INSTITUTO SUPERIOR DE ENGENHARIA DO PORTO
MESTRADO EM ENGENHARIA INFORMÁTICA




Análise de Performance de Técnicas de Optimização

MARISA DANIELA DE CAMPOS FERREIRA DA SILVA
Outubro de 2017

POLITÉCNICO
DO PORTO

Instituto Superior de Engenharia do Porto

Performance Analysis of Optimization Techniques

Marisa Daniela de Campos Ferreira da Silva

Dissertation to obtain the Master's Degree in

Computer Science, Specialized Area in

Decisional and Knowledge Technologies

Orientador: Prof. Ana Maria Madureira

Co-orientador: Prof. João Paulo Pereira

Júri:

Presidente: [Nome do Presidente, Categoria, Escola]

Vogais:

[Nome do Vogal1, Categoria, Escola]

[Nome do Vogal2, Categoria, Escola] (até 4 vogais)

Oporto, February 2017

Abstract

Real-world complex optimization problems are one of the most complex challenges faced by scientific community.
Achieving the best solution for a complex problem in an acceptable time interval is not always possible. In order to solve this problem metaheuristics are one of the available resources. Having this in mind, finding a technique among others that presents better results in most executions would allow solution choosing to be more directive and assertive.
Most used techniques comprises metaheuristics. These allow to find an acceptable solution in an acceptable time interval, even if the achieved solution was not the optimal possible.
In this sense, this thesis intends to analyse four optimization techniques. Two population based techniques, one of them based in the behaviour of the bees in colony (Bee Colony) and another based in computational evolution (Genetic Algorithms). And, two single solution techniques, one based in memory structures (Tabu Search) and another based in the metallurgy industry (Simulated Annealing). These techniques were applied to two different optimization problems and computational results were registered and analysed.
A prototype was built and used to obtain the results of applying metaheuristics to the Travelling Salesman problem (TSP) and the Knapsack Problem (KP). Evaluating the results, it was not possible to prove either that all algorithms are equivalent or that one of them is better in the majority of the cases.

Keywords: Combinatorial Optimization, Metaheuristics, Tabu Search, Artificial Bee Colony, Genetic Algorithms, Simulated Annealing

Resumo

A resolução de problemas de otimização reais complexos constitui um dos grandes desafios científicos atuais.
A possibilidade de obter as melhores soluções para os problemas nem sempre é possível em tempo útil e o recurso a técnicas de otimização para os resolver de forma eficaz e eficiente é constante. Neste sentido, encontrar uma técnica que sobressaia por entre as demais permitiria usar essas técnicas de forma mais direcionada e assertiva.
Algumas das técnicas de otimização mais usadas são as meta-heurísticas. Estas permitem encontrar uma solução em tempo útil, mesmo não sendo a melhor solução possível.
Neste contexto, a presente dissertação tem por vista a análise de quatro técnicas de otimização. Duas populacionais, sendo que uma técnica é baseada no comportamento dos enxames de abelhas (*Bee Colony*) e outra baseada na computação evolucionária, algoritmos genéticos (*Genetic Algorithms*). E, por oposição, duas de solução única, a pesquisa tabu (*Tabu Search*), que se baseia nas estruturas de memória e uma técnica baseada na indústria metalúrgica, o arrefecimento simulado (*Simulated Anealing*). Estas técnicas foram aplicadas a dois problemas de otimização e os resultados computacionais, eficiência e eficácia das técnicas, foram registados e analisados.
Um protótipo foi construído e utilizado para obter os resultados da aplicação das metaheurísticas ao problema de caixeiro viajante (TSP) e ao problema da mochila (KP). Após avaliação dos resultados, não foi possível provar que existia um algoritmo que se destacava entre os demais ou que os algoritmos eram equivalentes.

Palavras-chave: Otimização Combinatória, Meta heurísticas, Pesquisa Local, Colónia de Abelhas, Algoritmo Genético, Arrefecimento Simulado, Estudo Computacional

Acknowledgments

I would like to thank my family for all the support, specially my husband, Ivo Silva, for always believing in me and for all the support he gave me through this journey. I also have to thank to my son, Gustavo Silva, because he makes me want to be a better individual and always overcome myself.
A special thanks to Doctor Ivo Pereira for being such a good friend and for making helpful and valuable suggestion, besides supporting and motivating me.
I would like to thank my supervisors, Professors Ana Maria Madureira and João Paulo Pereira for their help and support.
And finally, I thank to Professor Nuno Silva for his support and motivational words.
To all, thank you very much.

Table of Contents

1 Introduction 1

1.1 Context and Problem 1

1.2 Value Analysis 1

1.3 Expected and Achieved Results 2

1.4 Document Organization 2

2 Context and State of The Art 3

2.1 Optimization Problems 3
- 2.1.1 Travelling Salesman Problem (TSP) 5
- 2.1.2 Knapsack Problem (KP) 7

2.2 Metaheuristics 9
- 2.2.1 Tabu Search (TS) 11
- 2.2.2 Simulated Annealing (SA) 13
- 2.2.3 Bee Colony (BC) 15
- 2.2.4 Genetic Algorithms (GA) 17
- 2.2.5 Parameter Tuning and Control 21

2.3 Metaheuristics Synthesis 23

2.4 Metaheuristics Performance 24

2.5 Value Analysis 27
- 2.5.1 Value Proposal 28
- 2.5.2 Canvas Analysis 30

2.6 Approach 31

2.7 Conclusion 32

3 The System Context and Execution 33

3.1 Architecture 33

3.2 Design 34

3.3 User Interface 35

3.4 Conclusion 41

4 Computational Results 43

4.1 Evaluation Metrics 43

4.2 Parametrizations 44

4.3 Test Results 45
- 4.3.1 SQ Analysis 47
- 4.3.2 Hybrid Analysis 49

4.4 Conclusion 50

5 Conclusions and Implications 51

5.1 Achievements 51
5.2 Limitations and Further Research 52
5.3 Final Appreciation 52

6 Bibliography 53

7 Appendix 61

7.1 TSP Benchmark Problems 61
7.2 KP Benchmark Problems 85

List of Figures

Figure 1 Optimization Problems Classification [4] 4
Figure 2 Graph with example of a simple TSP [7] 5
Figure 3 Optimization Techniques [11] [12] 10
Figure 4 Metaheuristics Evolution 10
Figure 5 Guidelines for solving a given optimization problem [3] 11
Figure 6 Local search (steepest descent) behavior in a given landscape. [3] 12
Figure 7 SA escaping from local optima [3] 15
Figure 8 Behavior of honey bee foraging for nectar [23] 16
Figure 9 Mutation Techniques Representation [26] 19
Figure 10 Different Crossover Techniques representation 20
Figure 11 Order Crossover and Partially Mapped Crossover Representation 21
Figure 12 Taxonomy of parameter setting (Adapted from [13] and [28]) 22
Figure 13 Benefits versus Costs 27
Figure 14 New Concept Development Model (NCD) [35] 28
Figure 15 Canvas Analysis [37] 30
Figure 16 System Representation 33
Figure 17 Application Domain Model 34
Figure 18 Main Window 36
Figure 19 TS Parametrization 37
Figure 20 GA Parametrization 37
Figure 21 BC Parametrization 38
Figure 22 SA Parametrization 38
Figure 23 Results Main Window 39
Figure 24 Data Analysis Section 40
Figure 25 Chart Analysis 40

List of Algorithms

Algorithm 1 Representation of an LS algorithm 12
Algorithm 2 Tabu Search Algorithm [13] [17] 13
Algorithm 3 Representation of an SA algorithm [13] 14
Algorithm 4 BC Algorithm [10] 17
Algorithm 5 GA Algorithm [3] [13] 18

