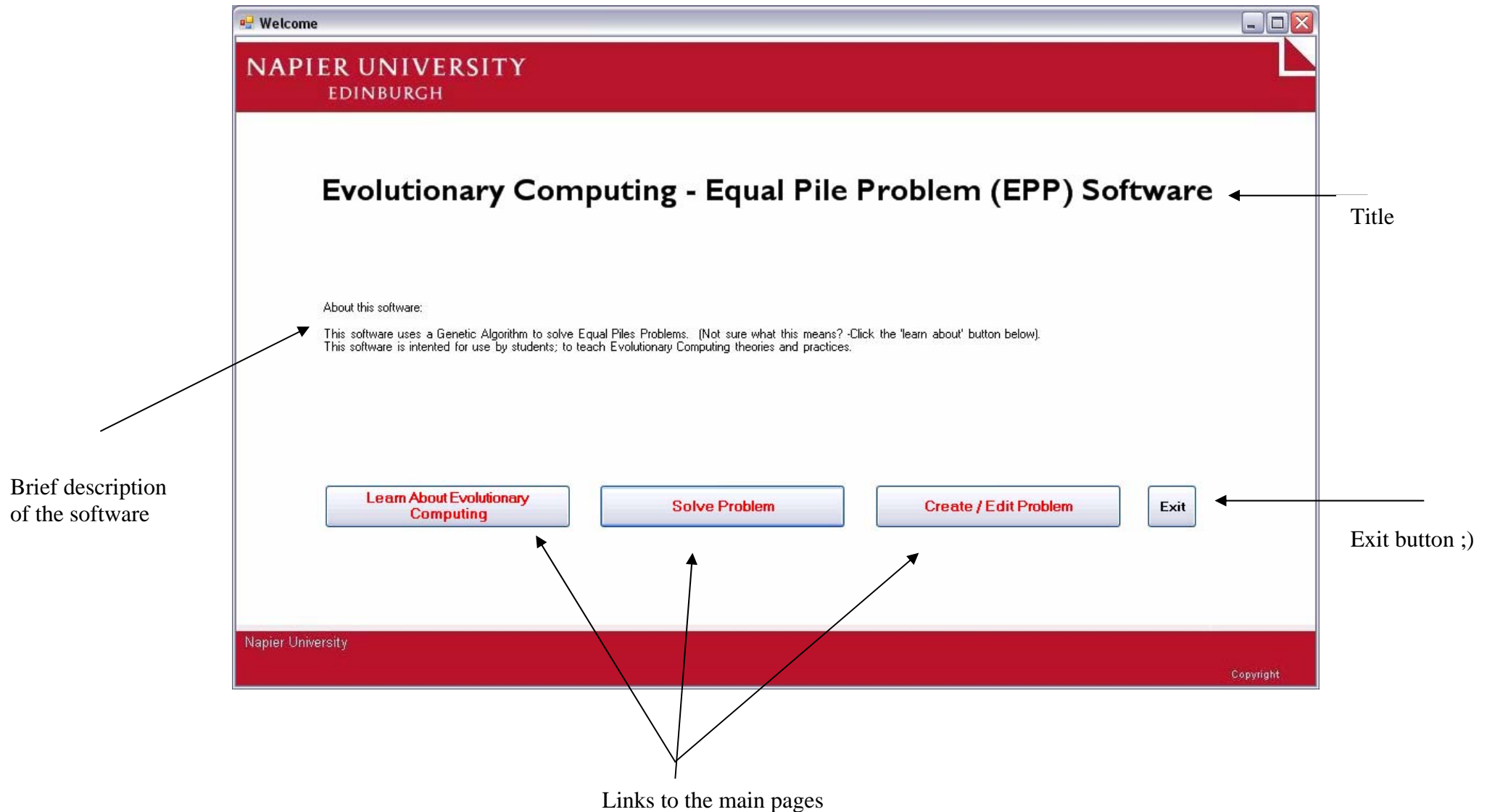


Emergent Computing Teaching Aid

A SIMPLE USERS GUIDE

Welcome and Menu Page



Learning Page

WWW link to the Napier University's Evolutionary Computing Module (Co42007) homepage

Evolutionary Computing information which is relevant to understanding the application

Learning

Menu

Introduction

Evolutionary Computing

Evolutionary Computing is the collective name for a range of problem-solving techniques based on principles of biological evolution, such as natural selection and genetic mutation. Visit the Evolutionary Computing module homepage [here](#)

Evolutionary Algorithms

Evolutionary Algorithms are a subset of Evolutionary Computing.

