Master Project Dissertation
Heriot Watt University

Project Title: Development of an application with the Play framework and CSPs libraries to compute the seating plan problem.

Student Name: OUVRY Paul
Student Course: Information Technology (Software Systems)
Supervisor Name: GABBAY Murdoch James

August 19, 2014
## Contents

List of Figures 5  
1 Signed Statement of Non-Plagiarism 7  
2 Acknowledgments 7  
3 Abstract 8  
4 Introduction 9  
  4.1 Overview 9  
  4.2 Objectives 10  
5 Background 11  
  5.1 Algorithm & Modelisation 11  
    5.1.1 Constraint Satisfaction Problem 11  
    5.1.2 Decidability 13  
    5.1.3 Computability 14  
    5.1.4 Graph Theory 15  
    5.1.5 Seating Plan Problem 16  
    5.1.6 Travelling Salesman Problem 17  
  5.2 Programming Language & Libraries 18  
    5.2.1 Web Based Application 18  
    5.2.2 Web Application Operation 19  
    5.2.3 CSP Libraries & Algorithms 20  
    5.2.4 Play! 22  
  5.3 Web Design & Web Users Behaviour 24  
    5.3.1 Global Architecture 25  
    5.3.2 Billboard Design 27  
    5.3.3 Elements Design 29  
    5.3.4 Customer Time Satisfaction 30  
6 Project Development Ways & Requirements 31  
  6.1 Methodology 31  
    6.1.1 Research Methods 31  
    6.1.2 Software Development Methodology 33  
  6.2 Requirements Analysis 36  
    6.2.1 Project Requirements 36  
    6.2.2 Research Requirements 37  
    6.2.3 Optional Requirements 37  
    6.2.4 Project Risks 38  
    6.2.5 Application Main Functionnalities 39  
    6.2.6 Basic Design 40
## 11 References & Bibliography

11.1 Books Referenced ........................................................................ 75
11.2 Journals Referenced ................................................................... 75
11.3 Web Documents & Web Pages Referenced ................................... 76
11.4 Bibliography ............................................................................. 76
11.5 Webography ............................................................................. 76
11.6 Figure Sources ......................................................................... 77

## 12 Appendix

12.1 Basic Design ............................................................................... 78
12.2 NQueens Problem ...................................................................... 79
   12.2.1 Choco Library .................................................................... 79
   12.2.2 Cream Library .................................................................. 80
   12.2.3 Google CP Library ............................................................. 81
   12.2.4 JaCoP Library .................................................................. 82
12.3 Travelling Salesman Problem ....................................................... 83
   12.3.1 Choco Library .................................................................. 83
   12.3.2 Google CP Library ............................................................. 84
   12.3.3 JaCoP Library .................................................................. 85
List of Figures

1 Decidability - Permutation Table ................................................. 13
2 Computability - Permutation Table .............................................. 14
3 Directed Weighted Graph .......................................................... 15
4 Rectangular Table & Round Table ............................................... 16
5 Complete Graph: K6 ................................................................ 16
6 Graph Usable for the Travelling Salesman Problem ....................... 17
7 Web Based Application ............................................................... 18
8 PHP Operation ........................................................................... 19
9 CSP Libraries Table .................................................................... 20
10 Logo Play ! ................................................................................ 22
11 Linkedin Logo ........................................................................... 22
12 The Guardian Logo ..................................................................... 22
13 Clicks to Completion .................................................................. 25
14 Obvious Hierarchy ...................................................................... 26
15 What Users See ........................................................................... 27
16 Page Design ................................................................................ 27
17 Sight Distribution ......................................................................... 28
18 Page Scroll and Page Size ............................................................ 28
19 Text Terminology ......................................................................... 29
20 Buttons Design ............................................................................ 29
21 Speed & Satisfaction ..................................................................... 30
22 Research Triangle ........................................................................ 31
23 Research Design Figure ............................................................... 32
24 Waterfall & V Models .................................................................. 33
25 Loop Software Development ........................................................ 34
26 Use Case: User Register - Login .................................................. 39
27 Use Case: User Other Tasks ........................................................... 39
28 Main Page Preview ....................................................................... 40
29 Login Page ................................................................................... 41
30 Signup Page ................................................................................ 41
31 New Page ..................................................................................... 42
32 People Page .................................................................................. 43
33 Load Page ..................................................................................... 44
34 Personal Page ............................................................................... 44
35 People Removing Table ............................................................... 45
36 Relation Optimisation Table ......................................................... 45
37 Seating Plan Page ......................................................................... 46
1 Signed Statement of Non-Plagiarism

DECLARATION

I, Paul OUVRY confirm that this work submitted for assessment is my own and is expressed in my own words. Any uses made within it of the works of other authors in any form (e.g. ideas, equations, figures, text, tables, programs) are properly acknowledged at any point of their use. A list of the references employed is included.

Signed:

Date: August 19, 2014

2 Acknowledgments

I would like to thank my supervisor, Professor M. Gabbay Murdoch James, who helps me during the whole project. Especially, to define the subject and the objectives of the project. He also continually guides me from the literature review until the end of the project with a lot of valuable advices.

This report would not have been possible without the unfailing support from my family. Each member stays as well as my friends confident about my work and remind an essential encouragement.
3 Abstract

This document is a master dissertation conducted by an IT project named: *Here Are The Seats*. The general purpose of this report is the literature review pre-development, the development of the application and the result of the research accomplished. The goal of this project is to develop a web application able to design a seating plan. This application will conduct the research in three areas: the *Play* framework, the Java CSPs (Constraint Satisfaction Problem) libraries and the web design.

The solving of the seating plan problem is the main task to perform. Before developing the application a literature review is done in order to expound the modelisation of the problem which is correlated to CSPs. In a second time, it is explained how the programming language and the libraries to develop the web application are chosen.

To develop this one the chosen language is Java. This one is chosen in order to develop a smart web application. The application is developed thanks to a Java framework named *Play !*. In addition, to compute the seating plan problem some Java libraries are tested. Two libraries were perfectly suitable to the problem: Choco and JaCoP.

The website was successfully implemented. Then research feedbacks are related in this report. The feedback given on this report is principally in favor of the *Play !* framework which is an interesting opportunity to develop websites. Finally, an evaluation mainly positive is given on several good practices used to develop a website.
4 Introduction

4.1 Overview

Some people have a lot of problems to design a seating plan for meetings or meals. Commonly, several people do not want or should not be sit close. Sometimes it is really difficult for the organizer to make the seating plan. It could be interesting to implement a web application which automatically creates a suitable seating plan in accordance with people affinities.

Placing people in a specific order can be desired for some reasons. In a professional context, it can be interesting to study the way to place people in order to make efficient meeting. In a context less conventional, it can be useful to design a seating plan for meals or receptions.

Then to solve this problem an application was designed. It was chosen to try to resolve it on a web application in order to ease the use and to avoid an installation for the user. The first part of this application is to allow the user inserting the different participants of the meeting. After that, the relations between participants and some information about them is inserted. Then the application can compute the best seating plan accordind to these data entered. Again the goal of this one is to make this work simpler for the user. Accordingly, the application has to be easy to use and intuitive.

In addition this application is a support for the research in some areas. Researches are done around the technologies used, the graphic design, the way of development, the mathematical algorithms, etc.
4.2 Objectives

The main aim of the project is to design a web application. Then this web application will conduct some research which are presented in this report. The goal of the web application is to compute a seating plan. To design it correctly a background is necessary around this area. The main objectives of these research are to expound how to solve the problem and to find out what can help the project to go on. A literature review was redacted, composed by three main parts:

Algorithm & Modelisation  Because the seating plan problem is a CSP (Constraint Satisfaction Problem), it will be lightened some way to ensure that it is computable and how to model it.

Programming Language & Libraries  Another objective is to define the libraries which will be compared. Some research will be introduced in relation with the choice of the libraries and then the adapted programming language to test them.

Web Design & Web Users Behaviour  Finally, in order to provide a final application which will be smart and user-friendly, some research will be expounded about how to design intelligently a web application.

Thanks to this literature review the objectives are clearly defined. The development of the application is the first one and will conduct the research objectives:

Play Framework!  The first one is the discover of the framework Play! and the establishment of an evaluation. This one can be compared to a web development using AMP (Apache, Mysql and Php) in order to point differences.

Java Libraries  Another objective is to test Java libraries available on the CSPs (Constraint Satisfaction Problems). Those ones are useful on the application, nevertheless, test a little further can be interesting.

Interoperability  Even if the Play! framework and the libraries work well the project can be compromised if both are not working together. The objective is to test the interoperability between them.

Web Design  To be suitable for the web, an application should have a well formed graphic design. The objective is to work around the web design to shape a smart application.
5 Background

The literature review is composed by three distinct parts. The first one is relative to the different review of the mathematical problems and the modelisation. The second one discusses the implementation of the algorithm, especially, the programming language and the libraries. The third is about the graphical rules of a web application design.

5.1 Algorithm & Modelisation

5.1.1 Constraint Satisfaction Problem

The constraint satisfaction problems (CSP) are a recurrent mathematical problems as it said by Vipin Kumar (1992) "A large number of problems in AI and other areas of computer science can be viewed as special cases of the constraint-satisfaction-problem" (AI: Artificial Intelligence). This mathematical domain is really large and this subject is mainly based on the satisfaction of constraints on defined objects. The object introduced can be restricted by constraints but also by limitations. The goal of the resolution algorithm is to find a solution satisfying all constraints and restrictions. Some existing algorithm were created to maximize the satisfaction of the constraints as much as possible.

A CSP problem can be resume as follow:

- \( X = \{X_1, X_2, ..., X_n\} \) is a set of different objects.
- \( D = \{D_1, D_2, ..., D_n\} \) is a set of different definition domains where \( D_i \) is the definition domain of \( X_i \), \( \forall i \in [1,n] \)
- \( C = \{C_1, C_2, ..., C_m\} \) is a set of constraint in relation with the definition domains.

The two main domains of CSPs are the artifical intelligence and the operations research. The second includes our problem. The operations research deals with anatytical methods to help to make good decisions. This is directly in relation with the subject: define the best seating plan thanks to people’s relations viewed like constraints.

The CSP are then really dependant on the constraints and the definition domains corre- sponding to variables. The computation of a valide solution can be really time consuming. In addition special CSP consist in minimize or maximize one or more variables. In that way, it is again more difficult first to find a solution, and then ensure this is the best one. To solve as well as possible problems it exist two main algorithm categories to solve CSP. Ones are exact algorithms which try only to find the best solution. Others are inexact algorithms which try to find the best solution they can. Exact algorithms can be really time consuming and not efficient but ensure that the solution found is the best one. Both types have advantages and drawbacks.
In order to answer to specific needs, the inexact algorithms are indexed in three different categories:

- **Backtracking**: The aim of this method is to go back when the research is blocked by the constraints and no solution is found. When no solution is going to satisfy the constraint, the algorithm go back until a choice made. Then, it can test another alternative and maybe find a solution.

- **Constraint Propagation**: The goal of this method is to take care of the contraints domains. This method pay attention if the domain is consistent with the problem constraints. For example, if $X_1 \in [0, 10]$ and $C_1 : X_1 \leq 5$, then it is possible to change $D_1$ the domain of $X_1$ to $D_{1*} = [0, 5]$.

- **Local Search**: The aim of this method is to take care about the first constraint and to satisfy it as much as possible. Obvisously the other contraints shoud not be violated. Then the algorithm do the same with an another constraint and keep trace of the result. The algorithm end happens when a solution is considered as relevant or when the time limit is reached.
5.1.2 Decidability

In the world of the computer science many problems exist. It is important to introduce here the notion of decidability. It exist some mathematical problems and some computer problems on which it is proved that it is not possible to find a solution. This is the main topic of the decidability. If it is possible to find a solution of a problem this one is called "decidable". In an other hand, if it is impossible to find a solution for a problem this one is called "undecidable".

It exists two kind of decidability : the logical decidability and the algorithmical decidability. On the first hand, the logical decidability mean that it is possible to find a solution to the problem using formulas and theorems. On the second hand, a problem is algorithmically decidable if it is possible to create a machine able to compute if the problem has a solution or not in a finite number of steps. Before to begin to design it is important ensure that the seating plan computation is a decidable problem. If this one is not, it will be impossible to compute the seating plan. By the way the application will not have any use.

A seating plan is how to place people around a table. One of the basic approach of this problem is to try each possible seating position and to compare the result. The seating plan problem have a finite number of possibility. Finally, this problem is not more than a permutation of a finite set. It can be introduced that the present problem is a discret one. If my seating plan is composed by \( n \) people : \( \{X_0, ..., X_n\} \). And if each seat is represented by a position in an ordered set the number of different seating plan is the number of possible permutations.

Example: If my seating plan is composed by three people : \( \{X_0, X_1, X_2\} \). The number of permutation is six : \([X_0, X_1, X_2], [X_0, X_2, X_1], [X_1, X_0, X_2], [X_1, X_2, X_0], [X_2, X_0, X_1], [X_2, X_1, X_0]\).