List of Tables

Table 1 TSP matrix based on graph presented in Figure 2 6
Table 2 Analysis of the solutions in the TSP represented in the Figure 2 graph [9] 7
Table 3 List of items and it´s properties 8
Table 4 List of items ordered by efficiency, highest to lower 9
Table 5 Analysis in order to obtain a solution for the exemplified Knapsack problem 9
Table 6 Algorithm Parameters 22
Table 7 Synthesis of Metaheuristics 24
Table 8 Instance of cities distance matrix for 6 cities by Arabi [30] 25
Table 9 Comparison of working time and best results for ESM & GA presented by Arabi [30] 25
Table 10 TSP computational results [31] 25
Table 11 Results of TSP Algorithms Execution [32] 26
Table 12 TSP Algorithms Parametrization 44
Table 13 Algorithms Parametrization 45
Table 14 TSP Computational Results 46
Table 15 KP Computational Results 46
Table 16 TSP SQ Summary 47
Table 17 KP SQ Summary 47
Table 18 TSP SQ Analysis of Variance 48
Table 19 KP SQ Analysis of Variance 48
Table 20 TSP Hybrid Summary 49
Table 21 KP Hybrid Summary 49
Table 22 TSP Hybrid Analysis of Variance 49
Table 23 KP Hybrid Analysis of Variance 50

Notation and Glossary

ACA	*Ant Colony Algorithm*
BC	*Bee Colony*
CI	*Confidence Interval*
Deterministic	*Having the same initial conditions, the output is exactly determined and is always the same*
DNA	*Deoxyribonucleic Acid*
DTO	*Data Transfer Object*
ESM	*Evolutionary algorithm*
Fitness	*Ability to satisfy the objective function of the optimization problem*
GA	*Genetic Algorithm*
GRASP	*Greedy adaptive search procedure*
HS	*Harmony Search*
KP	*Knapsack Problem*
LS	*Local Search*
NCD	*Nes Concept Development Model*
NP	*Nondeterministic Polynomial*
NPPD	*New Product and Process Development*
P	*Polynomial*
Prolog	*Logic programming language*
PSO	*Particle swarm optimization*
QA	*Quantum Annealing*
SA	*Simulated Annealing*
SAT	*Satisfiability*
SI	*Swarm Inteligence*
SQ	*Solution Quality*
Stochastic	*The models have elements with randomness*
TSP	*Travelling Salesman Problem*
TSPLib	*Library of sample instances for Travelling Salesman Problem*
UML	*Unified Modeling Language*
VLSI	*Very-Large-Scale Integration*
XAML	*eXtendible Application Markup Language*

1 Introduction

This chapter briefly presents the context and problem discussed in this document. It also presents a short description of the value analysis, expected and achieved results, as well as the document organization to give the reader a document overview.

1.1 Context and Problem

In real life, there are problems faced daily that if solved could simplify our lives or maximize profits. For instance, the Global Positioning System (GPS) based applications that suggests the best way to arrive to our destiny, a factory that is operating in order to maximize profit, a store that wants to minimize costs, and so many other problems. If there is a maximization or a minimization opportunity, then we are in front of an optimization problem.

Optimization problems can involve intensive resources consumption depending on how many variables are involved and, sometimes, it is not possible to achieve a solution in a reasonable time.

Metaheuristics came up as a new approach to this problem. They do not always grant the best solution but instead, can provide a good solution in reasonable time. Thought, the range of metaheuristics available is very wide and the aim of this document is to evaluate if it is possible to determine a better algorithm between Tabu Search (TS), Simulated Annealing (SA), Bee Colony (BC) and Genetic Algorithms (GA), when solving different optimizations problems.

1.2 Value Analysis

Despite the amount of papers and dissertations about metaheuristics applied to solve optimization problems, there are few conclusions about the impacts of each and few inferences about which one is the best in each case. [1]

This thesis purpose is to contribute with more knowledge about metaheuristics behaviour and performance by analysing impacts of selected metaheuristics when applied to different optimization problems. Having as main goal to infer if their performance is similar or if one of them standout as a better choice. Applying them to different problems makes possible to determine if their behaviour depends on problem particularities.

Making these inferences will enrich the scientific community and contribute to take a step forward helping to choose algorithms for solving optimization problems.

1.3 Expected and Achieved Results

It is expected to obtain computational results that allow to infer the impact of different algorithms applied to knapsack problem (KP) and travelling salesman problem (TSP), and also how the contraints and input parametrizations influence these results.

1.4 Document Organization

The rest of the paper is organized as follows. The context and state of art are presented in the second chapter. That chapter, besides presenting the problem, also exposes a value analysis in order to position the solution in the target market, contextualizes optimization problems and metaheuristics as one effective solution to such problems. After this positioning, is presented the state of art of metaheuristics performance and the selected approach to solve the problems.

In the third chapter it is possible to find the system context and execution, which covers the solution design, architecture as well as the user presentation.

The fourth chapter is an analysis of the computational results, considering the execution environment parametrization and evaluation metrics applied.

Chapter five concludes this document, giving an overview of the achieved results, known limitations and future work that allows to express the conclusions about the approach to the problem solution.

2 Context and State of The Art

This chapter will present the problem definition and document value analysis, why is it needed and the contribution it will give and to whom.
Further ahead, there will be a definition of optimization problems and the inclusion of concepts as polynomial problems or nondeterministic polynomial. After this context, a presentation of the metaheuristics will be made, approaching each one of the purposed metaheuristics and their role solving optimization problems.
After presenting the context, a state of art enunciating different approaches found in the literature will be made, as well as some infers about those approaches and the path to follow in order to achieve the aimed goals after all the gathered knowledge.

2.1 Optimization Problems

An optimization problem is a decision problem that requires an optimal or near-optimal solution. To talk about a problem implies to talk about objective functions which implies decision variables and restrictions.
Papadimitriou and Steiglitz [2] classify optimization problems "into two categories: those with continuous variables, and those with discrete variables, which we call combinatorial. In the continuous problems, we are generally looking for a set of real numbers or even a function; in the combinatorial problems, we are looking for an object from a finite, or possibly countably infinite, set—typically an integer, set, permutation, or graph. These two kinds of problems generally have quite different flavours, and the methods for solving them have become quite divergent."
In terms of computational complexity, optimization problems can be categorized into polynomial (P), problems that can be solved in polynomial, or nondeterministic polynomial (NP) problems, that cannot be resolved in polynomial time, and ultimately there are undecidable problems which cannot be solved by an algorithm.
Talbi [3] states that an algorithm needs time and space in order to solve a problem. The number of steps an algorithm takes in order to solve a problem reflects **time complexity**. Complexity of a problem is given by the worst case scenario analysis.
The harder NP problems are classified as NP-complete. These problems cannot be solved in polynomial time, though they can be verified in polynomial time. Meaning, with a candidate solution is possible to analyse, in polynomial time, if this solution is optimal for the problem.

There are also NP-Hard problems that “are optimization problems whose associated decision problems are NP-complete. Most of the real-world optimization problems are NP-hard for which probably efficient algorithms do not exist. They require exponential time (unless P = NP) to be solved in optimality. Metaheuristics constitute an important alternative to solve this class of problems” [3]. Some NP-hard problems are travelling salesman problem, knapsack problem, job scheduling, satisfiability (SAT), sparsest cut and bin packing problem.
In order to solve NP-Hard problems, researchers attempt to create algorithms that reach an optimal solution. Finding a solution, that may not be the global best, that qualifies as the best found within the applied solution quality measurements.