Evolutionary Algorithms use mechanisms inspired by biological evolution: Reproduction, Selection, Mutation, Recombination and Replacement. Fig 1.0 shows a typical evolutionary cycle.

There are 5 main types of Evolutionary Algorithm. Genetic Algorithms are the most common and one is used in this software.

For more information, see google search for [Evolutionary Algorithms](#)

Genetic Algorithms

Genetic Algorithms (GA) are a subset of Evolutionary Algorithms.

GAs are most commonly used to solve optimisation problems. GAs like EAs are stocastic in nature; they contain some element of randomness, this results in different answers being produced each time the algorithm is run.

What is an Equal Pile Problem

Example: You have 'X' items of varying weights, and you need to fit these items into 'N' amount of groups in such a way that the weight is distributed as evenly as possible. - This is an EPP.

In the real world EPP has many practical application including, equal weight distribution in transportation.

If your doing the Evolutionary Computing module (Co42007) lecture notes and tutorials are available on [WebCT](#)

The Evolutionary Cycle

Selection

Population

Parents

Recombination

Mutation

Offspring

Replacement

Fig 1.0

Learn about these mechanisms, these are changable parameters in the algorithm

Evolutionary cycle image

Links to parameter information page

Opens a Google web search for 'Evolutionary Algorithms', promoting further learning

Links to Napier's WebCT page

The user can set the 'max group height' this value defines the largest value of an item (or total value of a set of item) that can fit into one group.

The user can set the amount of groups they wish to have (between 2 and 10 groups)

The 'aqua' coloured rectangle is Group number 2, which contains three items.

Reset button, clears and resets the form to its original state (removes all items and groups) – re-enables the 'generate groups' button

Creates and displays the groups (and associated value label holders and fill buttons)

Adds a new item to the problem area

Neatly stacks all items, see next page

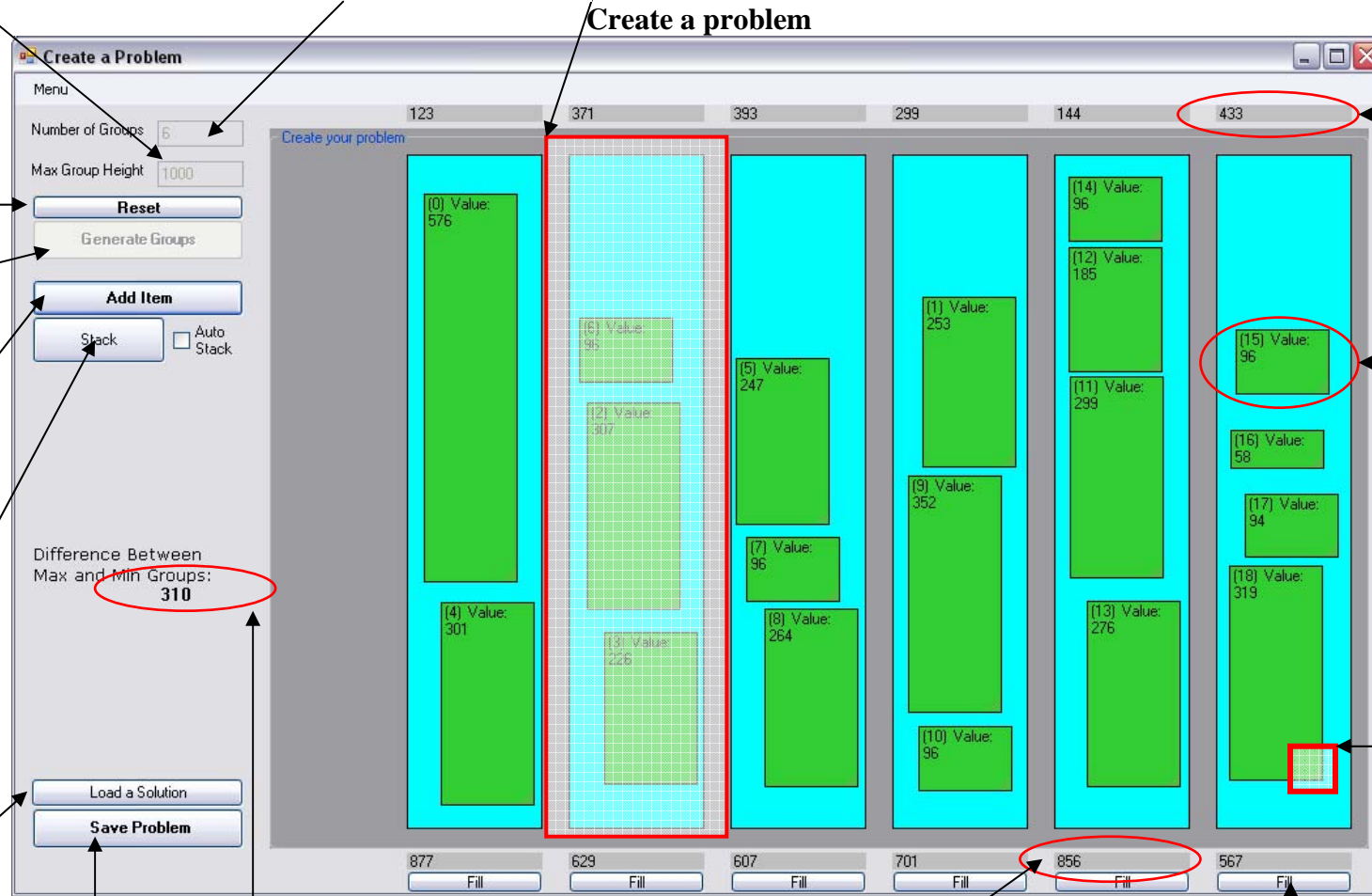
Loads in a solution and displays it on screen (see solution loaded page)