The number of permutation is \( n! = \prod_{i=1}^{n} i = n \times (n-1) \times (n-2) \times ... \times 2 \times 1 \)

<table>
<thead>
<tr>
<th>1!</th>
<th>(1)</th>
<th>(1)</th>
</tr>
</thead>
<tbody>
<tr>
<td>2!</td>
<td>(2\times1)</td>
<td>(2)</td>
</tr>
<tr>
<td>3!</td>
<td>(3\times2\times1)</td>
<td>(6)</td>
</tr>
<tr>
<td>10!</td>
<td>(10\times9\times...\times2\times1)</td>
<td>(3 , 628 , 800)</td>
</tr>
<tr>
<td>20!</td>
<td>(20\times19\times...\times2\times1)</td>
<td>(2 , 432 , 902 , 008 , 176 , 640 , 000)</td>
</tr>
</tbody>
</table>

Figure 1: Decidability - Permutation Table

Consequently while the factoriel number is not very huge it is always possible to determine how many permutations exist. The future application will not compute a very large seating plan. Then it is possible to affirm that the problem is decidable.
5.1.3 Computability

The computability is a notion in relation with the decidability. Decidability is a good point, nevertheless the second point to check is the computability. As is mentioned by Homer and Selman (2011, p.23) the computability is created because of today’s computers are not enough efficient to compute all mathematical problems. A problem is computable if it is possible to find a solution to the problem before a time limit. First of all if a problem is not decidable it cannot be computable. Obviously if it is proved that the solution does not exist it will be impossible to compute something which does not exist. The main issue covered by the computability is: even if the problem is decidable (i.e a solution exists proved by mathematics) it is sometimes impossible to compute it.

Again before beginning to design the web application it is important to be sure that the seating plan problem is calculable. Again without this point the application can be totally useless.

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1!</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2!</td>
<td>2x1</td>
<td>2</td>
</tr>
<tr>
<td>3!</td>
<td>3x2x1</td>
<td>6</td>
</tr>
<tr>
<td>10!</td>
<td>10x9x...x2x1</td>
<td>3 628 800</td>
</tr>
<tr>
<td>20!</td>
<td>20x19x...x2x1</td>
<td>2 432 902 008 176 640 000</td>
</tr>
</tbody>
</table>

Figure 2: Computability - Permutation Table

A seating plan is no more than a permutation of a finite set (explained in the last subsection). Then the first estimation can be done around the efficiency of the application. If we considere that the seating plan introduced no more than:

- 10 people. We can compute the 3 628 800 possible seating plans quickly to find the best solution.
- 20 people. That means that 2 432 902 008 176 640 000 permutations exist and should be tested.

This kind of computation should be really time consuming and not really efficient done by a basic computer or server. Nevertheless the application will not use the exhaustive search and try all computations when the number of people inserted is too large.

Some algorithms around the CSP and the efficiency of the permutation can take off some permutations not pertinent. Accordingly to this explanation the efficiency of the libraries used to develop the web application will etablish the limits of this one. Especially, the maximum number of people insertable on the web application.

However it is possible to conclude that for a reasonable number of people, the seating plan problem will be computable.
5.1.4 Graph Theory

The graph theory is a mathetical and informatical theory around graphs. A graph is designed by one or more vertices and by edges which link the nodes. Each edge links two nodes. Wilson and Watkins (1990, pp. 1-14) explained basic knowledges around graphs. For example, here are the basic types of graph:

- Undirected graph: this one is the most basic one. That’s mean we do not take care if edges are directed or not.

- Directed graph: this graph is the same than undirected graph but each edge is directed. In that graph, it is possible to find an edge which allows going from the node A to the node B but not from B to A.

- Weighted graph: on this one a number is affected to each edge.

It exists some other kind of graphs and some combinations of this one. For example, a directed weighted graph is one of them.

![Directed Weighthed Graph](image)

The goal of graphs is to model mathematical or logical problems. Thanks to these modellisations a lot of problems can be demonstrated with algorithms on it. That is why, in order to solve a lot of mathematical issues several type of graph exist. Some constraint satisfaction problems are modelised by graphs. One of the most famous ones is around the graph colouring. The issue is how many colours are needed to color a graph without two linked nodes with the same colour.

Two others really common problems are:

- The shortest path problem. Problem which is to find the shortest path between two nodes in a graph. Often solved by the Dijkstra algorithm. (Weighted graph)

- The travelling salesman problem. Problem which is to find the shortest way to go throught all nodes in the graph. (Weighted graph)

The last problem will be detailed in a next part.
5.1.5 Seating Plan Problem

The seating plan problem can be modelised as a graph problem. Seats can be represented by vertices and relations between people can be represented by a weighted edges.

Here are two examples of graph modelisation, suitable for a seating plan:

Two shapes are drawn above. The first one represent a rectangular table with several relations. Two people who sit close have a relation and two people who sit in front of another one have a relation too. The second one represent a round table. In this one the model is more simple, a person only has relation with his two neighbors. The complexity of the problem is: different shapes can introduce some constraints which are not existing in another shape. Consequently, the difficulty of the graph infers on the complexity of the best solution. And may as well change the algorithm approach.

Basically the relation between people is a complete graph. A complete graph is a graph where each pair of nodes are linked by an edge. To compute a solution first we have to enquiry from the user about the relation's values. And after that remove edges inessential to comply with the shape above.
5.1.6 Travelling Salesman Problem

A famous CSP is the travelling salesman problem. A salesman should go through a number of town on a map to sell his product. To be efficient the salesman should use the shortest way to travel by all expected towns. The travelling salesman problem is to find this shortest route.

A basic modelling of this problem is by using graphs. Each vertex represents a town to visit and each weighted edge the distance between them. After that, the aim is to choose on the graph the shortest way to link the nodes.

To introduce this problem we will only explain the symetric problems. Signifying that the route who link the point A to the point B is the same than to link the point B to the point A. That’s means that this graph is undirected. If it is possible to go from A to B, it is as well possible to go from B to A and these two routes have the same distance value.

![Graph Usable for the Travelling Salesman Problem](image)

The travelling salesman problem is an extended problem from another one named the Hamiltonian cycle. An Hamiltonian cycle is a cycle in a graph which goes through all vertices but only one time as is described by Lawler et al.(1985, p.2)

The graphic above a is not very large. Accordingly it is simple to find an Hamiltonian cycle, for example: \( A \rightarrow B \rightarrow C \rightarrow D \rightarrow E \rightarrow A \). Yet the next step to resolve the travelling salesman problem is to find the shortest Hamiltonian cycle. Here the good solution is obvious: \( A \rightarrow B \rightarrow C \rightarrow D \rightarrow E \rightarrow A \) with a total of twenty-four. Because the presented graph contains only five nodes and only height edges, it is easy to find the solution. But more the graph is large more is difficult to find the solution.
5.2 Programming Language & Libraries

5.2.1 Web Based Application

A web based application is an application which works on the web. This kind of application is available in a web browser which is a link between a client computer and a provider server.

In order to create this kind of application a common way to achieve it is to use a client programming language and a server programming language and to mix both to develop the application. A client programming language means that the developer will develop code source file and this one will be launched and performed on the computer of the user. In contrast, the server programming language will be executed on the server where is accessible the web application. As is mentionned by Bhakti Satalkar (n.d) web application have many advantages. First of all, this kind of service is really designed to ease the use. No installation is needed; only a common web browser. Information might be accessed from any computer on the internet. In addition, the development of a web application is quite often cheaper than a basic software.

The web based applications are today still in expansion. Because of the oncoming of the cloud computing this kind of application are growing in popularity. The cloud computing consist in the use of remote servers by a network. These remote servers are usable for different services because they can have a strong efficiency on computation or thanks to their huge storage capacity as is detailed by Baun et al (2011, pp.5-22). This way of development support the commercial exploitation model named SaaS: Software as a Service. Then software are not installed on the client’s computer but available online as any services.

The web application are basically conceived on three tier architecture. A three tier web application in an application which is created with three independant modules: the presentation tier, the logic tier and the data tier as is mentioned by Gallaugher J. and Ramanathan S. (1996). The presentation tier is the visualisation displayed to the user, the logic tier is the main body of the application. The data tier is essentially composed by a database or a file system.

Thanks to all these details we will go deeper in how work web applications to define how can be designed the future seating plan web application.
5.2.2 Web Application Operation

One of the requirement of the application is to develop an application available on internet. This application is a type of application which really needs to be accessible quickly and must be on internet because of the subject covered.

In order to develop the web application a server will be needed: to store the seating plan in a database, to compute the seating plan, etc. There are three usual server programming languages to develop website:

- PHP: Hypertext Preprocessor
- JSP: Java Server Pages
- ASP: Active Server Pages

![Figure 8: PHP Operation](image)

To explain how works a web system, the common use of the PHP will be explained. First of all, a client on his computer wants to access to a web page. His web browser send a request to the HTTP server corresponding. His page request go through the internet to find the specific server. When the server received the customer request he loads the page containing some PHP. Then the server recognize that this page contains some PHP and calls a PHP interpreter. The PHP Interpreter transforms the PHP code to HTML code often with specific data from the client. Finally, the HTML web page is sent back to the client web browser and is displayed.

This process is common in the internet because some data is often stored on servers and computation are also done on it. Both, will be needed in the seating plan application.
5.2.3 CSP Libraries & Algorithms

As it explained in the last parts, the CSPs are a very well known area. Some researchers still work on it to find new algorithms more efficient due to the complexity and the huge number of usage possible. One of the powerful programming language to resolve CSPs is Prolog which is based around the logic programmation. In order to do some researches around others areas, Prolog will be not used.

Algorithms The current algorithms to resolve CSPs are the AC-3 and AC-2001. Both work around the Arc Consistency as is explained by Lecoutre, C., Boussemart F., Hemery F (2003). The Arc Consistency is the checking of the variable’s domains. Each variable in a CSPs has a definition domain. Yet, this domain can be reduced because of the constraints assigned to the problem. These two algorithms work around this area. The AC-2001 is an improvement of the AC-3 algorithm and is considered today as the most efficient. The reduction of variables’s domain is really important because that reduce as well the number of possible solutions. Then the algorithm can be more efficient, and in that case, reduce the time consumed to find a good solution.

Libraries Here is an overview of the libraries which solve the CSP:

<table>
<thead>
<tr>
<th></th>
<th>Cassowary</th>
<th>Choco</th>
<th>Cream</th>
<th>Google CP Solver</th>
</tr>
</thead>
<tbody>
<tr>
<td>Java</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>JaCop</td>
<td></td>
<td>JOpt</td>
<td>OptaPlanner</td>
<td></td>
</tr>
<tr>
<td>Scala</td>
<td>OscaR</td>
<td>Copris</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Python</td>
<td>Python Constraint</td>
<td>Emma</td>
<td>Google CP Solver</td>
<td>Number Jack</td>
</tr>
</tbody>
</table>

Figure 9: CSP Libraries Table

All libraries not divulge which algorithms they use. Nevertheless, at least Choco, JOpt and Google CP Solver use the AC-3 or AC-2001 algorithm.

After this research it was decided to develop the application using the language Java. First of all, because it exists several libraries and it could be interesting to test them. It can be useful to have several points to test and several libraries. These libraries are only known by the provided documentation, some of them can be not efficient or not usable in the seating plan problem case. Accordingly, it is interesting to have high number of libraries to be sure that at least some of it should be working.

In another hand Java is a language which permits to create web application. The JSP technique allows generating HTML using the JAVA programming language as is mentioned by Downey (2012, p.20). Yet another way to use JAVA to generate HTML is to use the framework Play! presented in the next part. Finally the choice of these languages are in agreement with most used programming languages as is reported on the Tiobe Software website (n.d).
In the choice of the libraries only the free licensed ones were chosen. Only these one can be used without the agreement from the authors. The aim of the application is to use the libraries to test and compare them. Accordingly only the free licensed libraries can be used. Nevertheless some software’s owners could be possibly efficient like ILOG CP or Artelys Kalis. Here is an overview of the libraries:

**Cassowary**  This library is not more maintained so we will no introduce it in the project.

**Choco**  (X11 - Style License) Choco is a library usable for teaching, research and real-life application developed by a french school named: *Ecole des Mines de Nantes*. This library implemented in Java is originated from 1999 and some engineer are still working on it. It is possible to model some common problem like the travelling salesman problem explained in the algorithm part of this report.

**Cream**  (GNU GPL License) Cream is a CSP library developed between 2004 and 2009. The last module released are Copris a version in Scala of the solver released in 2011 and Sugar a efficient SAT constraint solver. This one was developed by Naoyuki Tamura and the helps from others contributers.

**Google CP Solver**  (Apache License Version 2) Google CP Solver is an operating research tools developed in C++, Python, Java, and .NET.

**JaCoP**  (GNU GPL License) JaCoP is a large project than the other ones. Two core developers actively work on it since 2001. Some papers related to JaCoP were published around the research of CSP. The basic CSPs like Hamiltonian circuit can be solved with this one. The publishers really take care to provide example to show their work. It can be important for the future development.

**JOpt**  (Open Source) JOpt is a very wide project. This one includes an OPL (Optimization Programming Language) language parser, a solver for CSP, a MP (Mathematical Programming) solver and a scheduling solver. It provides as well some features to compute the TSP (Travelling Salesman Problem).

**OptaPlanner**  (Apache License) OptaPlanner is a library to optimize business resource usage by constraint solving. This one is released in 2014. One of the basic example is to solve the TSP and more complex problems.

At this time, it is difficult to argue around the libraries’ differences. Ones seem to be designed more conscientiously than others. But it is impossible to state around the possible efficiency of the algorithms. This master project research will answer as much as possible to this issue.
5.2.4 Play!

**Presentation** Play! is an open source framework designed to create easily web applications. It was created to ease the development of web applications. This one is available in Java since 2007 and in Scala since 2012. The particularity of this framework is that this one is not based on the Java Servlet Engine.