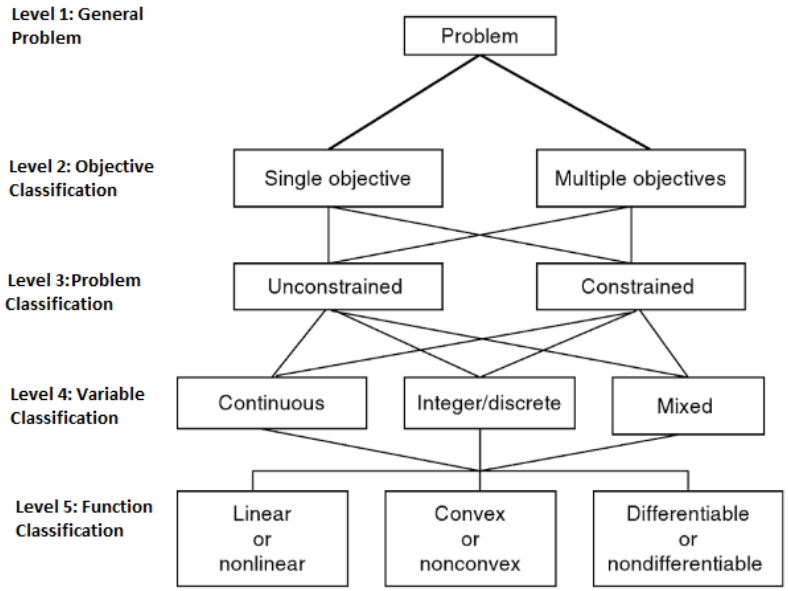

Figure 1 Optimization Problems Classification [4]

As shown in Figure 1, problems can have a single objective, when we have one objective, or multiple objective, when more than one objective to achieve. As for the problem classification “indicates whether the problem contains constraints or not. Some people believe that there are no unconstrained optimization problems in the real world, as these all will have either constraint functions or variable bounds (upper or lower) or both” [4]. As for the variable types involved in the problem they can be continuous, discrete or mixed. The function classification involves either objective and restriction functions. Functions can be linear or nonlinear. According to the linearity of the functions, if at least one of the functions involved is nonlinear, we are facing a nonlinear problem. As for

convexity is an important part of the classification because it serves as base to many optimization techniques. The differentiability is related to functions continuity and it is important when using derivative-based techniques [4].
In order to solve an optimization problem, modelling the problem is required and, therefore, as already stated, a definition of **decision variables** is needed which settles resources quantity that are needed to solve the problem, and an **objective function** which represents the relation between the variables in order to achieve the purposed objective. These can be minimization or maximization and **restrictions** are the problem limitations.

$$P = \begin{cases} min/max(f(x)) \\ \text{subject to x} \in \text{F} \subseteq \text{C} \end{cases} \quad (1)$$

Generally, optimization problems can be represented as showed in equation 1, presented by Madureira [5], in which the "decision sets C and F are discrete and may be defined as a set of variables. Each instance has a set of solutions C. The set of all possible solutions F ($F \subseteq C$) is defined by the problem restrictions, the region of impossible solutions $\frac{C}{F}$ and the objective function that attributes a real cost to each solution $x \in C$, this translates itself into $f: C \rightarrow R$. So, the goal is to achieve an admissible $x^* \in F$, in order to obtain $f(x^*) = f(x')$, so that any $x' \in F$ and $F \subseteq C$. The optimal solution would be $f \in F$: $\forall_{y \in F} C(f) \leq C(y)$" [5] [6].
Looking into each optimization problem, it is possible to extract a pattern. Finding the optimal solution may not happen in acceptable time or within computation resources consumption limits. A trade-off between resources and time spent must be agreed prior to execution; get a near optimal solution that would satisfy our relaxed restrictions sometimes could be the best solution.

2.1.1 Travelling Salesman Problem (TSP)

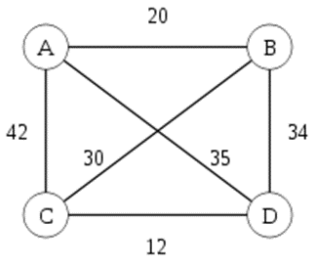

Figure 2 Graph with example of a simple TSP [7]

Travelling salesman is a typical NP-hard optimization problem. This problem consists in a salesman that wants to visit a given number of cities only once and return to the start point. The task must be done in the least time possible in order to minimize the time cost and maximize his sales revenue.
The problem resource is represented by the salesman which is assigned to the task of visiting cities in order to sell his products. Each path is categorized by having a cost, as the example shown in the Figure 2.

Table 1 TSP matrix based on graph presented in Figure 2

VERTICES (CITIES)	***A***	***B***	***C***	***D***
A	*-*	*20*	*42*	*35*
B	*20*	*-*	*30*	*34*
C	*42*	*30*	*-*	*12*
D	*35*	*34*	*12*	*-*

If we want to travel from the point A to the point B it will have a cost of 20. Each of these path costs is used to calculate the best route, defined by the least time consumable possible path found (this is what we want to minimize). A matrix can be built exposing the costs of each pair of possible paths (Table 1).

$$\begin{cases} \max(d_{\tau(1),\tau(2)} + d_{\tau(2),\tau(3)} + \cdots + d_{\tau(n),\tau(1)}) \\ subject\ to\ d_{\tau(1),\tau(2)} + d_{\tau(2),\tau(3)} + \cdots + d_{\tau(n),\tau(1)} \leq b \end{cases} \quad (2)$$

Mathematically, this problem can be interpreted by the following, “given n vertices 1,…,n and all n(n − 1)/2 distances (*d*) between them, as well as a budget (*b*). We are asked to find a tour, a cycle that passes through every vertex exactly once, of total cost b or less—or to report that no such tour exists. That is, we seek a permutation τ (1),...,τ (n) of the vertices such that when they are toured in this order, the total distance covered is at most b” [8] having as objective to maximize the number of cities visited as shown in Equation 2 where:

- $\tau\ (n)$ represents the vertex;
- *d* is distance between vertices; and
- *b* is the available budget.

For this problem we can find (N-1)! solutions to explore, five cities have 24 possible routes, but 50 cities have 6.08281864 E+62 possible routes available. This means that the complexity of the problem increases exponentially to its dimension.
Analysing specifically the problem in the Figure 2, we would have 24 possible combinations because there are only four towns to visit.

Table 2 Analysis of the solutions in the TSP represented in the Figure 2 graph [9]

SOLUTION #	*VECTOR*	*WEIGHT*	*BEST SO FAR*
1	*(A, B, C, D, A)*	*97*	*#1*
2	*(A, B, D, C, A)*	*108*	
3	*(A, C, B, D, A)*	*141*	
4	*(A, C, D, B, A)*	*108*	
5	*(A, D, B, C, A)*	*141*	
6	*(A, D, C, B, A)*	*97*	
7	*(B, A, D, C, B)*	*97*	
8	*(B, A, C, D, B)*	*108*	
9	*(B, C, D, A, B)*	*97*	
10	*(B, C, A, D, B)*	*141*	
11	*(B, D, C, A, B)*	*108*	
12	*(B, D, A, C, B)*	*141*	
13	*(C, A, B, D, C)*	*108*	
14	*(C, A, D, B, C)*	*141*	
15	*(C, B, A, D, C)*	*97*	
16	*(C, B, D, A, C)*	*141*	
17	*(C, D, A, B, C)*	*97*	
18	*(C, D, B, A, C)*	*108*	
19	*(D, A, C, B, D)*	*141*	
20	*(D, A, B, C, D)*	*97*	
21	*(D, B, C, A, D)*	*141*	
22	*(D, B, A, C, D)*	*108*	
23	*(D, C, B, A, D)*	*97*	
24	*(D, C, A, B, D)*	*108*	

We can build a table with all possible solutions in order to find the one that has the minimum tour cost, as represented in Table 2. In this case the first solution is the best to our problem.