Allows the user to save a problem they have created to file

Shows the value of difference between the group with the largest value of items and the group with the lowest value of total items

Shows the sum of the values of all the items within this group

Fill button, fills the remaining group value (space) with an item. Fill functionality is illustrated in the next page.



Shows the remaining value (free space) left for group 6

Shows an item with a value (or weight) of 96. (15) indicates that this item was the 15th item to be added

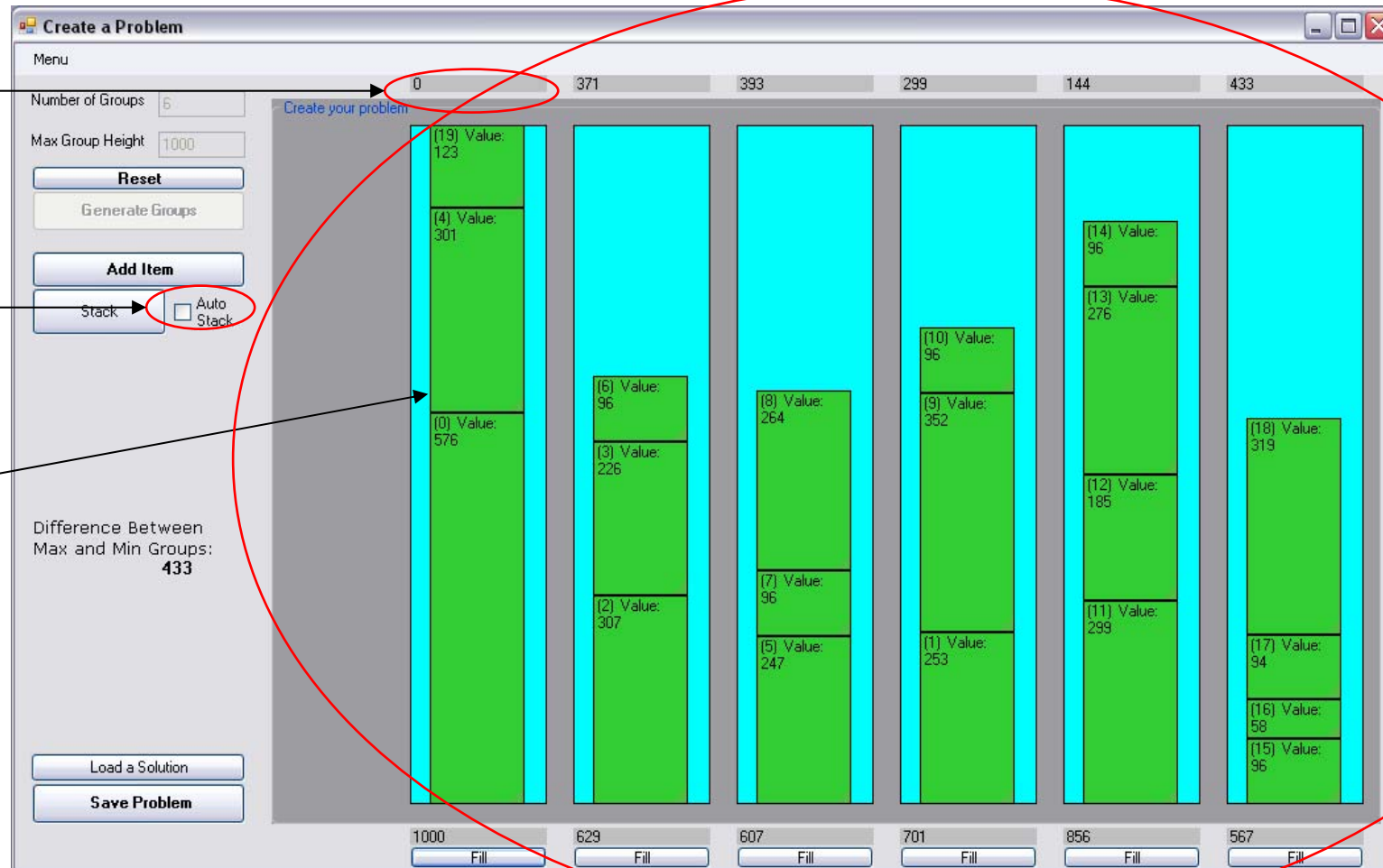
In the corner of an item there is a 'resizable' image, the user can click and drag this in order to **resize** the item