Play 2!, the new version, is in comparison to the other frameworks a stateless one. There are no relations between two requests and the server does not retain session informations. Moreover, this one is considered like a RESTfull framework: special architecture for distributed hypermedia systems.

![Play Logo](play.png)

**Figure 10: Logo Play!**

**Advantages** Play! has some advantages, that's why even in the web technologies which is an area really mature they are more and more known. Even if this framework is not really famous nowadays, some notable websites were developed with it: the professional social network Linkedin or the website of the newspaper the guardian.

![Linkedin Logo](linkedin.png)

**Figure 11: Linkedin Logo**

![The Guardian Logo](guardian.png)

**Figure 12: The Guardian Logo**

In addition, the framework Play supports a lot of IDE (Integrated Development Environment): Eclipse, NetBeans, IntelliJ, Textmate, vi, etc. Another advantage of the Play framework is that he allows bookmarking pages.

The authors also really took care of the developers. Errors are mainly well explained. It is important especially when the technology is really young. Moreover, the framework is designed around the MVC (Model - View - Controller) pattern essential for a well designed development. Accordingly it is easy to extend a development thanks to the module architecture.
**Drawback** One of the huge drawback of the framework Play! is like all recent softwares or libraries: the small community using the technology. Consequently, it is sometimes difficult to find a good explanation for not common errors.

Some users of this library think that the authors work more and more on the Scala project unlike the Java project. Nevertheless, this purpose is not official and nowadays the Java one is not abandoned.

The Play! framework is under the Apache 2 License and can be used free of charge accordingly to conditions mentioned in the license.
5.3 Web Design & Web Users Behaviour

The web design is something which developers really need to take care. First because the graphical design is the first view proposed to the customer who is coming on a website. It can attract or reject people, so it is really important. The first sight is like a lot of things determinant for the user appreciation. The seating plan problem web application is like most of websites: it will be designed to be used by people. Accordingly this one should satisfy as well as possible the user.

Referring to Krug (2006, p.18) "On the Internet, the competition is always just on click away, so if you frustrate users they'll head somewhere else.". Because of the huge number of website on the Internet, a lot of website are talking about close subjects. If a website not satisfy the user this one can use another one to find what he is looking for. It is important to know that a web user not stay in average more than a minute by page as is reported by Nielsen (2011). In addition some people not stay more than between ten and twenty seconds. That seems owing to the difference of quality of web pages and the difficulty to find what the user look for. Then the first sight on a website is clearly decisive for the end of the user experience.

Consequently, the first point is to really take care of the Home Page in the website conception. It should introduce all the content available on the website. Nevertheless, the developer should be careful to create something clear and simple. Accordingly to Krug (2006, p.99) an Home Page should contain the answer to four questions often asked by the user:

- What is this website ?
- What can I do on it ?
- What is the content ?
- Why I chose this one and not another one ?

In addition, the user habits can varied accordingly to nationality. As it is evaluated by Cyr D. et al there are some differences between the navigation, the availability, etc. Nevertheless, the design of a website can be divided in several main topics which will be studied in the next pages:

- The website architecture .
- The web pages space design.
- The web pages elements.

Something else really important about the availability and the attractiveness is the ranking of the website in the search engine. The specific area is known as the Search Engine Optimization (SEO). Yet, this one is not relevant for the current project because the produced web application will certainly not be published on the web. Consequently the next parts will review the three basic points listed above.
5.3.1 Global Architecture

The global architecture of the website is: how the structure of the website is designed. For example the number of pages, the links between pages, etc.

As it said below an user spends as little time as possible on a website because he wants to find what he is looking for as quickly as possible. The user is commonly in a hurry. Accordingly and for other reasons the global architecture should be as simple as possible. First if the website is simple the user will understand quickly how it is designed and where he could find what he is looking for. Make him satisfied is a good point and could convince him to come back for other information.

A non-official old rules was called the *The Three-Click Rule*. This one reported that because of the frustration of users all data in a website should be available in a maximum of three clicks. Like is expounded on the article of C. Chapman (2010) it is possible to don’t follow this rule. Nevertheless, the process to access to the information should ease the use and be clear to guide the user to what he is looking for.

![Figure 13: Clicks to Completion](image)

Figure 13: Clicks to Completion
As is mentioned by Krug (2006, p.32) one of the best way to structure a web site is to create an obvious hierarchy between the part in the website. Provide more space for important details. More the information is important more the information should be high, etc.

![Figure 14: Obvious Hierarchy](image)

As it said before, in order to make the architecture understandable for the user this one must be easy to visualize. To do that menu with items can be created. Each link item should have the same position in the architecture. Similarly, the same rules can be applied for a submenu. A tabbed pane is for example something really well designed because it allows to the user knowing everytime in which tab he is.

Nowadays a current feature present in some websites is a site map. Obviously, one bad thing for an user is to be lost cause of the substantial amount of pages. Site map prevent this effect. A good practice is to clearly guide the user in the architecture. One way can be to present continually the position of the user in the architecture. This process is usually space expensive and might take the place needed for informations. Another way to make it clear is to call each time the page by the name of the link where the user click. This allows ensuring the user that he is in the right page and secondly to avoid the misunderstanding between titles and pages.

Working around this aspect is more and more common. In addition tools can be added to a website. Those tools can be added to collect website information. One of them is google analytics which for example let the user know statistic like how many people went on webpages, which page is not used, etc. Those tools are useful to control how a website is used and how to improve it.
5.3.2 Billboard Design

As well as the global architecture the billboard design of an application is very important. The billboard includes the subject: how to place web elements on a page. First knowing where is looking an user on a web page can be useful.

As it shown above the user look at different points depending on his interest on the website as is described by Krug (2006, p.23).

In addition the user is always expecting obvious design. The title should commonly be on the top of the pages and the menu on the top or on the left. Those user habits and conventionnal rules ease the use of the user. After that it should be the title of the page and the global content of the that page.
Something else which seems really important is the distribution of the element on the page. As it shown on the diagram below, people look more on the left than on the right of the page. Then important information need to be on the left.

![Figure 17: Sight Distribution](image)

In addition, one detail which really slow down the user is the noise. The noise is all information not useful for the user. Advertising is for example an unexpected noise. Redundant information can be detrimental too. It was already explained in last pages: more the website is simple more the user will find quickly what he is looking for.

![Figure 18: Page Scroll and Page Size](image)

Finally, as it yarnd by C. Chapman (2010) the length of the page has no effect on the possible scrolling of the page by the user. Whatever the length of the page the number of people who scroll until the end is the same. However the information placed on the top of a page (available without scrolling) will be sight by more people. Conventionnaly, more the purpose is in the bottom of the page less it is important.
5.3.3 Elements Design

As is developed by Krug (2006, p.23) the best way to do a website simple is to don’t let the user think. The goal is to design everything self-evident. Some features bad implemented could interfere during the use of the website. On the figure below the terminology used is important. The exemple is made with just a button. If the terminology is simple the user will not think and directly go to the next page. He will finally find quicker his information and will be more satisfied.

![Text Terminology](image1)

Figure 19: Text Terminology

On the next figure another point is highlighted. Even if the terminology is good the visual design should be as well smart. That point is in relation with the part around the global architecture and the site map. In order to be understandable it is important to make sure that everything clickable will be obviously clickable. Two things are important for that. First design a button like a button as it is displayed. Secondly, used as much as possible same design for same elements. Each button should be exactly designed than others. By the way, the user can easily reconized what is an element where an action can be launched.

![Buttons Design](image2)

Figure 20: Buttons Design

Finally, it is recognized that there is not only one way to design a website. It exists some conventional manners to improve the attractiveness and the usability. Nevertheless, the developer always has to make some choices and good ideas are always well accepted.
5.3.4 Customer Time Satisfaction

The customer time satisfaction is a very important issue for web based application. A common behaviour of internet users is to leave a website if this one is too long to display what they are looking for. Some statistics were done around this subject. Fauconnier (2011) reveals that a customer spend in average 72 seconds on a website before leaving and finding out another one.

The main goal of the web application will be to provide the seating plan which comply with the data inserted by the user. In order to be sure that the user will want to use the application the website should compute a solution very quickly. Then the time satisfaction is an issue on which it needs to pay attention.

In addition, accordingly with Forrest (2009) studies demonstrate that customer satisfaction decrease when the time of tasks like clicks grow up.

![Figure 21: Speed & Satisfaction](Image)

Consequently and ideally the seating plan application should be quick. Normally, one solution to answer quickly is to give to the user a partial solution of the computed result. But in our case it will not be possible to give half of a seating plan. A property of the CSPs is that a solution satisfy all the constraints. Accordingly, the whole solution must be computed entirely to ensure that everything is satisfied.

To ensure good use conditions for the customer a maximum time could be fixed for the computation. Then the application should compute a result only during this period of time. As is explained in the first part of the literature review the CSPs are solvable by exact methods as well as approached method. Depending on the amount of data inserted the application must choose the good method to use to be as much as possible quick and efficient.

The approached methods are more quicker than the exact ones, but they can be not precise. Yet it is possible that a seating plan which is not the best one satisfies the customer. A process of proposition should done in the project to know if the user wants the best solution or only a valid one. Anyway, developers should inform the customer when the proposed solution is not the best.
6 Project Development Ways & Requirements

6.1 Methodology

6.1.1 Research Methods

The research objectives of this master project and dissertation are:
- Compare some libraries to compute the seating plan problem.
- Give a feedback around the use of the Framework Play !.

These research objectives are in relation with two other objectives which are: develop an application usable for an user and work around the graphical design suitability.

Accordingly to Jonker and Pennink (2010, p.25) a research methodology incorporates four different levels. As it shown on the figure below these ones are: the research paradigm, the research methodology, the research methods and the research techniques. By this pyramid is shown that a research methodology is mainly defined by the research methods and the research techniques.

Figure 22: Research Triangle

**Research Paradigm** The research paradigm corresponds to how the researcher understand the basic real problem. In the seating plan problem this part of the issue does not exist. The global subject is to compare libraries which are really specific and does not need any interpretation. Similarly for the testing of the Play !. This step is more important when the researcher is different than the person who design the research subject.

**Research Methodologies** The research methodologies are means to guide the researcher in the good ways. Here the way is easy to find. For the libraries comparison: libraries just need to be compared. The research around the Play ! framework conduct to a test of this one. No more specific methodology are needed. The conclusion is quite the same than the precedent step.
**Research Methods**  The research methods are the different steps to conduct to the research. The two objectives will be implemented in order to test them.

1. For the testing of the libraries, first a solvable problem will be designed. After that a simple graphical user interface will be implemented. Then the test will begin by the installation. Afterwards, it will be possible to try to resolve the seating plan problem. Finally the efficiency test could be done.

2. For the *Play!* framework, the steps will be to install it, learn the specific language and then develop the application with it.

**Research Techniques**  The research techniques will be different depending on the objectives.

1. For the testing of the libraries, I will focus on the comparison. Lots of details can be compared: how it is easy to install, how is it easy to use, how many possibilities the library is offering, how efficient is the library, etc.

2. For the *Play!* framework, the research technique will be simply to use. During the use of this one, I will certainly find advantages and drawbacks compare to a basic HTML development. I will see as well if this framework provides enough features to implement my application.

![Research Design Figure](image)

Figure 23: Research Design Figure

As is mentionned by J. Jonker and B. Pennink (2010, p.25) the methodology is a part of the research design. The research design is how will be conduct finally the research. Two more thinkings are involved in the research design: the question context and the theory. Before to do some searches the researcher which is not a specialist in the domain of the studied area has his self thinking around how to conduct the research. His mind will change because, first the theory will improve his understanding and change his first plan. Secondly because context of the research question will interfere in how to answer to it. These three topics together create the research-design during the research.
6.1.2 Software Development Methodology

Consequently to the amount of development to do, take a sofware development methodology could help to earn time in the development phase. Software development methodology is today a recurrent subject around software development.

Referring to Dooley (2011, p.7) whatever the size of the project and how large is the team who work on it the cycle of the development methodology always contains heigh essential steps:

1. Conception
2. Modeling
3. Design
4. Coding
5. Testing
6. Release
7. Maintenance
8. Retirement

V Model & Waterfall Model  Two well known process are displayed below. The V model and the Waterfall model have a close architecture and are designed with the basic phases listed above. Both use backing up: that’s mean if a phase which does not work is discovered it is possible to go back to the previous phase. Both again use some verifications after each step to guarantee as much as possible to do not go back to a previous step. Because even if a return is imperative it is always a waste of time and money.

![Figure 24: Waterfall & V Models](image-url)
After that process some others emerged. One of the most important improvement is the loops. As is explained by Dooley (2011, p.12) use increments and iterations allows adapting code according to requirement change.

Because the understanding between clients and developers are not always perfect and because sometimes requirements change, it is important to anticipate that changes can happen in the development. One way to do is to establish an evolutionary prototyping process model J. Dooley (2011, p13).

Here is one example of this kind of process:

![Loop Software Development](image)

**Figure 25: Loop Software Development**

**Agile** One of the most famous software development methodology is Agile. From this methodology two sub-methodologies are well known (Scrum and Extreme Programming). These methodology are evolutionary, all the project will be not plan before the beginning. Particularities of Agile is that the client is involved in the project. In addition this method is characterised by a lot of release at each step of the project to be sure that the requirements are done correctly.
**Method Choice**  In our case the Agile methods are not useful because they are designed for team projects. Moreover, the research project is no going to works on a client - provider relation which is fundamental for Agile process.

Assuredly, the development cycle will be close to the V and the Waterfall models. Because I am alone to do this project the method is not as fundemantal as for a team work. However follow one of these architectures can ensure that tasks will be done in the good order.