But, if we have more cities this would not be so straight forward. We can have an exponential number of solutions to analyse depending on N, which could take a huge amount of time. This is why sometimes it is not feasible to analyse all the solutions. Instead, we need to find a near optimal solution for the problem.

2.1.2 Knapsack Problem (KP)

The KP is also an optimization problem, knapsack is the resource available and the task is to pack available items in the most efficient way. KP is defined as:

Having a knapsack and a variety of items that have a determined weight and value, and we want to carry the maximum number of items possible.

$$\begin{cases} \max(\sum_{k=1}^{n} x_k w_k) \\ subject\ to \sum_{k=1}^{n} x_k v_k \leq b \end{cases} \quad (3)$$

As in the TSP we want to maximize profit, though this is translated by maximizing the function given by Equation 3 where:

- x_k is the number of items loaded into the knapsack;
- w_k is the weight of each *k* item, for *k*=1,2,…,*n*;
- v_k is the value of each *k* item, for *k*=1,2,…,*n*; and
- b is the available budget.

$$Eficciency = \frac{Profit}{Weight} \quad (4)$$

Table 3 List of items and it´s properties

ITEMS	***WEIGHT (KG)***	***PROFIT (€)***	***EFFICIENCY (€/kg)***
A	*1*	*45*	*45*
B	*13*	*880*	*67,69*
C	*3*	*414*	*138*
D	*32*	*727*	*22,72*
E	*20*	*606*	*30,30*

To understand this easily, it is possible to see an example. Having a list of five items, as defined in Table 3, and a knapsack that allows to carry 40 kg. Intuitively, at first site you can decide to order by profit and take the most profitable items, though is not that easy, you will have to let some items behind and you do not want to make wrong choices. That is why the efficiency (Equation 4) is so important, because two items with lower profitability and also lower weight can be more valuable than one more profitable with higher weight. In Table 3 we can take by example the item D that would be the second choice if we only analyse profit, though items C and E together, only weight 23 kg and take a profit of 1020€, so it would be a better choice to take these two instead of B.

Table 4 List of items ordered by efficiency, highest to lower

ITEMS	*WEIGHT (KG)*	*PROFIT (€)*	*EFFICIENCY (€/kg)*
C	*3*	*414*	*138*
B	*13*	*880*	*67,69*
A	*1*	*45*	*45*
E	*20*	*606*	*30,30*
D	*32*	*727*	*22,72*

So, if we order the items by efficiency, we will have the Table 4 as result. In order to solve our problem, we need to add to our knapsack the items until we achieve the maximum value.

Table 5 Analysis in order to obtain a solution for the exemplified Knapsack problem

ITEMS ADDED	*TOTAL WEIGHT*	*TOTAL PROFIT*	*KNAPSACK SPACE LEFT*
C	*3*	*414*	*37*
B	*13*	*1294*	*24*
A	*1*	*1339*	*23*
E	*20*	*1945*	*3*
D	*32*		*-9*

As seen in Table 5 we would only left one item but we maximized the profit, even having 9 kg free in the knapsack [10].

2.2 Metaheuristics

In order to solve optimization problems, investigators came up with two distinct method classes, exact methods and approximate methods.

Exact methods are the methods that analyse all the available solutions in order to find the best option among them. When is not possible to solve a problem with an exact algorithm due to the solutions range, time and resources needed, the **approximate methods** approach allows finding the better solution according to the solution quality measurements (this means that there is no guarantee that is the best solution).

In the group of the approximate algorithms we can find two subsets, the **approximation algorithms** and the **heuristic algorithms**. According to Talbi [3] “unlike heuristics, which usually find reasonably “good” solutions in a reasonable time, approximation algorithms provide provable solution quality (SQ) and provable run-time bounds”.

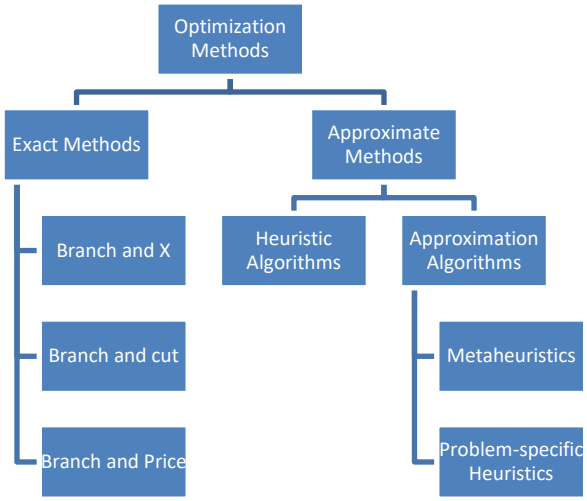

Figure 3 Optimization Techniques [11] [12]

In Figure 3 is possible to see optimization methods schematized.

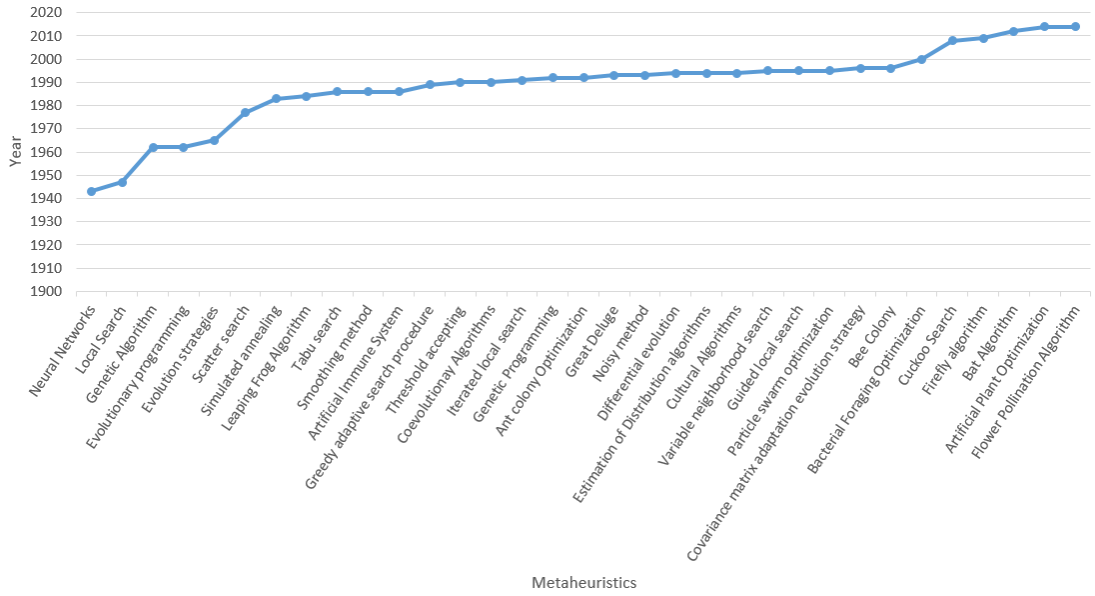

Figure 4 Metaheuristics Evolution

Metaheuristics are classified as heuristic algorithms and they allow to find optimal solutions, which can be or not the global optimal solution [5]. The term Metaheuristic was introduced by Fred W. Glover in 1986 and results from the combination of the Greek words “heuriskein”, that means finding, and “meta” that means superior level methodology [13]. Figure 4 represents metaheuristics evolution along the years.