This pages illustrates the 'Fill' and 'Stack' functionality

Shows that group 1 is full as there is no remaining space

Automatically stacks all items once an item is placed in (or moved between) a group

Group 1 is full the fill button below that group has just been pressed.

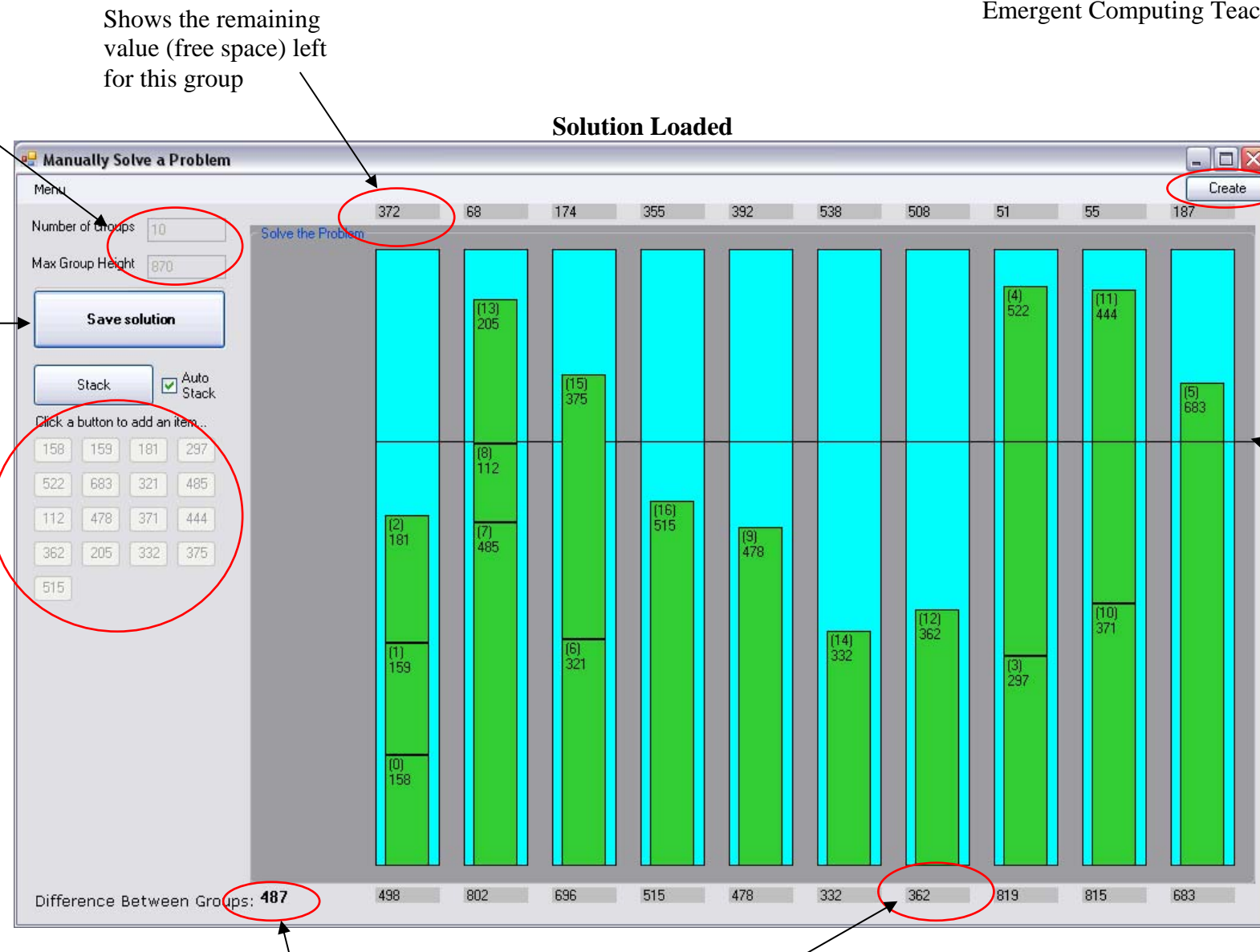


As you can see the items in all the groups have been neatly stacked, this is because the 'Stack' button has been pressed, the previous page illustrates non stacked items

These values are auto filled using the data loaded in from the solution file, the visual groups are also auto generated

Allows one to save a new solution to file