To be structured in my development I will use the Test Driven Development (TDD) for the model Java part of the code. The Test Driven Development is a method which consists to first develop the test relative to a function an then to create the code until the test are passed. Accordingly to this method the JUnit test will be inserted to provide the test part of the project. These techniques are commonly used to provide a well-designed code done by some functions’ refactoring.
6.2 Requirements Analysis

6.2.1 Project Requirements

**Algorithms**  The two objectives around the algorithms are: show that is possible to compute a solution of the seating plan problem and model it as a CSP.

In order to provide an application usable by the user, the solution must be computed quickly. Compute the best solution for a small seating plan is quite easy. Yet, an user of the website could want to design a large one. The web application should work as well as possible according to the possibility offered by the libraries.

To produce a powerful application several libraries will be tested during the implementation. The computation time will define how many people can be inserted in the application and define the limits of the web application.

- **PR-1** Compute a solution for basic problem.
- **PR-2** Use some libraries to compute it.
- **PR-3** Define the limits of the web application.

**User Interface**  The other main objective is to design the application on the web. The web application is mainly oriented like a service provider. Accordingly the presentation of the website should be professional and follow some rules explained in the literature review.

- **PR-4** Designing an application where the user should login.
- **PR-5** Saving and loading his seating plan.
- **PR-6** Implementing a model to allow adding people and compute a seating plan.
- **PR-7** Working around a nice displaying in order to attract users.
- **PR-8** Developing the website with the framework *Play !*. 

\(^1\text{PR: Project Requirement}\)
6.2.2 Research Requirements

Thanks to the previous development several research objectives will be elucidate. They can be briefly resumed as follow:

- **RR-1** A feedback will be given around the interoperability between the seating plan problem and the implementation of a web site. The implementation will allow showing how it is possible to mix these two areas.

- **RR-2** The different libraries used will be compared to give their advantages and drawbacks accordingly to the current problem. Again, the development will permit to find out about these libraries and to compare them in some ways.

- **RR-3** An evaluation will be given around how works the *Play!* framework as well as the possibilities furnished. The comparison might be done with a basic HTML development.

- **RR-4** Finally, an appreciation will be awarded around how the libraries and the *Play!* framework work together.

6.2.3 Optional Requirements

The time to do the overall work is variable. Testing libraries could be long because of difficulties to model the problem or to understand how the libraries are designed. But in another hand that could be quick because the libraries may not model or solve the problem. Consequently to this variability optional requirements were defined.

A way to improve the application is to provide more and more features. These improvements can be used to test more the framework Play or to work more and more on the potential attractiveness of the application. Implementing a seating plan problem with different table shapes can be interesting. But this objective can be considerable. Adding table shapes can force to find a new approach of the problem and find a new algorithm to solve it. Then the used libraries could become not efficient.

**Outstanding requirement** One objective which can be interesting is about the *Play!* framework. This one is available in Java and in Scala. It could be interesting to work as well on the Scala. Doing this change requires to learn a new language. In addition, one of the CPS Java libraries named Cream has a Scala version named Copris. The testing of the Scala version of the framework with a specific Scala library could be really interesting to compare the difference with the Java development. Consequently an overview of the *Play!* framework could be done.

\(^2\)RR: Research Requirement
6.2.4 Project Risks

Referring to Raftery (2003, p.4) construction is always a risky job. These risks are due to the nature of construction. Each project is different that is why construction like design a website is always a risky work. It is impossible to predict the future. In the seating plan web application risks are really present.

First of all the main goals of the application are:
- Compare Java CSPs libraries
- Test the Java framework Play!

Both objectives are really dependent on something external of the project. The first objective can be completed only if libraries can answer to the seating plan problem and if algorithms implemented into are efficient on it. The second objective is also dependent on the framework Play!. If the development of this one is not enough advanced for the features required or if this one is really difficult to use the project will necessarily slow down. Another mystery point is the interoperability between these libraries and the Play! framework.

One other important point is the time. Only three months are allocated to develop the master project and to write the master dissertation. Accordingly, attention must be paid on the time used to develop the master project, to ensure that enough time are still remaining to write the following master dissertation.

One manner to thwart risks is to know that they are present and to take care of them. In addition, to be sure that the project and the dissertation will be hand in on time a project plan with a Gantt diagram was designed and followed.

To be sure that the master project is well conducted this one is monitored by a supervisor which knows the basic traps and can guide the dissertation to the right direction.
6.2.5 Application Main Functionalities

The goal of the application is to allow the user designing a seating plan. To do that, obviously the user should perform some actions explained in this part.

First of all the user should log into the web application. Because he can save his seating plan each session should be protected by credentials. Below is displayed the process to register and to login.

Figure 26: Use Case: User Register - Login

In addition the user can then accomplishes some tasks more evolved:

Figure 27: Use Case: User Other Tasks
6.2.6 Basic Design

In order to provide a first view of the application some sketches were drawed. They might be changed but they offer a global thinking about the features provided. The main page displayed below will mostly contain only four buttons which redirect the user to all the other pages:

- **New** which can be used to create a new seating plan.
- **Load** which allows loading an older seating plan.
- **People** which allows managing the different people inserted in the application.
- **Exit** to exit the application.

![Main Page Preview](image)

Figure 28: Main Page Preview

Other sketches are available in appendix. They show the basic design of the four pages described above. More details are available on these screenshots. However, it is possible that positions and details change a little during the implementation.
7 Project Development & Feedbacks

Thanks to the precedent parts the application was developed completely with the *Play* framework and CSPs libraries. In the next subparts are presented first how look likes the final application. Then a feedback around all elements tested is given.

7.1 Web Application Presentation

The first page of the application is composed by two blocks. The first one is designed to take user’s credential in order to enter in the application. Obviously, as is mentioned in the legal issues the password entered in the website is encrypted with a message digest *MD5*. In addition to be logged in it is imperative to check the checkbox with the rules and conditions.

![Login Page](image)

Figure 29: Login Page

As you can see above the login page requires only the user’s email and password. The button *Go* permits to be logged in and the button *Sign up* to go to the signup form below.

![Signup Page](image)

Figure 30: Signup Page

The sign up form is composed by the fields on the figure above. Some have restrictions such as minimal size, contains only letters, etc. The data entered is immediately checked when it is typed. These two blocks presented are switching together when buttons *Sign up* and *Sign in* are pressed.
7.1.1 Main Pages

The page "New" is available to create a new seating plan. A seating plan is characterized by a name, a size, a tag, a category and the algorithm to solve it. Then when the plan is created, new blocks appear. One to add people to the plan and one to add relations between people. Then the last possible action available is a button to launch the computation of the solution of the seating plan like it is presented below.

![Figure 31: New Page](image)

On that page it is possible to choose people only if there are already entered in the people part. That allows using easily the same people if there is lots of plans using same ones. There is no need to create again and again people. It is possible that the user add more people than the size allocated to the plan. In that case best participant are chosen. These ones are chosen thanks to there best skill and the category of the plan.

Afterwards, when more than two persons are added it is possible to add relation between them. Each relation is quantified by a number between zero and ten. In addition an optimisation number is added (between parentheses on the screenshot) according to persons' best skills. Finally, it is possible to press the Let's Go button to compute the seating plan with all data inserted.
The page "People" is available to add people in the application. It is possible further to use those people in a seating plan as is described above. People added are only available for the user logged in. In addition, you can add one major skill for each person added. That allows refining the seating plan. Thanks to that aberration will be avoided. For example it is not advised in a meeting to have two persons disruptive sitting close.

Finally another block is displayed with the statistics. That allows knowing how many person with each major skill are in a group. That can be useful to make an elaborate seating plan.

![Figure 32: People Page](image-url)
The page "Load"  The third page was designed in order to allow the user loading a seating plan. On that page all seating plans are listed. Nevertheless, two forms are available to search a specific seating plan. It is then possible to search by the name, the category and by the tag. A click on a seating plan of this page, load it as in the New page.

Figure 33: Load Page

The page "Personal"  is the page where it is possible to change the user’s password. The name and the email can’t be changed in order to guarantee the security.

Figure 34: Personal Page
7.1.2 Seating Plan Page

When the button *Let’s Go!* of the New page is clicked the seating plan is computed thanks to all data entered. The first use of the data is when the number of people inserted in the seating plan is higher than the plan’s size. Then some people need to be removed. The chosen people are designate depending to the plan category (creative, calm, information or festive) and thanks to people best skill. The high scores consequently to the following table are chosen:

<table>
<thead>
<tr>
<th></th>
<th>Creative</th>
<th>Calm</th>
<th>Information</th>
<th>Festive</th>
</tr>
</thead>
<tbody>
<tr>
<td>Talkative</td>
<td>0</td>
<td>-5</td>
<td>-4</td>
<td>0</td>
</tr>
<tr>
<td>Shy</td>
<td>-3</td>
<td>0</td>
<td>0</td>
<td>-4</td>
</tr>
<tr>
<td>Disruptive</td>
<td>-3</td>
<td>-5</td>
<td>-2</td>
<td>0</td>
</tr>
<tr>
<td>Thoughtful</td>
<td>-2</td>
<td>0</td>
<td>0</td>
<td>-5</td>
</tr>
<tr>
<td>Friendly</td>
<td>0</td>
<td>-1</td>
<td>0</td>
<td>-1</td>
</tr>
<tr>
<td>Open-Minded</td>
<td>0</td>
<td>0</td>
<td>-2</td>
<td>0</td>
</tr>
<tr>
<td>Creative</td>
<td>0</td>
<td>-1</td>
<td>-1</td>
<td>0</td>
</tr>
<tr>
<td>Ingenious</td>
<td>0</td>
<td>-1</td>
<td>-2</td>
<td>-4</td>
</tr>
<tr>
<td>Moderator</td>
<td>0</td>
<td>-1</td>
<td>-3</td>
<td>-3</td>
</tr>
<tr>
<td>Scattered</td>
<td>-2</td>
<td>-5</td>
<td>-4</td>
<td>0</td>
</tr>
<tr>
<td>Bored</td>
<td>-4</td>
<td>-1</td>
<td>-2</td>
<td>-5</td>
</tr>
<tr>
<td>Director</td>
<td>-3</td>
<td>0</td>
<td>0</td>
<td>-4</td>
</tr>
</tbody>
</table>

Figure 35: People Removing Table

In the case of the number of people inserted in the seating plan is equals to the size of the seating plan or under it, the plan is directly computed. The computation uses the relation entered by the user as well as an optimiisation consequently to best skill relations. The optimisation is following this tab:

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Talkative</td>
<td>-5</td>
<td>-4</td>
<td></td>
<td>-1</td>
<td>-1</td>
<td>-3</td>
<td>-2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Shy</td>
<td>-3</td>
<td></td>
<td>-2</td>
<td></td>
<td></td>
<td>-1</td>
<td>-3</td>
<td></td>
<td>-1</td>
</tr>
<tr>
<td>Disruptive</td>
<td>-5</td>
<td>-1</td>
<td></td>
<td>-3</td>
<td></td>
<td>-3</td>
<td>-1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Thoughtful</td>
<td></td>
<td>-4</td>
<td>-1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Friendly</td>
<td></td>
<td></td>
<td></td>
<td>-2</td>
<td>-1</td>
<td></td>
<td></td>
<td></td>
<td>-2</td>
</tr>
<tr>
<td>Moderator</td>
<td></td>
<td></td>
<td></td>
<td>-2</td>
<td>-1</td>
<td></td>
<td></td>
<td></td>
<td>-2</td>
</tr>
<tr>
<td>Scattered</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>-3</td>
<td>-3</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Bored</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>-3</td>
<td>-3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Director</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>-2</td>
</tr>
</tbody>
</table>

Figure 36: Relation Optimisation Table
Thanks to the tab above the real coefficient for each relation is established. Then another relation tab is designed and transformed to be suitable for a library in order to compute the seating plan (explained in a next part). The result page is as follow with a visual position of people:

![Seating Plan Page](image)

Only two things are available on this screen. The seating plan with ten persons and just above the explanation about this seating plan. The seating plan is never really saved. But all data to compute it is. Then to see again the result of the computation can be launched again easily. That allows making some changes if they are required.

Finally, for the client understanding a score is given to his seating plan. Then he can understand his result and compare with another one. Moreover, that can helps to improve it; the score can vary because of the number of persons included, the relations between them and their best skill chosen.

Because the seating plan is computed by a complex approached algorithm. Two same seating plan can be resolved by two different solutions. Nevertheless, these solutions will have a cost equivalent or almost equivalent.
7.2 Play Framework

One of the requirements was to test the java framework named Play!. In this part it will be detailed all elements tested around this framework. This framework was designed to develop web applications using java without the servlet system.

7.2.1 Installation & Creation

The first step of the development of a Play application is to download the framework. This one is easily downloadable on the Play website. After that a basic process is explained as well as to create a first application. One thing to point is that play contributors took care about play developers. Special commands were developed in order to transform a play project in a play project importable into several common integrated development environment. Two commands are available to transform the project in an Eclipse project and in an Intellij project. These two are really commons IDE to develop Java application.

In order to meet user requirements and habits, the play framework is manageable by different ways. It is possible to manage the different applications with a GUI interface available in a web browser. It is as well possible to create/delete/modify the application with shell commands.