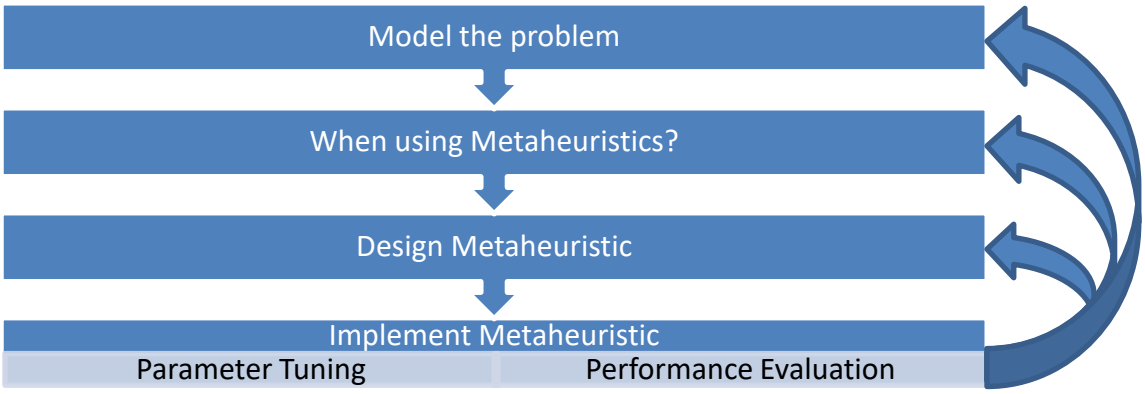

Figure 5 Guidelines for solving a given optimization problem [3]

As showed in Figure 5, Talbi [3] states that to solve an optimization problem the first step is formulate the problem model. This is possible by three main steps:

- **Determine decision variables and parameters** - The decision variables are the undetermined variables and the parameters are the known values of the algorithm. For example, if we want to buy in maximum 10€ of items, and each item costs 0.80€, we will have the decision variable *x*, that represents the number of items, and the parameter *a* would have the value of 0.80.
- **Identify Constraints** – Constraints are the model limitations. For example, in the TSP, the restriction is the travel salesman cannot visit a city more than once.
- **Identify Objective Function** – the objective function defines the quality of a solution, having in account decision variables. For instance in the items problem exemplified previously, the objective function would be $f(x) = 0{,}80x$.

After the problem is modelled is necessary to evaluate if a metaheuristics is the best approach to solve the problem. If so, metaheuristic must be designed and posteriorly implemented. When implementing a metaheuristic is necessary to account parameter tuning and performance evaluation. In any of these steps, may be necessary to take a step to previous step or even return to problem modelling.

2.2.1 Tabu Search (TS)

In order to explain TS it is important to refer that this is a local search strategy and was proposed by Fred W. Glover in 1986 [14].
Local search (LS) came up in 1947, which means this is probably the oldest metaheuristic known and boosted a great number of new metaheuristics.

```
s = s0; /* Generate an initial solution s0 */
Do
        Generate neighbor candidate
        If fitness(s) < fitness(s') then
                s = s'; /* s' is the new better solution*/
        Endif
While stopCriteria not met
Output: s (local optima).
```

Algorithm 1 Representation of an LS algorithm

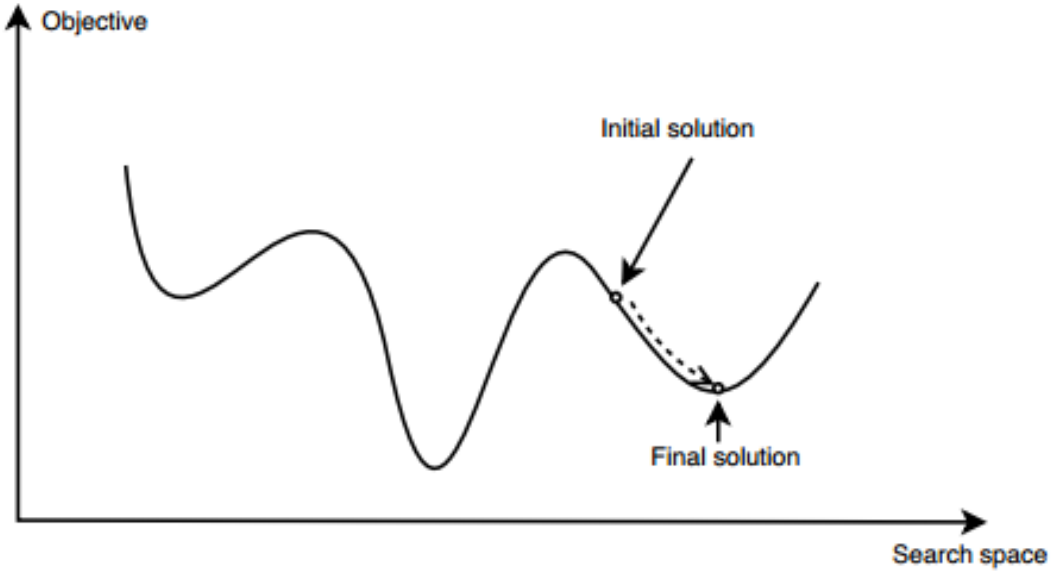

Figure 6 Local search (steepest descent) behavior in a given landscape. [3]

The LS starts from an initial point/solution and searches all the neighbours in order to infer if there is a better neighbour. If so, it selects it and analyses the neighbourhood again (Algorithm 1). The loop is stopped when, in the neighbourhood, it cannot find a better solution [3] [15].

The best algorithm performance would be to find the **optimal solution**, absolute minimum or maximum of the objective function. Though, as shown in Figure 6 the solution found was not the best solution, but the **local optimum,** which means that is the optimal (minimal or maximal) found in the neighbourhood of the initial solution and not from the whole solution space.

"TS is based on the premise that problem solving, in order to be qualified as intelligent, must incorporate adaptive memory and responsive exploration. The

adaptive memory feature of TS allows the implementation of procedures that are capable of searching the solution space economically and effectively. Since local choices are guided by information collected during the search, TS contrasts with memoryless designs that heavily rely on semi-random processes that implement a form of sampling. The emphasis on responsive exploration (and hence purpose) in tabu search, whether in a deterministic or probabilistic implementation, derives from the supposition that a bad strategic choice can often yield more information than a good random choice" [16].

```
s = s0; /* Generate an initial solution s0 */
Initialize the tabu list
Do
        Generate admissible neighbor candidate /*not tabu*/
        Update tabu list
        If fitness(s) < fitness(s') then
                s = s'; /* s' is the new better solution*/
        Endif
While stopCriteria not met
Output: s
```

Algorithm 2 Tabu Search Algorithm [13] [17]

In order to escape local optima, TS introduces the use of memory that stores in a list solutions already visited, tabu list, in order to avoid their re-visitation. As showed in Algorithm 2, the TS algorithm starts at an initial solution, initializes the tabu list and updates it in every neighbour generation. In order to be admissible, each neighbour generation must verify if the neighbour was already in the tabu list in order to avoid revisitation. The algorithm terminates when the stopping criteria is met, generally the stopping criteria consists in a total number of iterations, and returns the best solution found.

