 🡨 Heuristic-Algorithm(Candidate-solution)  IF (Temp-solution isBetterThan Candidate-solution)  Candidate-solution 🡨 Temp-solution  RETURN Candidate-solution  END |

The idea of heuristic is, there was generated a candidate solution which is represented by a array of *binary value* called domain bearer, then the heuristic algorithm would operate the domain bearer to generate a new candidate solution, then compare them through the requirements(or constraints) and choose the better one. Finally, if the programme meet the constraints and stopped then the current candidate solution would be returned.

Furthermore, some basic ideas of those heuristic algorithms can be represented as:

* **Flip**: since every bit in the domain bearer is a binary value, it is efficient to flip one or several of them to opposite values. And also, the number of flipped bits is flexible.
* **Swap**: this means to choose two bits in the domain bearer randomly and swap them as a pair; and there also can be one or several these pairs.
* **Combination:** that means to use both the **flip** and **swap** methods in one heuristic.

Usually, heuristics is easy to implement and maintain, so it can be used to solve the problems such as data-mining, cutting, packing, and scheduling; and it also has been used in the areas such as anti-virus technology. But the individual heuristics does not always work well: sometimes it can make serious error.

## Hyper-Heuristics

Since those existing problems of heuristics influenced its performance, hyper-heuristics was developed as an improvement. In general, a hyper-heuristic method is always described as “a heuristics to choose heuristics”, who operate on a specified domain of existing heuristics instead of the candidate solutions. Moreover, hyper-heuristics algorithms also have candidate solutions and constraints; but different from the low-level heuristics, hyper-heuristics would decide how to use low-level heuristics to generate new potential solutions rather than generate those new candidate solutions itself.

There are several heuristics-selection methods, such as:

* **Simple Random (SR)**, which select the low level heuristics randomly.
* **Greedy Random (GR)**, which apply all the low level heuristics to the candidate solution and choose the one who generated the best solution.
* **Choice Function (CF)**, which records the performance of each low level heuristics and compare them with several criteria.

In this project, we will mainly use the Simple Random (SR) method in the prototype, and then we will extend it with the other two. The pseudo code of hyper-heuristic can be probably described as:

|  |
| --- |
| BEGIN  INITIALIZE Candidate-solution  INITIALIZE Low-level-Heuristics[n]  WHILE (Candidate-solution !satisfied Constraints)  random 🡨 RANDOM()  a 🡨 random%n  Temp-solution 🡨 Low-level-Heuristics[a](Candidate-solution)  IF (Temp-solution isBetterThan Candidate-solution)  Candidate-solution 🡨 Temp-solution  RETURN Candidate-solution  END |

## What is VCH?

* A visualization tool
* Can give a direct and clear expression of hyper-heuristics to the users
* There is increasing interest about hyper-heuristics
* No such a software yet
* market research

# Requirements Specification

Requirements specification for the system to be built (agreed between the group and supervisor).

## Functional Requirements:

1. Implement the heuristics as well as the hyper-heuristics.
2. Ability for the user to select specific function for their requirements.
3. A number of standard heuristic choices to be available, at least three to begin with, with the option of adding additional ones if need be.

## Non-functional Requirements:

Programming language: JAVA

OS environment: mainly Windows

Project website: <http://code.google.com/p/vch/> subversion

\*use some diagram (use-case, dataflow, sequence….)

# Initial Design

* Software and hardware to be used, along with reasons for those decisions.
* Initial design of the proposed system and its user interface.
* Results of any initial implementation steps/prototyping.
* Results of technical research into suitable platforms, tools, technologies, algoritims, data structures, etc.

## Key implementation

* Implementation for a specific hyper-heuristics
* A base-user interface
* A process-display interface
* Attached functions such as show the history of the chooses each time, set low-level heuristics, set objective function, set the range of domain bearer, etc.

# Discussion of Problems

Being that this was the first time any member of the group had ever worked alongside so many others, one of the first problems we encountered was organising the workload, keeping to deadlines and making sure everyone understood what needed to be done. This was solved by having two weekly group meetings, one formal; with the project setter Ender Ozcan, the other, informal which was usually carried out in the Computer Labs.  
The formal meetings underpinned what was to be done over the upcoming weeks, keeping to a timeline that allowed us not to get caught behind. The informal meetings allowed us to keep in contact mid week to settle any problems we had with any of the work and help other group members out.

Initially one of the problems we face is the GUI and how to draw it efficiently, currently the graph is drawn using 1024 points, this requires, on every redraw that 1024 individual dots must be created on screen, the x y coordinates of these dots is calculated using the problem domain, the graph offset and the size of the graph. Currently this is the most efficient and accurate way we can think of for drawing the graph.

Another problem we encountered is when implementing the graph it dawned upon us that we need to choose where to run the calculations for the selected problem domain, being that our GUI is laid out using panels it seemed wise that each problem domain had its own panel, and that the calculations for the heuristic were run in its own panel, as opposed to the run of the main program.

When coding with others its hard to know what you can and cant edit, and what will get in each others way, when a lot of code us uploaded to the SVN at one time it is very hard to keep up with other group members, because everyone has access to other peoples code, once you have done some work you may leave it and come back to find it is completely different. This then leads to programmers being confused as to what needs to be done. In order to resolve this problem code must be well documented with reasonable variable names and comments explaining what each function and, if ambiguous, any other piece of code that isn’t simply understood. Despite this increasing the amount of time to program, when debugging it will make it much easier for other programmers to understand what has been done.

One problem we are faced with is the display of information, by using animations we have tried to make the heuristic as easy to follow as possible, however currently the graph is not displayed at the same time as the animation which could lead to confusion as to what is going on as the used may feel pressured to see the graph while it is only displayed for a few seconds as opposed to being able to see it when they wish.

-allowing user to input problem domain function

-displaying the information which is easily understandable

## Technical Issues

* Draw the dynamic diagram of the function
* Problems with the extend functions such as set the low-level heuristics by the user, read objective functions, set domain bearer, extended hyper-heuristics algorithms, etc.
* To organise and optimize the GUI

## Non-Technical Issues

* E.g. Individual work

# Timeplan

# Appendix