The framework Play provides several possibility to link a database. It is possible to work with the in-memory database H2. It as well possible to work with other database such as phpmyadmin or oracle. The only requirements are to link the adapted java connector and to configure the database at a specified location.

```java
db.default.url="jdbc:mysql://localhost:8889/HereAreTheSeats"
db.default.driver="com.mysql.jdbc.Driver"
db.default.user=root
db.default.pass="****"
```

Figure 38: Database Configuration

With all of this, Play provides a nice environment for developers. The downloading and the installation were done in no more than three hours. These steps are mainly well documented in Play’s website.
7.2.2 Development Structure

The architecture of the Play framework is not intuitively understandable. To develop a web application with it some knowledges are mandatory. Nevertheless a web page is documented to explain how is formed a new project. To develop the front side of the application, the one visible by the user, the developer has to know how works HTML, CSS, JavaScript, etc. In another hand, the back side of the application is designed in Java.

The project is created with some folders, the most useful is apps this one is divided in three branches:
- models where the objects corresponding to the java object paradigm are defined.
- controllers where Java classes are developed to handle the actions performed.
- views this folder is composed by all HTML view presented to the user.

Another first ranked folder is public. It represents the resources publicly available in the project. In a website, the images, the cascade style sheets and the JavaScript code can be accessed.

A folder test is provided to allow developing some test around the model and the controller of the application. In addition, a log folder with log files is available. Finally, a folder conf is also basicly created with the files usable to configure the application.

This architecture could seem complicated because everything is already defined and it is composed by a lot of folders. However, the composition of the project is well done and the folders names are explicit. If the developer is experimented the architecture of the projet will not repel him. This one can be compared to other developments using Java such as the development of Android application. In both, public ressources are available in a different parts and the project is well structured.

One of the drawback of that mandatory architecture is that is impossible to change it. If names are not convenient for the developer or if he wants to add some libraries and needs to re-organize the project it is impossible. Play is using this architecture to perform some actions like generate a database (developed in a next part). Modifying the project will make it unusable.
7.2.3 Play Operation

In order to explain carefully why Play has very useful features it is important first to explain how Play works. First of all, the website appears when an HTTP request is done. On the local project that looks like that: http://localhost:9000/signin. A file named routes is called. This one contains all routes on the website and is formed as follow. For each available route the HTTP method, the action requested and the controller function are mentionned.

```
# Manage page
POST /signin controllers.Application.signin()
GET /signup controllers.Application.signup()
POST /exit controllers.Application.exit()
```

Figure 39: Route File

Accordingly when an action is requested the associate controller method is called thanks to the route file. At that moment the controller handles the request. The Java server side program is running. It is then possible to use the Java model, the access to the database, the Java libraries linked, etc. The last Java instruction is a call to an HTML view in order to present the next display according to the request.

```
<!-- View parameters-->
@((name: String)(mail: String))

<!-- HTML Code-->
<h1>Title</h1>
....
```

Figure 40: Play HTML Example

Finally, in the view above a new Http request is proposed. Then the cycle is running again until the user leave the website.

Figure 41: Play Operation
7.2.4  Features

**Database**  The Play framework is mainly designed to develop web application using java without the servlet system. The play framework contains some features in order to ease the development. It will be detailed in this part the majors features encountered.

As is mentioned in the installation part of the play framework, only few lines of configuration are necessary to link a database. After that, a database need to be installed if the use of a specific one is required. All next steps are handle by the play framework. The table’s creation is done in accordance with the java class designed in the model part of the application. Some special annotations are implemented in order to apply the agreed constraints.

```java
@Entity
public class SeatingPlan extends Model{
    private static final long serialVersionUID = -709144405378755964L;

    @Id
    private long id;

    @Required
    private String name;

    @Required
    private String category;

    @Required
    private String algorithm;

    @Required
    private String tag;

    @ManyToOne
    private UserApp userApp;
}
```

Figure 42: Database Creation

Each model class should extends to the Model class from the Play framework. That extension allows inheriting functions to save/delete/update the instance used in the database automatically. No knowledges are necessary to manage this database. It is not mandatory to know how works SQL even if an SQL database is used. The goal of the Play framework is to hide totally how works the database. Some other functions are designed to search data in the database and to look into the data entered.

```
List<SeatingPlan> seatingPlan = SeatingPlan.find.where().eq("user_app_id",getUserId()).findList();
```

Figure 43: Database Search
Webpages Links The framework Play is designed with several other features to ease the development. Two of them are mandatory thanks to the architecture of the project. All resources (JavaScript, CSS and images) must be defined in a specific folder. In the source code, these resources are linked directly with a specific path. Then, there are no ways to do not link correctly the resources.

![Figure 44: Resources Link Examples](image)

The other one is quite comparable to the feature above. The links between webpages are all in a specific file named *routes* in order to organize clearly the projects.

![Figure 45: Part of Route File](image)

As you can see above, the first column is the HTTP method (post or get), then the second column is the page wondered and finally the third is the Java method called to load the page.

Others Features The Play Framework works as well on the security of the application. The application uses a secret key encryption for many things: data exchange, sign session cookies, CSRF tokens, built-in encryption utilities, etc.

In addition the Play Framework is based on the Model-View-Controller pattern which works to design correctly an application. This pattern contributes to improve the possible future adaptation of the code and to separate the code correctly.

To conclude, some other tools such as JUnit which is a tool to test the Java part of the application are embedded in the Play package. This one was improved in order to test the HTML view too. Then JUnit can be used to test the whole application. These features provided with Play are quite often used. The adding of these ones in the package is a very good thing. It is less work for the developers. Firstly because the interoperability is ensured, secondly because there is less tools to install.
7.2.5 Java & Web Languages Interoperability

In order to give a performing feedback around the play framework, the interoperability between the java server part and the client web part were tested. The link between them is working correctly.

JavaScript and DOM web technologies were tested as well. Both work very fine, no problems were reported during the different use of them.

The AJAX technology was tested. The process done was exactly the same as a web form request with Play. A post request can be launched with parameters via AJAX. The framework Play handles the request, find the Java function corresponding to the request in the routes files. Finally, it is possible to collect HTML code available in a view called by the controller Java function. Ensure that AJAX works is very important to develop dynamic websites. All tests done passed successfully. The interactions on the website seem to be not altered at all. The navigation between pages is very quick.

For the user of the application the Java part is nonexistent. The website works exactly like a basic website using HTML and Php. In the view part, all elements developed in the new standard HTML5 are usable. The new CSS3 works as well efficiently. For example, lots of transitions were tested.

```css
.submit_search {
    font-size: 16px;
    transition: all 1s linear;
}
```

Figure 46: CSS3 Transition

As it shown below it is possible with HTML5 and CSS3 to design very nice graphical user interface. Accordingly, most of the rules explained in the literature review are applicable to a website designed with the Play framework.

Figure 47: CSS3 Design
7.2.6 Documentation & Examples

The documentation provided by Play framework has some advantages and some drawbacks. The first steps with the Play framework such as the installation and the creation of a project are well explained. A video is available easily on the website where is detailed all basic tasks used to developed something with Play.

A step further, everything is as well explained meticulously, how the interoperability is guaranteed, how works the redirections or how to install a basic database. Again the Play framework team worked incredibly.

Nevertheless, when the next step further is reached, the progression is more difficult. More advanced tutorials are provided. However, the provided examples are huge, it is then complicated to understand a project very large at the beginning. Too much advanced features are used in the same time without enough explications. More steps are missing between the two ranges of help.

Another problem of the Play Framework is the number of version. In addition the changes between the versions are consequent. It is then complicated to find out a solution of an encountered problem which is corresponding to the correct version.

<table>
<thead>
<tr>
<th>Version</th>
<th>Released Date</th>
<th>Version</th>
<th>Released Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>Play 2.3.2</td>
<td>21 July 2014</td>
<td>Play 2.1.4</td>
<td>11 September 2013</td>
</tr>
<tr>
<td>Play 2.3.1</td>
<td>25 June 2014</td>
<td>Play 2.1.3</td>
<td>06 August 2013</td>
</tr>
<tr>
<td>Play 2.3.0</td>
<td>30 May 2014</td>
<td>Play 2.1.2</td>
<td>5 July 2013</td>
</tr>
<tr>
<td>Play 2.2.4</td>
<td>21 July 2014</td>
<td>Play 2.1.1</td>
<td>13 April 2013</td>
</tr>
<tr>
<td>Play 2.2.3</td>
<td>1 May 2014</td>
<td>Play 2.1.0</td>
<td>6 February 2013</td>
</tr>
<tr>
<td>Play 2.2.2</td>
<td>1 March 2014</td>
<td>Play 2.0.8</td>
<td>20 September 2013</td>
</tr>
<tr>
<td>Play 2.2.1</td>
<td>31 October 2013</td>
<td>Play 2.0.7</td>
<td>11 September 2013</td>
</tr>
<tr>
<td>Play 2.2.0</td>
<td>20 September 2013</td>
<td>Play 2.0.6</td>
<td>06 August 2013</td>
</tr>
<tr>
<td>Play 2.1.5</td>
<td>20 September 2013</td>
<td>Play 2.0.5</td>
<td>01 August 2013</td>
</tr>
</tbody>
</table>

Figure 48: Play - Release Table

Finally, the most serious Play framework’s weakness is the youthfulness. It is very difficult to find solution to advanced problem on the internet. Even if the Play framework is more and more known. There is not a lot of people developing with it. Accordingly, there is not a lot of forums dedicated to this technology. Consequently, it is difficult to find problems solutions and help. This problem is amplified by the number of change between the first and the last version. Because development change, problems change and each new changement reduces the number of people able to know how to fix it.
7.2.7 Play Conclusion

The Play framework is a very powerful system to develop websites or web applications. Its best advantage is to include Java in the server part of the framework. By the way, some libraries are already included and some others can be added.

Combine the web technologies with Java without using servlet is really powerful. In addition, this development solution is perfectly adapted to solve the seating plan problem on a website application. Furthermore, the Play framework seems to work easily on different web browsers. Some of them were tested (Safari, Google Chrome and Mozilla Firefox) with good result.

The tab below summarize roughly how the Play framework was perceived during the application’s development:

<table>
<thead>
<tr>
<th>Subject</th>
<th>Feedback</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Installation &amp; Project Creation</td>
<td>9/10</td>
<td>Took around one hour.</td>
</tr>
<tr>
<td>Features included</td>
<td>8/10</td>
<td>Database, Security, MVC Pattern, etc.</td>
</tr>
<tr>
<td>Powerfulness</td>
<td>8/10</td>
<td>Good Rapidity.</td>
</tr>
<tr>
<td>Complexity</td>
<td>5/10</td>
<td>Framework reserved to professional</td>
</tr>
<tr>
<td>Documentation &amp; Example</td>
<td>5/10</td>
<td>Could be enriched</td>
</tr>
<tr>
<td>Web Community</td>
<td>1/10</td>
<td>Almost non-existent</td>
</tr>
<tr>
<td>Average</td>
<td>6/10</td>
<td>-</td>
</tr>
</tbody>
</table>

Figure 49: Play - Overall Feedback

The table above shows how is positioned this framework on the market of web technologies. This can become a real problem for the others technologies because this one is really easy to install and the first steps with it are easy. The features included thanks to Java ease the use of the developer without reducing the navigation’s power and rapidity.

Play proposed a new technologie. They should teach as clearly as possible how this one works. An API is available as well as basic examples. The main limitation is the small size of the Play framework community. This is not easily solvable. Play should be more famous to attract developer to use it. Complex things are not always explained clearly and sometimes unexpected problems appears. These problems are usually handled and solved by the community which is non-existent.

Finally another drawback is the difficulty of the development. To develop an application with the Play framework, the developer must know how work Java and web technologies (HTML, JavaScript, CSS, etc). Consequently, this one is mainly reserved to computer scientist and professional with developing code.
7.3 Libraries

7.3.1 Introduction

In this part of the report is explained the research done around the Java libraries able to solve the CSP (Constraint Satisfaction Problem). The research has been mainly oriented around Travelling Salesman Problem which allows solving the Seating Plan Problem needed for the web application.

In the literature review seven libraries were found to solve CSP: Cassowary, Choco, Cream, Google CP, JaCop, JOp and Opta Planner. Two of them have been quickly eliminate of the research:

- The first one is Cassowary because the last release is in 2000 and the sourceforge folder containing the project is now empty. The project is clearly abandoned and cannot be retained for the application.

- The second one is JOpt. This library is still available. Nevertheless, it is impossible to model graphs with it. This one is a little project designed to solve more simple constraint satisfaction problem. It is then irrelevant.

For all other libraries some tests performed are explained in the next part. Two main tests are performed: the TSP and the Queens Problems. It was chosen to test libraries with another problems for two reasons:

- Some libraries cannot be tested on the travelling salesman problem because it is impossible for them to compute it. Another problem should be introduced to compare the libraries’ performance.

- Even if libraries are able to solve the seating plan problem the testing will be more accurate if two different performances are measured. In order to be again more complete the comparison will detail some advantages and drawbacks in the nexts parts.
### 7.3.2 Travelling Salesman Problem Research

To solve the problem used in web application the research was oriented first around the travelling salesman problem. To solve this problem only three libraries get tools to solve it. These ones are: Choco, Google CP Solver and JaCop. You can see below the time to get the result of a problem containing four towns. JaCop and Google CP Solver give quickly the result of the problem. Conversely Choco is four times longer.