2.2.2 Simulated Annealing (SA)

The SA algorithm appeared in 1983, proposed by Kirkpatrick et al. and Cerny, applied to graph partitioning and VLSI design [3]. It has its inspiration in metallurgy. Annealing is a heat process whereby a metal is heated to a specific temperature /colour and then allowed to cool slowly. This softens the metal which means it can be cut and shaped more easily. Mild steel, is heated to a

red heat and allowed to cool slowly. However, metals such as aluminium will melt if heated for too long." [18].
In LS algorithm it is common for a search to end in a local minimum, leaving behind better solutions. This problem is first addressed by SA, and this is the first algorithm aiming to avoid local minimums, by accepting search in the neighbours of a worst solution. [19]

```
s = s_0 /* Generate an initial solution s_0*/
s_b = s /* Sets initial solution as best solution */
t = t_0/* Starts at initial temperature t0 */
Do
        Generate neighbor randomly
        If (fitness (s') > fitness (s)) then
                s=s'; /*s' is the new solution*/
                If (fitness (s) > fitness (s_b)) then
                        s_b =s; /*s' is the new solution*/
                End If
        Else
                Accepts new solution with probability P (T, f(S'), f(S))
        End If
        Updates t
While stopCriteria not met
Output: s_b
```

Algorithm 3 Representation of an SA algorithm [13]

As represented in Algorithm 3, the SA algorithm begins with the generation of an initial solution, that can be or not obtained by a heuristic (it can be a random solution), and the initialization of the temperature T. Each iteration is characterized by a generation of a random S' ∈ N(S). The acceptance of this solution depends on f(S), f(S') and T. The solution analysed is the new solution considering if, and only if, f(S') < f(S) or if f(S') ≥ f(S) with probability of T and f(s') – f(s).

$$\mathrm{P}(\mathrm{T}, \mathrm{f}(S'), \mathrm{f}(S)) = \exp\left(\frac{-\mathrm{f}(S') - \mathrm{f}(S)}{T}\right) \quad (5)$$

The probability of accepting a solution is given by equation 5. So, once the temperature decreases during the cycle, which can happen linearly (T=$T-\beta$) or geometrically (T=αT), the probability of accepting a not so good quality solution

is greater in the beginning of the process, and diminishes during the process and the solutions found would be more and more selective and accurate. In order to find an optimal solution is necessary to maintain, not only the last solution found, but also the best solution found during the process [20].

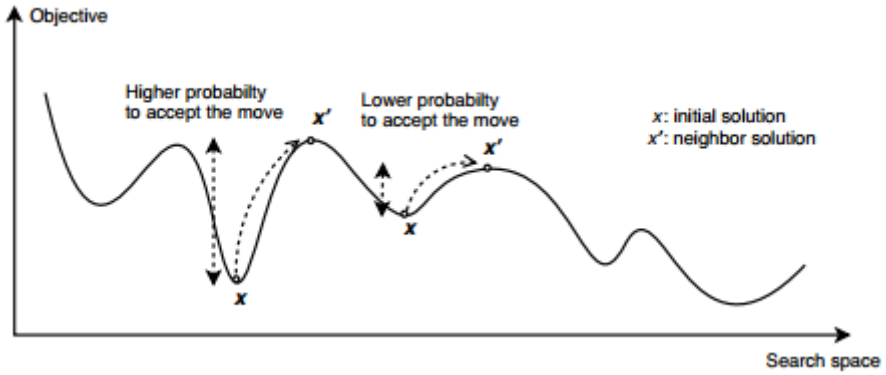

Figure 7 SA escaping from local optima [3]

As showed in Figure 7, “the higher the temperature, the more significant the probability of accepting a worst move. At a given temperature, the lower the increase of the objective function, the more significant the probability of accepting the move. A better move is always accepted” [3].

2.2.3 Bee Colony (BC)

SI comprehends collective intelligence of a swarm of animals when they are looking for their survival needs, such as food sources, mating partners or any other. “Fish schools, bird flocks, ant colonies and animal herds, with their amazing self-organization capabilities and reactions, produce collective behaviours that cannot be described simply by aggregating the behaviour of each team member” [21].
Millonas [22] enunciates five evolutionary principles: proximity, quality, diverse response, stability and adaptability. **Proximity** states that the group should be able to perform space and time computations; **Quality** enunciates that the group should respond also to quality factors, such as the quality of the food sources for example; **Diverse Response** concerns the need to explore other itineraries in order to produce different results if a scenario changes; **Stability,** stands for the ability of the group to maintain their behaviour when environment changes are introduced and finally, **Adaptability** corresponds to the group ability to change their behaviour when necessary.

As already referred in the introduction of this chapter, BC algorithm is an algorithm included in the SI class. This is a nature inspired stochastic algorithm and as the name inspires, it is based on the bee colony behaviour.

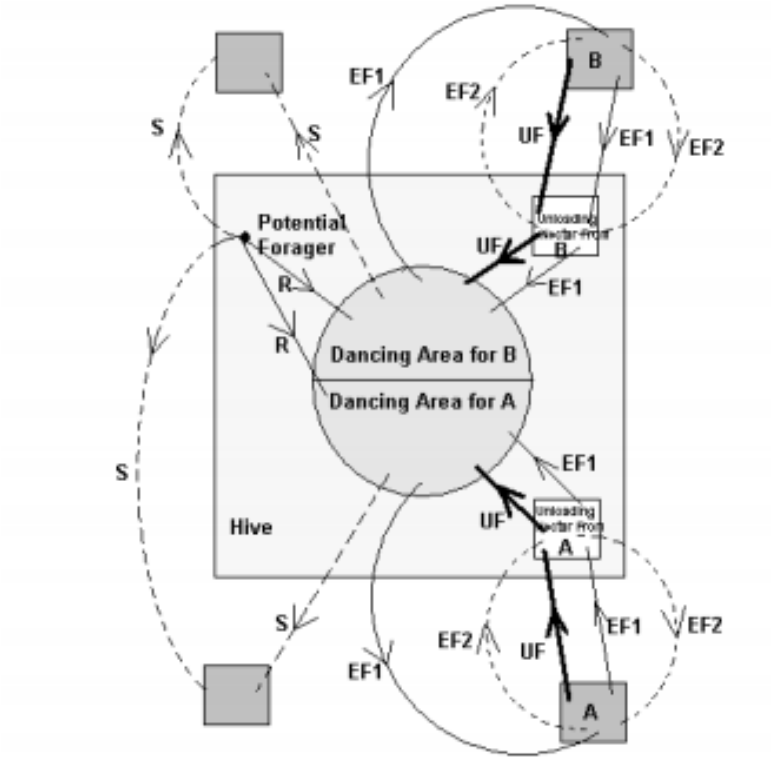

Figure 8 Behavior of honey bee foraging for nectar [23]

Talbi [23] states that the minimal model of honey bee swarms intelligence consists in three main components food sources, employed foragers and unemployed foragers. Having two leading models of behaviour: the recruitment to a nectar source and the abandonment of a source.

A bee colony as a **queen**, the reproductive female, and thousands of males, known as **drones**. Drones cope with the queen in order to produce **broods**, young bees. Colony **workers** are the sterile females [3].

Bee´s work as a team, they share knowledge in order to find food more efficiently. As showed in the Figure 8, bee´s behaviour can be interpreted as: the scout bee searches randomly around the nest for food (S), or they can be recruited by foraging bees with *waggle dances* that indicate food sites. If the bee is recruited it loads the food and unloads it on the nest, after unloading the food the bee can become an uncommitted forager (UF) and they can be a scout

or an onlooker that dances and recruits more workers before returning to food source (EF1) or continue to forage food (EF2) [23].

```
Initialization of the bee colony
Evaluate bee population fitness
While stop conditions not verified do
        Working bees phase
        Onlooker bees phase
        if food source is running low
                Scouting bees phase initializes
        End
End While
Output better solution
```

Algorithm 4 BC Algorithm [10]

As shown in Algorithm 4, the BC algorithm begins with the bee colony initialization and population fitness evaluation. Then the bee working phase is initialized, meaning bee´s find a food source and brings the food to the nest, then the onlookers bees phase is initialized, and bee´s transmit the food source in order to inform other bees and make them go work and bring more food. When the food source runs low, the scouting bees phase is initialized. This cycle is maintained until the stop criteria is met. And in the end, outputs the best solution found.