Shows the size of all the items in the solution, also shows that all the items have been added to the solve panel



This button resets the form and returns it to the 'Create a Problem' state as shown on page 4

This line represents the optimum solution (if there is one), and illustrates the total height that the items in each group should add up to

Shows the value of difference between the group with the largest total value of items and the group with the lowest value of total items

Shows the sum of the values of all the items within this group

The number of groups and max group height are automatically generated from the problem passed in

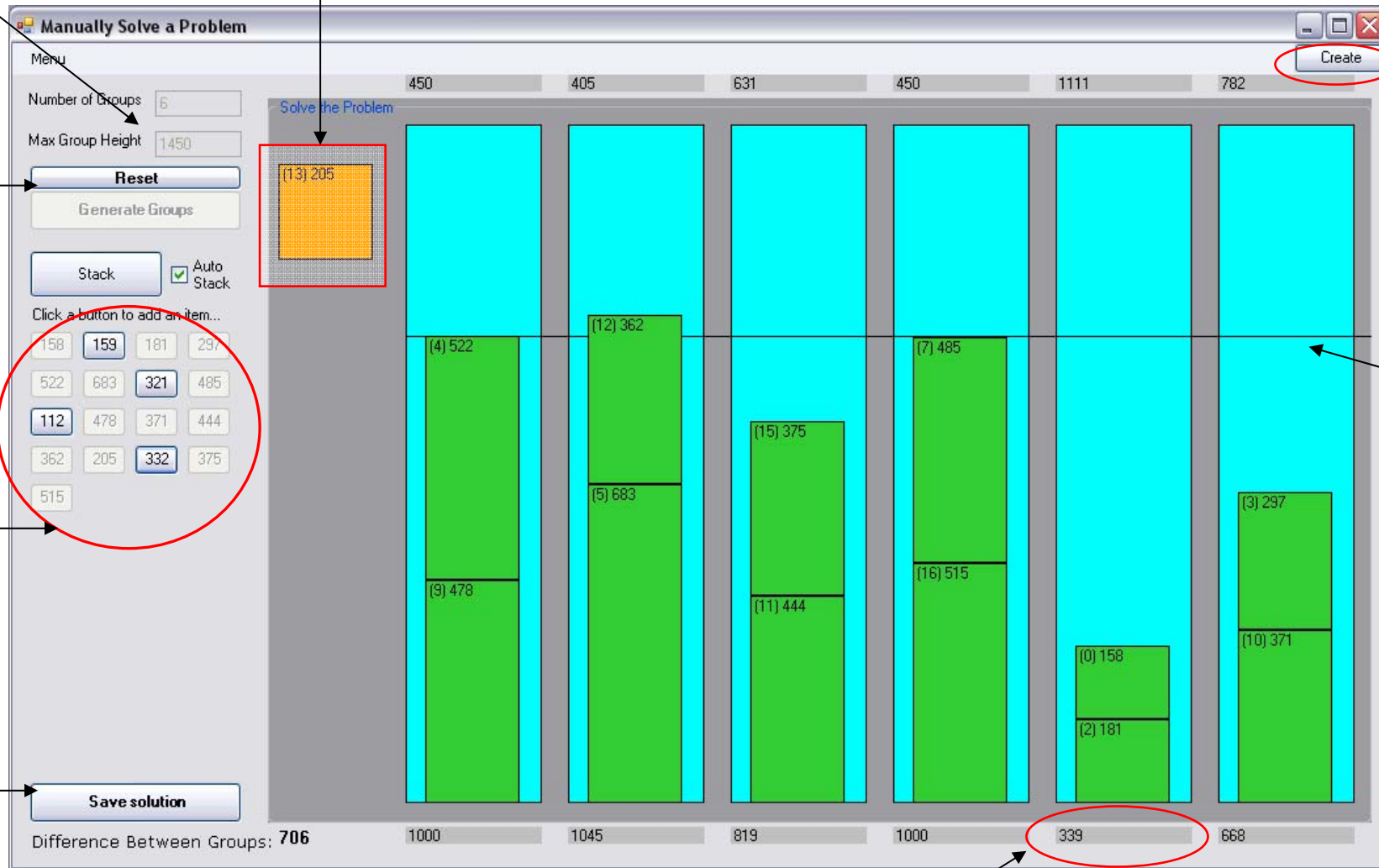
Reset the problem to its original state (no items added)