<table>
<thead>
<tr>
<th>Choco Solver</th>
<th>Google CP Solver</th>
<th>JaCop Solver</th>
</tr>
</thead>
<tbody>
<tr>
<td>28ms</td>
<td>8ms</td>
<td>8ms</td>
</tr>
<tr>
<td>24ms</td>
<td>13ms</td>
<td>11ms</td>
</tr>
<tr>
<td>49ms</td>
<td>9ms</td>
<td>9ms</td>
</tr>
<tr>
<td>42ms</td>
<td>10ms</td>
<td>9ms</td>
</tr>
<tr>
<td>40ms</td>
<td>9ms</td>
<td>8ms</td>
</tr>
<tr>
<td><strong>36.6ms</strong></td>
<td><strong>9.8ms</strong></td>
<td><strong>9ms</strong></td>
</tr>
</tbody>
</table>

Figure 50: TSP Test - Size 4

On the next tab, the result are done with a travelling salesman problem containing fifteen towns. The times are longer than in the first test and that is normal. More the number of town grow up more the time grow consequently. The ranking between libraries change radically on that step. The new quickest library is Google CP Solver.

<table>
<thead>
<tr>
<th>Choco Solver</th>
<th>Google CP Solver</th>
<th>JaCop Solver</th>
</tr>
</thead>
<tbody>
<tr>
<td>265ms</td>
<td>29ms</td>
<td>124ms</td>
</tr>
<tr>
<td>212ms</td>
<td>40ms</td>
<td>229ms</td>
</tr>
<tr>
<td>277ms</td>
<td>36ms</td>
<td>334ms</td>
</tr>
<tr>
<td>194ms</td>
<td>34ms</td>
<td>293ms</td>
</tr>
<tr>
<td>210ms</td>
<td>32ms</td>
<td>139ms</td>
</tr>
<tr>
<td><strong>231.6ms</strong></td>
<td><strong>34.2ms</strong></td>
<td><strong>223.8ms</strong></td>
</tr>
</tbody>
</table>

Figure 51: TSP Test - Size 15

On the final step of the test this one is done with thirty town. The Choco and Google CP libraries still grow up gradually. But, the Jacop solver time raised up incredibly.

<table>
<thead>
<tr>
<th>Choco Solver</th>
<th>Google CP Solver</th>
<th>JaCop Solver</th>
</tr>
</thead>
<tbody>
<tr>
<td>480ms</td>
<td>80ms</td>
<td>5079ms</td>
</tr>
<tr>
<td>442ms</td>
<td>224ms</td>
<td>≈ 1min</td>
</tr>
<tr>
<td>505ms</td>
<td>220ms</td>
<td>≈ 0.5min</td>
</tr>
<tr>
<td>1456ms</td>
<td>362ms</td>
<td>≈ 4min</td>
</tr>
<tr>
<td>667ms</td>
<td>134ms</td>
<td>≈ 4min</td>
</tr>
<tr>
<td><strong>710ms</strong></td>
<td><strong>204ms</strong></td>
<td><strong>121288ms</strong></td>
</tr>
</tbody>
</table>

Figure 52: TSP Test - Size 30
As is resumed below the Google CP solver is the best library to compute the TSP. Even if the JaCop solver is effective with little value this one grow up too quickly when the number of town raises. On the graph the vertical axis represents the time in millisecond and the horizontal axis represent the number of town in the TSP.

![TSP Test Graph](image)

The library OptaPlanner is different than the others because this one is a full Java package containing a graphical user interface. This one is able to solve the problem proposed above. Nevertheless, it is impossible to compare this one with the others. The library is obviously longer than the others because this one has to load the data, compute the solution and then display the result. This solution can be useful but is not entering in the kind of library required.

It is important to understand that the result above presents time elapsed during a test on one computer. The test is relative to the computer and can be totally different with another computer. For example, some libraries are using multi-threading to enhance the speed of the program. Consequently, results can really differ according to the environnement. However, the test allows presenting one view of how work these libraries. The full test code is on a CD nevertheless a part is available in appendix of this report.
7.3.3 NQueens Test Ranking

In order to give a truly feedback around the libraries these ones were tested around another problem which is the N-Queens problem. This problem needs a chessboard of size N. To solve this problem you have to put on the chessboard N queens and no queens must be able to attack another one. The queens can attack with the rules of chess e.g. they can attack horizontally, vertically and diagonally. Here is one solution of the eight queens problem:

![Chess - Eight Queens Problem](image)

Depending to the number of queens it can exist one or more solution. The tests done on the next page represent the time elapsed to find one solution of the NQueens problem. Two libraries allow finding any existing solutions: Cream and Google CP. The test include as well the libraries Choco, JaCop, Opta Planner. Again the result of Opta Planner are not really relevant because of the graphic display included and because only some values can be tested. In addition, the time result is produced by the program itself and cannot be verified.
<table>
<thead>
<tr>
<th>Queens</th>
<th>Choco</th>
<th>Cream</th>
<th>Google CP</th>
<th>JaCop</th>
<th>Opta Planner</th>
</tr>
</thead>
<tbody>
<tr>
<td>8</td>
<td>60 ms</td>
<td>29 ms</td>
<td>1 ms</td>
<td>12 ms</td>
<td>190 ms</td>
</tr>
<tr>
<td>16</td>
<td>49 ms</td>
<td>117 ms</td>
<td>2 ms</td>
<td>50 ms</td>
<td>810 ms</td>
</tr>
<tr>
<td>32</td>
<td>80 ms</td>
<td>163 ms</td>
<td>3 ms</td>
<td>90 ms</td>
<td>1422 ms</td>
</tr>
<tr>
<td>64</td>
<td>404 ms</td>
<td>218 ms</td>
<td>16 ms</td>
<td>126 ms</td>
<td>2078 ms</td>
</tr>
<tr>
<td>128</td>
<td>Out Of Time</td>
<td>428 ms</td>
<td>20 ms</td>
<td>420 ms</td>
<td>Inexistent</td>
</tr>
<tr>
<td>256</td>
<td>-</td>
<td>Out Of Time</td>
<td>41 ms</td>
<td>2186 ms</td>
<td>38813 ms</td>
</tr>
<tr>
<td>512</td>
<td>-</td>
<td>-</td>
<td>222 ms</td>
<td>10003 ms</td>
<td>Inexistent</td>
</tr>
<tr>
<td>1024</td>
<td>-</td>
<td>-</td>
<td>788 ms</td>
<td>Out Of Mem.</td>
<td>-</td>
</tr>
<tr>
<td>2048</td>
<td>-</td>
<td>-</td>
<td>Out Of Mem.</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

Figure 55: Times Table - Eight Queens Problem

On that test Google CP Solver is again strongly efficient to solve the NQueens problem. Whichever the library against it, Google CP is ten time quickly. Moreover, it is the one which permits to compute the highest problem. The libraries are declared Out Of Time when after five minutes the result is not found yet. The label Out Of Memory is given when an error is raised because of the memory. It is certainly in relation with the computer used. Nevertheless, that allow showing that this error happens more quickly with the JaCop library than with Google CP.

![Times Graph](image_url)

Figure 56: Times Graph - Eight Queens Problem

It is possible to see on the graph above that some libraries have strange behaviour such as Cream solver. It has very good result until 128 queens. Then the result is really out of range (around 500 000 milliseconds). It is the same for the Choco solver, results are good until the problem with 64 queens. After that step the time raised exponentially. The full test code is on a CD nevertheless a part is available in appendix of this report.
7.3.4 Documentation & Examples

Even if a library is sometimes performant on a problem, this one should be developed conscientiously. One thing important is how is documented and how many samples of the library are available. More the library is documented more it is possible to understand quickly how to use it.

**Choco** is a quite large solver. In addition of the program more than one hundred samples were developed. These ones are mainly commented. Ones are in relation with common problems such as graph coloring or timetabling. Others are written to show how works the library. A full documentation is available as well as the whole code on the repository hosting service github. The understanding of the library was a little difficult. However the owners answer in a day when an e-mail was sent to them. That rapidity is a real force.

**Cream** is more small than the previous library. Accordingly only twenty samples are proposed. This one is easier to understand thanks to it size. Nevertheless as is mentioned in previous parts there is nothing developed to work with graphs. No answer was received on the asked help by email.

**Google CP** is a very huge library able to solve many problems including the travelling salesman problem. This one is easy to understand and it is really easy to create a first program with it. The library include only fifty examples to show how to use the library. Yet, there is an huge documentation, a forum dedicated to the library as well as some blogs. The after-sales service is really quick to answer to questions asked by email.

**JaCoP** is again a quite huge project with more than two hundred examples proposed. A full user guide was edited. In addition the developer took care to create a library easily understandable. Like all others libraries an API is available.

**Opta Planner** is more than a library. It is a full package which contains a graphical user interface means that there is example to use the library attached. Moreover, this application is easily usable to solve common problems. These ones works with data inserted via the XML standard. Some XML examples are provided.
7.3.5 Libraries Conclusion

To conclude, the research around these libraries were interesting. Two were definitively out of the subject. One Cassowary because this one is not still updated. The other because the library works only on basic constraint satisfaction problems.

The Opta Planner solver is an application and not a library. Nevertheless, it can solve the problem required. This application is well conceived and is effective. But, it will be not possible to integrate it to the Play Framework to develop the web application to solve the seating plan problem.

Here is the summary tab about other libraries:

<table>
<thead>
<tr>
<th>Test</th>
<th>Choco</th>
<th>Cream</th>
<th>Google CP</th>
<th>JaCop</th>
</tr>
</thead>
<tbody>
<tr>
<td>TSP</td>
<td>++</td>
<td>Non Tested</td>
<td>+++</td>
<td>+</td>
</tr>
<tr>
<td>NQueens</td>
<td>++</td>
<td>++</td>
<td>+++</td>
<td>+++</td>
</tr>
<tr>
<td>Understanding</td>
<td>-</td>
<td>+++</td>
<td>++</td>
<td>+++</td>
</tr>
<tr>
<td>Documentation</td>
<td>+++</td>
<td>++</td>
<td>+++</td>
<td>+++</td>
</tr>
<tr>
<td>Examples</td>
<td>+++</td>
<td>+</td>
<td>++</td>
<td>+++</td>
</tr>
</tbody>
</table>

Figure 57: Summary Table - Libraries

It is impossible to establish an immobile ranking of these libraries. Each library has his strengths and weaknesses. Google CP Solver seems to be the most efficient. Yet if an user is not looking for performance, a library such as Choco which works really well can be suitable. On the case of the seating plan problem which can be modelised as a travelling salesman problem the Google CP Solver library should be the most convenient.
7.4 CSP libraries & Play Framework

7.4.1 Choco, JaCoP and Google CP

Choco, JaCoP and Google CP are the three libraries able to solve the travelling salesman problem. In order, to solve the problem of the application which is the seating plan problem it was important to understand how the TSP works. The three libraries need some steps with their special syntax to model the problem, configure the solver and then compute the solution.

The first step was to link the libraries to the application. Already the first problem happens. The three libraries used a jar archive to be linked. Hopefully, Play is handling that kind of archive. The only thing to do is to add these libraries to the folder named $lib$. Then, Play automatically link the libraries to the application.

However, the Google CP solver needs another library to work. This one should be absolutely in the main folder of the application. But Play is designed with an immobile architecture where libraries are not in the main folder. It moves files by itself when it is running the application. Accordingly nothing was planned to look after that kind of files which are not jar files. Then it was impossible to link the Google CP solver. It is impossible to use it with Play even if this one show really good performance tests. The interoperability is not ensured.

Then the two last libraries work almost similarly. After some initialization steps a data matrix is expected with the distance values. Here is an example of cost matrix with four towns for the travelling salesman problem:

\[
\begin{pmatrix}
1000 & 8 & 5 & 1 \\
12 & 1000 & 12 & 2 \\
7 & 3 & 1000 & 5 \\
1 & 3 & 6 & 1000 \\
\end{pmatrix}
\]

Figure 58: TSP Cost Matrix

The red eight represents the distance to go from the town one to the town two. Accordingly, the blue six represents the distance to go from the town four to the town three. Here the matrix is not symmetric that means that the distance between two towns is depending to the direction. The diagonal is filled with the number one thousand in order to avoid that the algorithm makes loops. A loop is when from a town the algorithm can go to the same town. Here the problem is to minimize the travel which goes through all towns. Then, loops need to be avoided.
The seating plan problem can be resolved quite in the same way. The goal is to maximize the good relationships between participants. Then, we can minimize the bad relationships between participants to resolve the problem. Accordingly, a matrix to solve the seating plan problem need to satisfy some conditions. The first one is that best relationships must be modelized by small numbers and worst relationships by high numbers. The second ones is that the matrix should be symetric because the relation between two people view by a third person is the same in both way. Finally, the diagonal should stay with huge number because there is no sense to modelise a relation between a person and himself.

Then the matrix should look like that:

$$\begin{pmatrix}
1000 & 8 & 5 & 1 \\
8 & 1000 & 3 & 4 \\
5 & 3 & 1000 & 9 \\
1 & 4 & 9 & 1000
\end{pmatrix}$$

Figure 59: SPP Cost Matrix

Again both libraries work quite similarly, the solver find the solution and then store it in an tab, the index of the tab represents a person and the value in the tab, the next solution node:

<table>
<thead>
<tr>
<th>index</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>value</td>
<td>3</td>
<td>2</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

Figure 60: SPP Solution Tab

Accordingly to the tab, it is quite simple to rebuild a seating table with the solution:

Figure 61: SPP Graphs

The interoperability between the Play framework and Choco and JaCoP is good. The solver allows finding the solution and to produce a result easily available to create a seating plan. Nevertheless the performance is not as good as during the tests. To obtain a result in less than few seconds the seating plan should not include more than sixteen people.
7.4.2 Interoperability Conclusion

To conclude, only two libraries are efficient around the encountered problem and usable with the framework Play. These two are Choco and JaCoP. However, the results presented are not as efficient as it was concluded during the tests. Reasonably only sixteen people can be inserted in a plan.