2.2.4 Genetic Algorithms (GA)

GAs were first described by John Holland in the 1960s and further developed by Holland and his students and colleagues at the University of Michigan in the 1960s and 1970s. Holland's goal was to understand the phenomenon of "adaptation" as it occurs in nature and to develop ways in which the mechanisms of natural adaptation might be imported into computer systems. Holland [24] presented the GA as an abstraction of biological evolution and gave a theoretical framework for adaptation under the GA [25].
GA are inspired by Darwin´s evolution theory of the species. Basically, Darwin proved that all species descend from common ancestors that suffered some mutation (**Mutation**) in order to survive and adapt. A species survives due to the capacity to reproduce, feed and win daily battles against predators and competitors (**Natural Selection**), so he believed that only the fit individuals survive. This way, females choose fit males in order to reproduce and guarantee

the species survival (**Crossover and Mutation**). Basically, to survive the individual needs to evolve and adapt.
Our chromosomes contain molecules of DNA (Deoxyribonucleic acid) which contains our genetic information, meaning that in the DNA we have the information that provides our, for example, eye colour or hair colour. When an individual is formed, there is a union between genetic information from the female and the male parent, this information is selected, crossed and mutated in order to create a new one with different characteristics for the new individual.

```
P = p0 /* Generate initial population p0 */
g=0 /* Starts at initial generation 0 */
Evaluate population fitness
Do
        Select Individuals
        Foreach pair, with reposition, M pairs of individuals from P
                Select randomly a number p ∈ [0,1]
                if p < probability of crossover χ then
                        Generate children with a crossover operator
                Else
                        Generate children identical to their parents
                End
        End For
        Foreach Children S
                Apply mutation Operator
                If f(P) < f(P') then
                        P'=P
                End If
        End For
        Increment generation g
        Evaluate population fitness
        Select 2 individuals from P without repositioning
        Replace the 2 individuals by their children
While stopCriteria not met
Output better solution
```

Algorithm 5 GA Algorithm [3] [13]

As showed in Algorithm 5 GA first initializes the population, starts a first generation and evaluates the fitness of the generated population. Then, it selects individuals to reproduce. For each pair of individuals (female and male), the algorithm tries to simulate natural selection, generating a random probability. If the crossover probability is higher than the random probability, then a

crossover method is applied in order to generate new children, else it is generated a child identical to their parent. In order to insert the environment adaptation of species, for each children generated a mutation operator is applied. If this operator updates the children and improves the fitness, it is maintained, else the children remains immutable. This cycle repeats itself until the stop conditions are met.

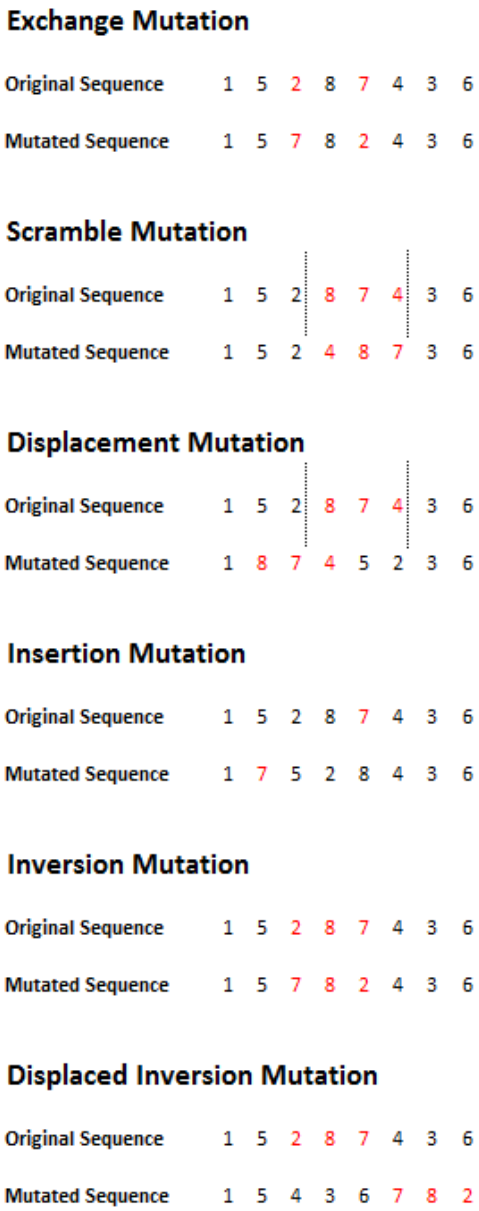

Figure 9 Mutation Techniques Representation [26]

Literature identifies several mutation operators such as [26]:

- **Exchange Mutation** (two random genes are selected, and they are exchanged) [26];
- **Scramble Mutation** (two points are selected, and the genes between them are scrambled) [26];
- **Displacement** (two random points are selected and the genes between them change their position in the group) [26];
- **Insertion Mutation** (one gene is selected randomly, this gene is removed and reinserted at a random position in the sequence) [26];
- **Inversion Mutation** (two points are selected, and the genes between them are reorganized from the end to the beginning) [26]; and
- **Displaced Inversion Mutation** (two points are selected, and the genes between them are reorganized from the end to the beginning and replaced in another position in the sequence) [26].

Crossover operator allows to take a sequence and recombine it in order to form new individuals [3].

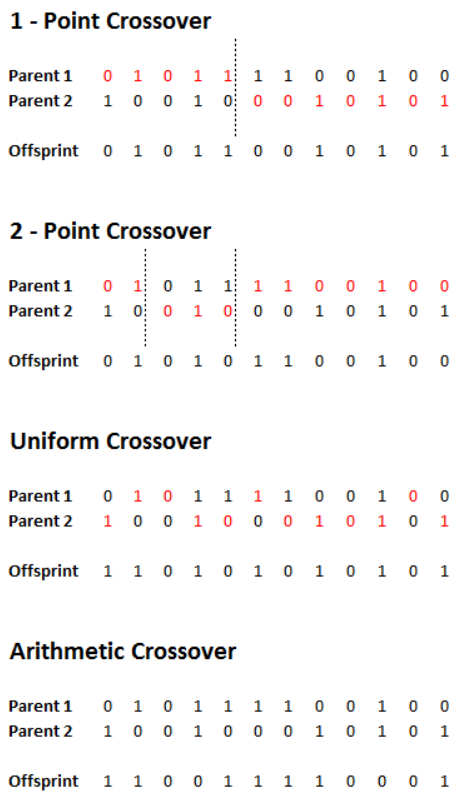

Figure 10 Different Crossover Techniques representation

Literature refers several crossover operators such as [3]:

- **1-point crossover** (divides the two sequences in half and recombines them with each other) or **n-point crossover** (divides each sequence into n parts and recombines them with each other);
- **uniform crossover** (copies randomly the bits from the sequence of one parent to the other); and
- **arithmetic crossover** (the crossover is made by the arithmetic sum of the bits from the parent's sequence).

In Figure 10 are represented these techniques in order to easily understand these concepts

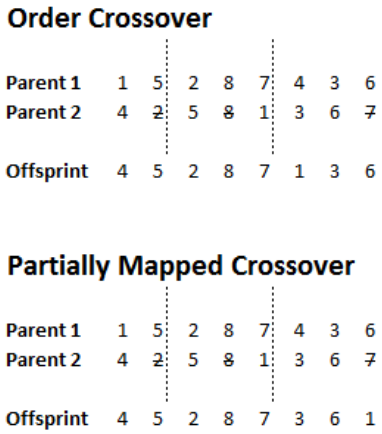

Figure 11 Order Crossover and Partially Mapped Crossover Representation

There are also permutation crossovers, such as:

- **order crossover** (as showed in Figure 11, after applying the 2-point crossover, a recombination is made by reorganizing the order the sequences are obtained), and
- **partially mapped crossover** (after applying a 2-point crossover, the middle elements are switched for the siblings and the remaining are recombined).