Represents all the items in the problem, when a button is clicked the associated item is added to the problem window (see orange item), the clicked button is then disabled so the item cannot be added again

Allows the user to save a solution to file

This orange item, indicates an item has been added to the problem window and needs to be placed, once placed inside a group the item will turn green

Manually Solve



This button resets the form and returns it to the 'Create a Problem' state as shown on page 4

This line represents the optimum solution (if there is one), and illustrates the total height that the items in each group should add up to

Shows the sum of the values of all the items within this group

By Julian Barrable

Displays the total values (numeric) of the groups sizes visually show in the panel to the left

Saves a solution to file (only available for problems with 52 items or less)

Allows the user to seed the population with a solution file

Allows user to solve the problem using other software

The about links provide detailed information about each associated button

Allows user to use enable swap mutation or use random mutation

Panel that shows the visual difference between total group sizes for a member of the population

Allows you to look at the groups sizes of other solutions, updates the panel above

The user can turn real time visual feedback on or off

The user selected how many groups the items will be placed into

Inform the user how many items are in the problem file

Displays the path that shows where the problem file is located

Simple Use

automatically loads in a test problem with 500 items

Pops up an 'Open' dialog box, allows the user to select a problem file

Takes the user to the Manual Solve page and loads in the currently loaded problem (providing the problem has 52 items or less)

Displays the best solution so far

Displays the total number of iterations the algorithm has performed on the problem, so far

Runs the algorithm (display progress page)

Initialises (resets) the problem

Control population size (solutions)