The Play framework offer the possibility to add some Java libraries. This is the main advantage of this framework: the connection between the Java and the web languages. This framework was built to use the universality and the powerfulness of Java in order to develop websites. It is really easy to add a new library if the library is a jar or sbt archive. Nevertheless, the library required for the Google CP Solver is quite specific and then unusable.

Most of libraries can be included. In addition Play framework owners are still working on the project. Then the higgest problems encountered by developers will be certainly fixed in the future.

7.5 Design Research Development

7.5.1 Design Research Introduction

The web application created is mainly composed by five webpages which were detailed earlier. This pages were conscientiously developed in order to satisfy the operationnal requirements and to satisfy the condition of a good web design.

As is explained in the literature review part, the goal of the different rules around the web design is to ease the use. Ease and speed are really expected on the web. In addition have a nice look is as well important. It will be expounded all web design principles used to develop the application.

Another part relates the result of a questionnary. The test was to show the website to several users and then let them fill the survey. To be sure that the developed website is well organised and efficient it can be interesting to collect what people think about. Accordingly a questionnaire was done and the result of it is summarized in the next part.
7.5.2 Web Design Principles Followed

In this part is explained how the website was designed. Thanks to the literature review some principles were highlighted around how is organized the website. Some rules exist to do it but it is sometimes difficult to follow them. Here is explained how is handled this problem.

Global Architecture Some principles are in relation with the global architecture of a website. It is recognized that the home page of an application should introduce the content available in it. Accordingly, the application home page is designed as follow:

![Home Page](Image)

The middle of the page is taken by the log in form. Then, everywhere else are presented the actions which can be performed on the website. That allows the user answering basic questions such as *What is this website?*, *What can I do on it?* and *What is the content?*. In accordance with the Three-Click rules, each part of the application is reachable in at most three clicks. Nevertheless, nothing is available without to be logged in on the website.

About the last common question: *Why I chose this one and not another one?* two answers can be presented. Firstly, the application cover a specific subject. Then on the web market there is not a lot of concurrent. Secondly, a work was done around the nice displaying of the application in order to show how performing this website is. That permits to let feel confident the user about using it.
**Billboard Design**  The next step is the billboard design. When the user enter in the application many things again could be done to let feel confident the user. As usual the menu is positionned on the top left of the website. In addition, to be easily understandable the application is only composed by four pages corresponding to the menu’s link. During the navigation the user can look at the menu to know where he is in the website. The current page link is colored in blue.

As you can see in the previous page the website was cleaned as much as possible. The goal was to let only the useful information. Moreover, redundant information has been avoided to expound a website as simple as possible.

It was related in the litterature review that the scrolling is commonly used but sometimes that can hide information. Accordingly the website was organized to avoid totally the scrolling. Obviously, if the user is using a very small screen he will need to scroll. Otherwise he normally does not need it.

Specialists used to express that web users are not looking everywhere on the page. They are commonly only looking at the higher titles and typically on the top-left of pages. The website conceived was really refined in order to ensure that everything can be catch quickly. But no special attention were done around where is placed the blocks in the page because all blocks need the same visibility.

**Element Design**  The design of the element is really important for the attractiveness but as well for the understandability. Then, some rules where followed. The first ones are around the navigation. Each button should be obviously clickable by the user, that is why buttons design stayed simple. Then to be constant all buttons have the same graphic design.

Consequently, the entire application follows these rules, for example the same font policy is used everywhere. The text terminologies are conscientiously chosen to make self-evident to avoid misunderstanding.

Another point to highlight is the customer time satisfaction. An evident point is that a web user does not like to wait. It was mentionned that the Play framework efficiency was excellent. That is a good point. To ensure that the customer of this website will be satisfied, it is impossible to compute a seating plan containing more than sixteen people. Consequently, the application works is always time reasonable.
7.5.3 Questionnaire Answers

Here is a questionnaire presented to inquire about the website:

---

**Questionnaire**

Web Application: Here Are The Seats

Owner: Paul OUVRY

Name: .................................
Profession: ...............................  

---

**Question 1:** How do you feel about the login page?

- [ ] Interesting
- [ ] Useful
- [ ] Nothing to say
- [ ] Unnecessary

**Question 2:** Does it look easy to use?

- [ ] Yes
- [ ] Maybe
- [ ] No

**Question 3:** After few minute of use, what is your feeling then?

- [ ] No problem
- [ ] Don’t understand one part
- [ ] That’s seem strange
- [ ] Understandable

**Question 4:** What do you think about the design?

- [ ] Amazing
- [ ] Quite Good
- [ ] Nothing to say
- [ ] Horrible

**Question 5:** Do you think that website can be useful for you?

- [ ] Yes
- [ ] No

**Question 6:** If yes, how often do you think you will use it?

- [ ] Once a day
- [ ] Once a week
- [ ] Once a month
- [ ] Once a year

**Question 7:** The application can compute seating plan until sixteen people.  
Do you think it is enough?

- [ ] Yes
- [ ] No

---

Something to add:

.............................................................................................................................................................................
In order to know clearly how is viewed the web application by user a questionnaire was elaborated. This one was presented to more than twenty people, mostly student from several countries. Even if, the number of answer is small it is interesting to conclude what these people think about it. Here is summarized the answers of each question:

1. The feedback around the login page is pretty good. Yet, one problem was underlined: people are disappointed because they have to log in the application before to test it. It is then a long process before to use it. Some solutions to fix the problem could be presented such as log in only when the user wants to save his plan to enhance the project.

2. People answered mainly yes to this question. It seems to be easy which is a good point.

3. Most people think the application is easy to use. But they think also that to compute a seating plan the process is long to do. There is no way to shorten this process. To compute a seating plan people and relation should be added.

4. The design is as well mainly appreciate. One or two people have noted that the colour of the website are a little sad because of the intensive use of grey and black.

5. Some people have some difficulties to answer to this question. In summary, they approve the concept but there is not a lot of situation where this one is inevitably useful. In addition is was mentionned that in that case where they have a seating plan to optimize, they never think to do it on internet.

6. Approximatively ninety percent of the answer are once a month or once a year. That answer summarize clearly the answer to the question number five.

7. The response to this last question is motley. The most little part thought about the professional use of the application. Then plan a meeting with sixteen persons could be enough. The other part thought about the personal events such as birthday and weddings. In that case, sixteen people were definitively not enough.

7.5.4 Web Design Conclusion

In conclusion some work was done around the design of the application in order to improve the efficiency and the attractiveness. The questionnaire allows evaluating roughly the work done. This one is mainly approuved even if some limitations appear. Web designer is real job which requires lot of knowledges and experience. However, this website is good enough and could find his place on internet.
8 Requirements Satisfaction Analysis

This part is a summary relating the different requirements fail and success. This part will detail all requirements proposed at the beginning around the project, the research and the optionnals requirements.

8.1 Project Requirements

Algorithms There were two objectives around the developed project. Requirements were established around the algorithms.

- PR-1 It was proved during the literature review and during the implementation that it was possible to design an application in order to solve the seating plan problem.
- PR-2 Then the use of some libraries was introduced to solve it as quickly as possible.
- PR-3 The requirements which says that libraries will define the limits of the web application is accomplished: the website is solving easily the problem while the number of people is under sixteen.

User Interface The other technical requirement was to design an application to use easily the libraries on the web.

- PR-4 The first user interface requirement done was to design a login page where the user can login.
- PR-5 The next one was to let the user save and load his seating plan. This one is as well implemented.
- PR-6 Another one was to allow the user adding people and compute a seating plan. It is possible to add people. It is possible to add some in the People page of the application and to add them to a seating plan in the New page. Then it is possible to add relation and to compute the result of the seating plan like it was required.
- PR-7 The next point was quite difficult to implement: work around a nice displaying in order to attract user. Satisfying this point was quite long because the web design is quite complicated. Some action were needed to format all HTML elements in the HTML page. The design was mainly made with the CSS language, in addition some image were designed and the font policy changed. Even if nice displaying is something quite subjective it was decided to work on it and try to valorize application’s functionalities. Some dynamic navigations process were deployed in JavaScript, AJAX and DOM to approach the design of mobile applications which are really famous nowadays.
- PR-8 Finally the last main requirement about the IT project succeed, it was to develop the whole application with the specific framework Play.
8.2 Research Requirements

Thanks to the previous application development several research were accomplished in parallel. Several research parts were to explore such as how works the CSPs libraries, the framework Play! and the how they can work together. In the requirement analysis they were expressed as follow:

- **RR-1** The first research requirements was to test the interoperability between the seating plan problem and his modelisation on a website. It was showed that the seating plan thanks to Java libraries is computable. In addition the interoperability between the libraries and the Play! framework were excellent. Consequently, the mix between these two areas work well. The website implemented is a real proof about how these two concepts can work together.

- **RR-2** Another requirement was to compare the libraries. Consequently, some tests were done on the libraries in order to emit an opinion on how works these libraries. Both conclusion were provided: a statement around the efficiency of these libraries and a statement on which ones are suitable to the application.

- **RR-3** In addition a complete feedback was expressed on the framework Play!. Some research were carried out to expound how is efficient this framework. The global statement relates that this one is really valuable. It seems to be as efficient as a basic AMP (Apache, MySQL and Php) development. Moreover, this one bring the Java back end which allows working with some included features.

- **RR-4** The last research requirement was to rule on the interoperability between Play! and the libraries. Even if the Play! is working well and the libraries too, if one is not working with the other the whole project can be compromised. Hopefully, the research and the project demonstrate that both are working well together.

8.3 Optional Requirements

The IT project and the research can be time variable depending mainly on the libraries. If some libraries are difficult to link with the project that can lengthen the time of the work required. The project was done approximatively on time thanks to the project planning. Nevertheless some work were improved such as the design of the application. Because some libraries were unable to compute the seating plan problem, another problem have been tested. The N-Queens allows knowing more about the libraries and to be more accurate on the comparison. Graphic new elements such as displayed statistics were developed.

One outstanding optionnal requirement was to develop the application in Scala and compare with the Java version. Yet, the CSP library available in Scala and in Java was not able to solve the seating plan problem. Then it was impossible to develop this one.
8.4 Difficulties Encountered

In that project some difficulties were encountered. Those problems were mainly technical ones. The project planning was conscientiously designed and leads the development in the good way. As it is explained in the background part, construction is always a risky job. Unexpected errors or obstacles can happen and should be overcome.

The first main problem was to understand how to exchange objects with the Play framework. This framework is using Java and HTML to develop web applications as it was clearly explained. Yet this two languages have some differences. The Java is a oriented object programming language. Then at the beginning it was difficult to understand how to use these objects in HTLM which is not using the object paradigm. Two solutions were finally found: use a JSON format or parse string in the JavaScript in order to simulate objects.

Another point who takes time was to configure the database with the Play framework. It was chosen to use a database in PhpMyAdmin to implement the project. Nevertheless, the basic Play’s database is H2. Then to install another one, some configuration were required. That takes some times to find the correct configuration and the correct process to link the data with the model. However, after that step, everything is designed to ease the use as much as possible.

Finally, the development which delayed the most the progress is the understanding of the libraries. Each library has its own working process. Functions names and attributes are not the same. In addition, some functions are not easily understandable. The project slow down sometimes during one or two days. Hopefully, the help from the libraries developers was extremely valuable and some were able to answer quickly. Then the project still go ahead. Moreover, because several libraries needed to be tested, it was possible to switch to another if the development was blocked on the current one.
9 Professional, Legal, Social and Ethical Issues

9.1 Professional Issues

The application developed in parallel of this report is quite small relatively to a professional project implemented by a professional team. In order to develop an application as close as possible to the professional usages I followed a specific development method as is explained in another section.

The application is guided by the Code of Good Practice and the Code of Conduct established by the British Computer Society.

In addition, to build a professional web application some indispensable work was done. First, the code was well commented to promote the understanding and the re-use of this one. Secondly thanks to development methods the application was tested as much as possible. Professionally, distribute a functional application is imperative.

Finally, a special attention was paid to write a support documentation for the user and for the developer.

9.2 Legal Issues

As is explained by A Tysver (n.d) the development of a website always introduce several legal issues. The most restrictive legal issues are about the distance selling and e-commerce website. The developed web application is not selling goods or services. By the way most of the legals issues are not relevant. Nevertheless some others are applicable. The most important are collected and explained here:

→ Copyright → Defamation
→ Domain Name → Data Protection
→ Trademark → Linking

Copyright In order to respect the property of the others and to build a website carefully designed, all the pages are entirely conceived. Pictures might be drawn by the developed or brought from a free provider of icons. The website uses some libraries and frameworks, yet all of these ones are under a free licensed which will be linked to the website to prove the authority. Finally because of the nature of the web application this one can be independent from the other website and use not more properties than the tested libraries and framework. In that way no copyright issues might be occured.
Domain Name  Domain Name’s are the name where the website is available. For example hw.ac.uk is the domain name of Heriot-Watt University. Find a domain name is important. It is important to know that some of them are already taken. Some others are reserved by the gouvernement, the companies or even by people. Nevertheless the web application stays a local application during the whole development. Accordingly the domain name doesn’t need to be chosen.