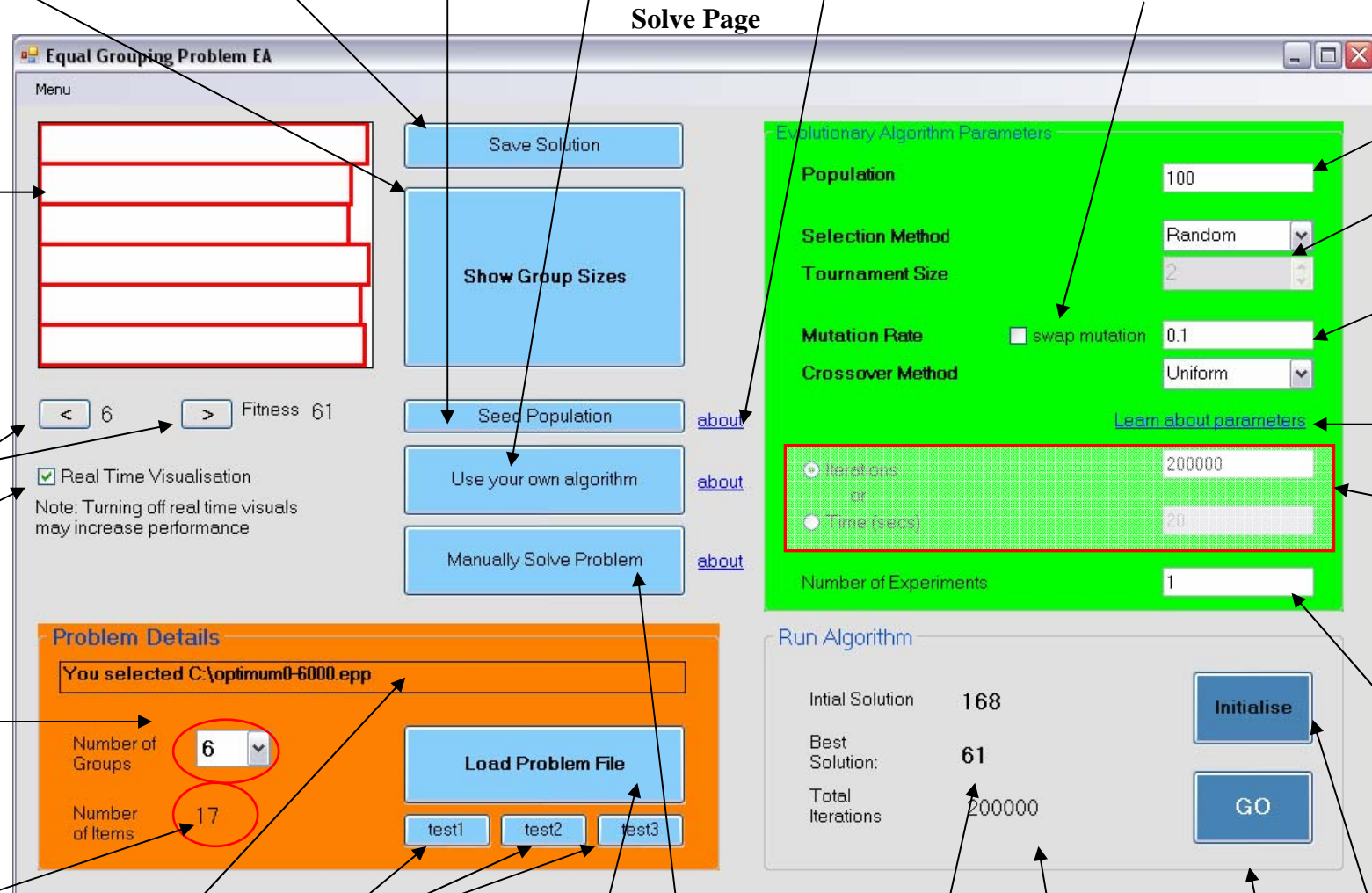
Selection method parameter controls

Controls mutation rate (must be between 1 and 0)

Links to the parameter information page

Settings that allow you to control how long the algorithm (or each experiment) will run for

When more than one experiment is conducted the best result for each experiment is saved to a text file along with the parameters used



Parameter Information

Activates drop
down menu

Learn

Menu

Population

The population is a pool of solutions. A valid solution is a representation that solves the given problem (see Representation below). For example if the population is 100, then this corresponds to pool of 100 solutions. The initial population is generated at random and contains good and bad solutions. The population improves over time as the algorithm runs, heuristics (rules) are used to ensure that worse solutions have a higher chance of getting replaced by better found ones.

Representation

In the application you are given, a direct representation is used to represent a candidate solution to a problem. If there are n items to be grouped into g groups, then a solution is represented as below:

Group that item
is placed in

1

Item 1

Group that item
is placed in

2

Item 2

Group that item
is placed in

2

Item 3

1	2	2	0	0	1	1
---	---	---	---	---	---	---

So, in the above example, there are 7 items, which must be placed into 3 groups. Item 1 goes in group 1, item, item 2 goes in group 2, item 3 goes in group 2, ... and item 7 goes in group 1. It doesn't matter how items are distributed amongst groups, and how many items go in each group - any solution described in this way is valid (though not necessarily very good!)

Crossover (aka Recombination)

Three crossover operators are available:

- Uniform crossover - forms a child by choosing the group an item should be placed in from either parent 1 or parent at random.
- 1-point - a random position within the chromosome is selected, say position p . The child is formed by copying the groups for items 1 to $(p-1)$ from parent 1, and the groups

Mutation

Every gene in the chromosome is considered for mutation. The gene is mutated with a probability defined by the rate entered in the application (which should be between 0 and 1). For example, if the mutation rate is set to 0.01, there is a 1 in 100 chance each gene is mutated. If the rate is set to 1.0, then every gene will be mutated. The mutation operator randomly selects a new group for the item being mutated.

Selection

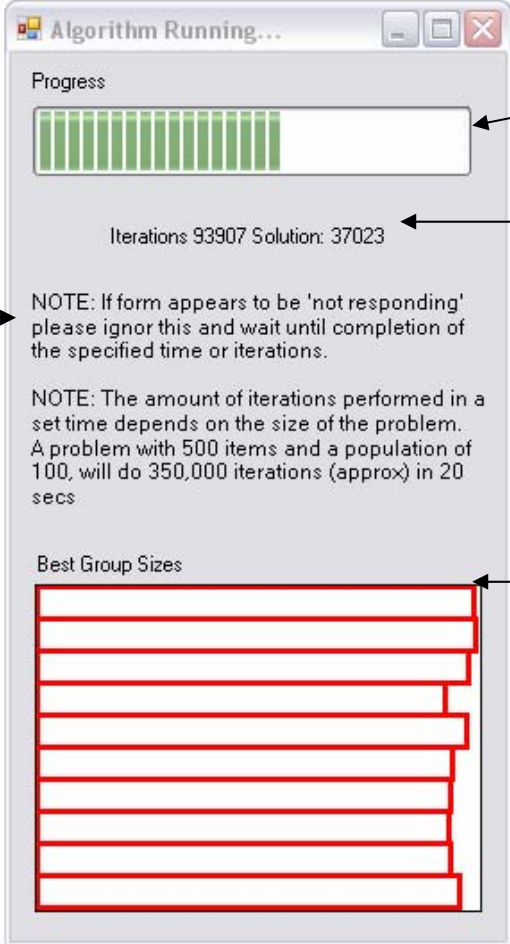
Two types of selection are allowed:

- Random - in this method, a single chromosome is randomly chosen from the population
- Tournament - in this method, s chromosomes are randomly selected. The selection method returns the best chromosome of this group. The size of the tournament (the group) can be altered in the application.

Information about
the mutation
parameter, a Hint for
setting a good
mutation rate is
given at the bottom
of this text

This page provides information about the various parameters that our used in the application. Information on the representation is also given.

Other Pages



The screenshot shows a window titled "Algorithm Running...". It contains a progress bar at the top with green segments. Below it, the text "Iterations 93907 Solution: 37023" is displayed. There are two notes: "NOTE: If form appears to be 'not responding' please ignore this and wait until completion of the specified time or iterations." and "NOTE: The amount of iterations performed in a set time depends on the size of the problem. A problem with 500 items and a population of 100, will do 350,000 iterations (approx) in 20 secs". At the bottom, there is a section titled "Best Group Sizes" with a table of red-outlined bars.

Useful information about the algorithm running

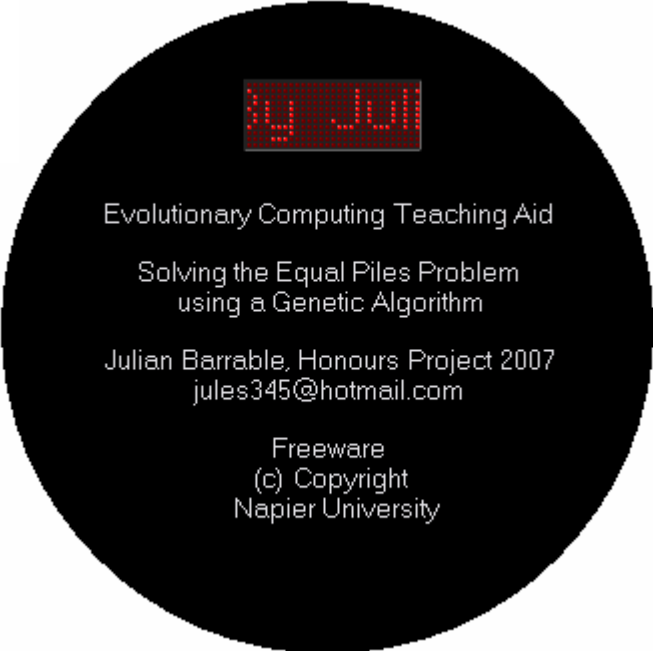
Progress bar which shows how long left until the algorithm finishes running

Displays the total amount of iterations and the best solution, these values update each time a new best solution is discovered

Updates, in real time, to display the total group sizes of the best solution so far

Progress Page

This pages provides real time information about the algorithm and problem being solved



The graphic is a black circle containing a red digital clock at the top showing "3:47". Below the clock, the text reads: "Evolutionary Computing Teaching Aid", "Solving the Equal Piles Problem using a Genetic Algorithm", "Julian Barrable, Honours Project 2007", "jules345@hotmail.com", "Freeware", "(c) Copyright", "Napier University".

About Page

Information about the application