Trademark  By respect of the business trade rules and laws the website is not using logo, image, slogan which are referring to another product, website or business. Consequently, researches were done around important theme approched in the seating plan website to avoid misunderstanding and confusion with something already existing. In addition the name of the application *Here Are The Seats!* is not used anywhere else.

Defamation  Consequently to a real attention around the trademark, the web application is not containing any information able to be understood as a defamation.

Data Protection  The data protection is a current prominent issue. As is explained on the government website the developer should explain why he needs personal data and how he is using it. Accordingly, during the login on the application the user has to check a checkbox to prove that he knows the rules of the website and how is treated his personal data. Sensitive data is crypted before the storage in the database to ensure confidentiality, especially passwords.

Linking  The linking between website is a common pratice on the internet. Yet some rules are applicable. The link must be not perceived as demafatory by the linked part. Use licensed image with link is also not permitted. Again because of the goal of the website it was not complicated to avoid this kind of issue.

9.3 Social Issues  The web application takes care to not be involve in any social issues. No discrimination is accepted on it whatever the purpose: physical, spirtual, ethnical, religious, sexual, etc. According to the legal issues and the availability of the website any pornography or violence as well as reference to one of these purposes is accepted on the website.

9.4 Ethical Issues  Although this web application is not using sensible data all professionals, legals and socials issues are respected. In the same direction written and unwritten ethical rules and laws are followed. Consequently, this project was conducted in a very rigorous way to develop a professional application in order to support the associate dissertation.
10 Conclusion

The goal of this master dissertation was to develop an application in order to conduct some research. The project named Here Are The Seats was designed in order to solve the seating plan problem on a web application. In parallel some research were accomplished around the Play! framework, the CSPs libraries and the web application design. Both project and research were carried out conscientiously to write this master dissertation.

The website was successfully implemented. Thanks to the development and more advances research some feedbacks are presented. The feedback given on this report is principally in favor of the Play! framework which is an interesting opportunity to develop websites. The evaluation given on several good practices used to develop a website is mainly positive. Finally, the result about the research around the CSP libraries shows the drawbacks and advantages of each one. Most of them are working well and can be used depending on what the user wants to do. All requirements were accomplished. Some optional ones were developed too as is detailed in the requirements satisfaction analysis part.

Personally, this project was a real opportunity for me. First of all I have never worked before during a long period on the same project. This one allowed me to manage my work during a long development with some obstacles. In addition it was important for me to learn about the framework Play. I knew that this one was maybe an alternative for a web development. Thanks to that dissertation, I am sure it can become important in an upcoming future. My future job will certainly enforce my computer skills. Yet, I get the possibility to guide a web development in another way than usual. That can be a real asset.

Another part of this project was to work around the CSPs. The CSPs are used in lots of problems nowadays. For example, the timetabling and the organisation of supply chain are two current areas. Again this master dissertation allows me working on that problems. Knowledges around it could be valuable for me in the future.

On the developed application some improvements can be done. The most important one can be to modify the organisation of the website in order to consider the feedback given by the questionnary. Secondly, improve the application’s efficiency to allow the user adding more than sixteen people seems essential. Otherwise, a service website can be always improved, some studies can be done to improve more and more the design. On the other hand some functionalities can be added such as allow the user choosing different shapes, allow the user adding more than one best skill, allow the user adding optional persons and mandatory persons in the seating plan, etc. Finally, the research about the Play framework can be pushed further. It might be interesting to develop the same website with other web technologies such as AMP (Apache, MySql and Php), JEE or ASP .NET C# in order to compare them with the Play development. Then it will be possible to target precisely the differences and expound the Play framework’s strengths and weaknesses.
11 References & Bibliography

11.1 Books Referenced


11.2 Journals Referenced

- Cyr D., Ilsever J., Bonanni C., and Bowes J. (n.d) Website Design and Culture: An Empirical Investigation
11.3 Web Documents & Web Pages Referenced


• Tiobe Software (n.d) F# on its way to the top 10 [online]; available: http://www.tiobe.com/index.php/content/paperinfo/tpci/index.html [accessed 24 March 2014]

11.4 Bibliography

• Neil D. Jones (1973) Computability Theory, An introduction

11.5 Webography

• Play Framework Website [online], available at: http://www.playframework.com

• Choco Library Website [online], available at: http://www.emn.fr/z-info/choco-solver/

• Cream Library Website [online], available at: http://bach.istc.kobe-u.ac.jp/cream/

• Google CP Solver Website [online], available at: https://code.google.com/p/or-tools/

• JaCop Library Website [online], available at: http://jacop.osolpro.com
• JOpt Library Website [online], available at: http://jopt.sourceforge.net/index.php
• OptaPlanner Library Website [online], available at: http://www.optaplanner.org
• Artelys Kalis Website [online], available at: http://www.artelys.com/en/optimization-tools/kalis
• BCS Code of Conduct [online], available at: http://www.bcs.org/server.php?show=nav.6030

11.6 Figure Sources

No 3 www.math.cornell.edu
No 6 www.csd.uoc.gr
No 7 www.pearsonhighered.com
No 10 www.playframework.com
No 11 www.playframework.com
No 12 www.theguardian.com/uk
No 13 www.uie.com
No 15 Steve Krug (2006) Don’t Make Me Think!
No 16 Steve Krug (2006) Don’t Make Me Think!
No 17 www.nngroup.com
No 18 blog.clicktale.com
No 19 Steve Krug (2006) Don’t Make Me Think!
No 21 radar.oreilly.com
No 24 bis www.pqrst.at
No 45 www.billthelizard.com
12 Appendix

12.1 Basic Design

Here is the basic design of the page to create a new seating plan and the one to load existing seating plan:

Figure 63: Saved Seating Plan Preview

Here is the page with the seating plan and the page with people’s abilities which can interfere in the seating plan:

Figure 64: People Management Preview
12.2 NQueens Problem

12.2.1 Choco Library

```java
public class NQueenGlobal extends AbstractNQueen {

    @Override
    public void buildModel() {
        vars = new IntVar[n];
        IntVar[] diag1 = new IntVar[n];
        IntVar[] diag2 = new IntVar[n];

        for (int i = 0; i < n; i++) {
            vars[i] = VariableFactory.enumerated("Q" + i, 1, n, solver);
            diag1[i] = VariableFactory.offset(vars[i], i);
            diag2[i] = VariableFactory.offset(vars[i], -i);
        }

        solver.post(IntConstraintFactory.alldifferent(vars, "BC"));
        solver.post(IntConstraintFactory.alldifferent(diag1, "BC"));
        solver.post(IntConstraintFactory.alldifferent(diag2, "BC"));
    }

    @Override
    public void configureSearch() {
        solver.set(IntStrategyFactory.minDom_LB(vars));
    }
}
```

Figure 65: NQueens - Choco Code
12.2.2 Cream Library

```java
public int queens(int n) {
    int test=0;
    Network net = new Network();
    IntVariable[] q = new IntVariable[n];
    IntVariable[] u = new IntVariable[n];
    IntVariable[] d = new IntVariable[n];
    for (int i = 0; i < n; ++i) {
        q[i] = new IntVariable(net, 1, n);
        u[i] = q[i].add(i);
        d[i] = q[i].subtract(i);
    }
    new NotEquals(net, q);
    new NotEquals(net, u);
    new NotEquals(net, d);
    Solver solver = new DefaultSolver(net);
    long T = System.currentTimeMillis();
    for (solver.start(); solver.waitNext(); solver.resume()) {
        Solution solution = solver.getSolution();
        if (test == 0) {
            val = System.currentTimeMillis() - T;
            for (int i = 0; i < n; i++) {
                int j = solution.getIntValue(q[i]);
                System.out.print(j + " ");
            }
            return 0;
        }
    }
    solver.stop();
    return 0;
}
```

Figure 66: NQueens - Cream Code
12.2.3 Google CP Library

```java
public long solve() {
    Solver solver = new Solver("NQueens");
    IntVar[] q = solver.makeIntArray(n, 0, n-1, "q");
    solver.addConstraint(solver.makeAllDifferent(q));

    for(int i = 0; i < n; i++) {
        for(int j = 0; j < i; j++) {
            solver.addConstraint(solver.makeNonEquality(
                solver.makeSum(q[i],i).var(),solver.makeSum(q[j],j).var()));
            solver.addConstraint(solver.makeNonEquality(
                solver.makeSum(q[i],-i).var(),solver.makeSum(q[j],-j).var()));
        }
    }

    DecisionBuilder db = solver.makePhase(q,
        Solver.CHOOSE_MIN_SIZE_LOWEST_MAX, Solver.ASSIGN_CENTER_VALUE);
    long T = System.currentTimeMillis();
    solver.newSearch(db);
    while (solver.nextSolution()) {
        return(System.currentTimeMillis() - T);
    }
    solver.endSearch();
    return(System.currentTimeMillis() - T);
}
```

Figure 67: NQueens - Google CP Code
12.2.4 JaCoP Library

```java
@Override
public void model() {
    store = new Store();
    vars = new ArrayList<IntVar>();
    IntVar queens[] = new IntVar[numberQ];

    for (int i = 0; i < numberQ; i++) {
        queens[i] = new IntVar(store, "Q" + (i + 1), 1, numberQ);
        vars.add(queens[i]);
    }
    store.impose(new Alldiff(queens));

    IntVar[] diagonalUp = new IntVar[queens.length];
    IntVar[] diagonalDown = new IntVar[queens.length];
    diagonalUp[0] = queens[0];
    diagonalDown[0] = queens[0];

    for (int i = 1; i < queens.length; i++) {
        diagonalUp[i] = new IntVar(store, -2 * numberQ, 2 * numberQ);
        store.impose(new XplusEqZ(queens[i], i, diagonalUp[i]));
        diagonalDown[i] = new IntVar(store, -2 * numberQ, 2 * numberQ);
        store.impose(new XplusEqZ(queens[i], -i, diagonalDown[i]));
    }
    store.impose(new Alldiff(diagonalUp));
    store.impose(new Alldiff(diagonalDown));
}

public long solve() {
    this.model();
    long T = System.currentTimeMillis();
    this.searchSmallestMiddle();
    return (System.currentTimeMillis() - T);
}
```

Figure 68: NQueens - JaCoP Code
12.3 Travelling Salesman Problem

12.3.1 Choco Library

```java
@override
public void buildModel() {
    final int n = costMatrix.length;
    solver = new Solver();
    totalCost = VariableFactory.bounded("obj", 0, 100, solver);
    // creates a graph containing n nodes
    graph = new UndirectedGraphVar("G", solver, n, SetType.SwapArray,
        SetType.LinkedList, true);
    // adds potential edges
    for (int i = 0; i < n; i++) {
        for (int j = i + 1; j < n; j++) {
            graph.getEnvelopeGraph().addEdge(i, j);
        }
    }
    solver.post(GraphConstraintFactory.tsp(graph, totalCost, costMatrix, 2));
}

@override
public void configureSearch() {
    final GraphStrategies strategy = new GraphStrategies(graph,
        costMatrix, null);
    strategy.configure(GraphStrategies.MIN_COST, true);
    solver.set(strategy);
    SearchMonitorFactory.limitTime(solver, TIME_LIMIT);
}

@override
public void solve() {
    int k=1; int val;

    long T = System.currentTimeMillis();
    solver.findOptimalSolution(ResolutionPolicy.MINIMIZE, totalCost);
    time = System.currentTimeMillis() - T;
    System.out.println("Cost = "+totalCost.getValue());
}
```

Figure 69: TSP - Choco Code
12.3.2 Google CP Library

```java
// Solve Method
public void solve() {
    RoutingModel routing = new RoutingModel(costMatrix.length, 1);
    routing.setFirstSolutionStrategy(RoutingModel.ROUTING_PATH_CHEAPEST_ARC);
    NodeDistance distances = new NodeDistance(costMatrix);
    routing.setCost(distances);

    long T = System.currentTimeMillis();
    Assignment solution = routing.solve();
    time = System.currentTimeMillis() - T;
    if (solution != null) {
        int route_number = 0;
        for (long node = routing.start(route_number);
            !routing.isEnd(node);
            node = solution.value(routing.nextVar(node)) {
            globalRes.add((int) node);
        }
    }
    globalResCost = solution.objectiveValue();
}
```

Figure 70: TSP - Google CP Code
12.3.3 JaCoP Library

@Override
public void model() {
    final int n = costMatrix.length;

    store = new Store();
    varsMatrix = new IntVar[n][2];
    cities = new IntVar[n];
    costs = new IntVar[n];

    for (int i = 0; i < cities.length; i++) {
        cities[i] = new IntVar(store, "cities[" + (i + 1) + "]", 1, cities.length);
        costs[i] = new IntVar(store, "costs[" + (i + 1) + "]", 0, 1000);
        varsMatrix[i][0] = costs[i];
        varsMatrix[i][1] = cities[i];
    }

    store.impose(new Circuit(cities));
    for (int i = 0; i < cities.length; i++) {
        store.impose(new Element(cities[i], costMatrix[i], costs[i]));
    }

    cost = new IntVar(store, "Cost", 0, 100000);
    store.impose(new Sum(costs, cost));
}

public boolean solve() {
    int k=1, val=1;
    search = new DepthFirstSearch<IntVar>();
    SelectChoicePoint<IntVar> select = new SimpleMatrixSelect<IntVar>(
        varsMatrix, new MaxRegret<IntVar>(), new SmallestDomain<IntVar>(),
        new IndomainMin<IntVar>();
    long T = System.currentTimeMillis();
    boolean result = search.labeling(store, select, cost);
    time = System.currentTimeMillis()-T;
}

Figure 71: TSP - JaCoP Code