2.2.5 Parameter Tuning and Control

Every algorithm has parameters associated. These values can make the difference when trying to find an optimal solution. This is why parameter tuning is widely referred and studied by the scientific community.

Table 6 Algorithm Parameters

Algorithm	***GA***	***SA***	***TS***	***BC***
Entry Parameters	*Population Size*	*Initial Temperature*	*Tabu List Length*	*Number of Food Sources*
	Crossover Rate	*Alpha*	*Number of Iterations*	*Number of Bees*
	Mutation Rate	*Number of Iterations at Same Temperature*	*Number of Iterations Without Better*	*Number of Cycles Without Better Food Source*
	Layers Without Better Individual	*Number of Iterations*		
	Number of Generations			

In Table 6 it is possible to analyse initial parameters for each algorithm. If we consider combinations between all initial parameters of an algorithm we can get an enormous number of combinations, the question is, which one is the best combination to solve the problem in hands? No Free Lunch (NFL) theorem proves that is not possible to answer this question, and it states that "if an algorithm does particularly well on average for one class of problems then it must do worse on average over the remaining problems. In particular, if an algorithm performs better than random search on some class of problems then it must perform worse than random search on the remaining problems. Thus comparisons reporting the performance of a particular algorithm with a particular parameter setting on a few sample problems are of limited utility. While such results do indicate behaviour on the narrow range of problems considered, one should be very wary of trying to generalize those results to other problems" [27].

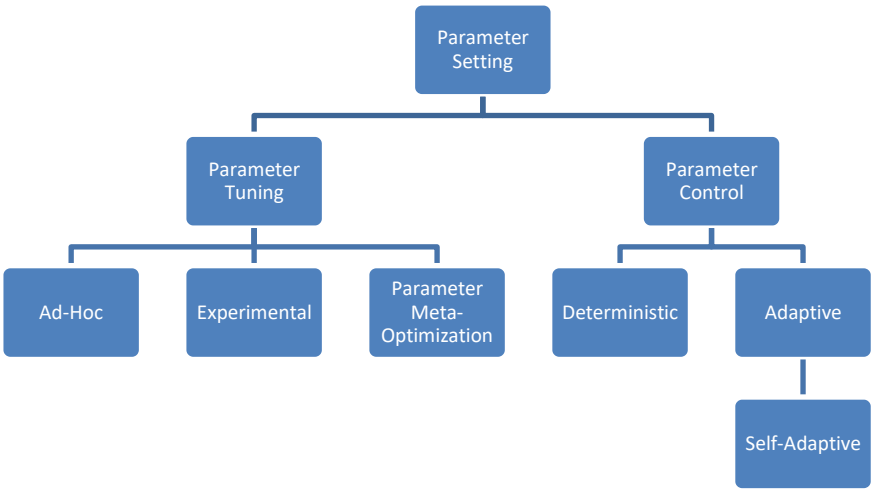

Figure 12 Taxonomy of parameter setting (Adapted from [13] and [28])

Eiben and Smith [28] distinguishes two major forms of setting parameter values: **parameter tuning** and **parameter control**. Figure 12 illustrates the taxonomy of parameter setting**.**

Parameter tuning concerns the definition of good values for parameters before executing the algorithm. This is particularly difficult since parameters are not independent, they are always a combination of parameters and they are influencers of each other, and the number of combinations is practically impossible; parameter tuning is time consuming due to all combinations available and parameters optimality change from one problem to another. Neumüller [29] classifies parameter tuning as: **Ad-Hoc**, defining parameter values based on conventions and default values; **Experimental**, when the values are experimented in order to achieve a better solution (sometimes is impossible to test all combinations) or **Parameter Meta-Optimization**, which faces the parametrization tuning as an optimization problem by itself.

Parameter control is a parametrization made during execution and, as showed in Figure 12, can be classified into three main groups as stated by Eiben and Smith [28]: **Deterministic**, this strategy concerns the parameter update based in some deterministic rule, not having in account the search process feedback; **Adaptive**, this takes place when the strategy accounts the search process feedback; and finally, **Self-Adaptive** is evolutionary based and takes place when parameters are encoded and associated to an individual, and genetic selection, mutations and crossovers are applied in order to find individuals with best fitness (this strategy only works at individual-level).

2.3 Metaheuristics Synthesis

There are several algorithms that can provide a solution to an optimization problem, though they all have advantages and disadvantages. The difficulty is to determine which one is the best to solve the problem we are addressing.

It is important to mention that metaheuristics are widely studied and there are several advances regarding this matter, introducing hybrid versions and evolutions of existing algorithms, given their flexibility and large spectrum of application. Chosen algorithms are a small sample, though they represent genetic and evolution theory, metallurgy and nature inspired, as well as a memory inspired algorithms.

Table 7 Synthesis of Metaheuristics

Algorithm	***GA***	***SA***	***TS***	***BC***
Author	*J. Holland*	*Kirkpatrick et al*	*Glover*	*D. Karaboga*
Year	*1960s*	*1983*	*1986*	*2005*
Inspiration	*Genetic and evolution theory*	*Metallurgy*	*Memory*	*Nature –Bee Colony Behavior*
Initial Solution	*Population*	*Single Solution*	*Single solution*	*Population*
Advantages	*Avoids local optimum due to crossover and mutation techniques associated with probabilities*	*Avoids local optimum*	*Easy to implement and to achieve a solution* *Avoids local optima due to Tabu list*	*Fast convergence to a solution* *Avoids local optimum*
Disadvantages	*Cannot guarantee the optimum solution*	*Cannot guarantee optimum solution*	*Cannot guarantee optimum solution* *Can result in a large number of iterations*	*Cannot guarantee optimum solution* *Initial solution delimits the search*

As shown in Table 7, by grouping algorithms by their initial solution it is possible to have two groups, population, that includes GA and BC, and single solution, includes SA and TS.

One of the major disadvantages of all the addressed metaheuristics is the absence of a guarantee that solution found is the optimal solution, though it provides a local optimum, the best result possible in the timespan available. All algorithms implement techniques to avoid being stuck on local optimum which can prevent the algorithm to find better options before stopping criteria is met.

2.4 Metaheuristics Performance

Talbi [3] states that to evaluate metaheuristics performance, there are three main steps to follow, which are:

- **Experimental Design** – This is the first step to be followed and includes a definition of experiments goals and select instances;
- **Measurements** – This second step involves the metrics definition to be computed, application of statistical analysis to results and finally, the performance analysis is made based in state-of-art optimization;

- **Reporting** – Last but not least, a comprehensive analysis of obtained results should be done according to experimental goals defined.

Table 8 Instance of cities distance matrix for 6 cities by Arabi [30]

Cities	***0***	***1***	***2***	***3***	***4***	***5***
0	*0*	*17*	*19*	*3*	*1*	*14*
1	*17*	*0*	*3*	*10*	*12*	*13*
2	*19*	*3*	*0*	*16*	*4*	*6*
3	*3*	*10*	*16*	*0*	*3*	*11*
4	*1*	*12*	*4*	*3*	*0*	*18*
5	*14*	*13*	*6*	*11*	*18*	*0*

Arabi [30] studied the computational performance of GA versus Exhaustive Search Method (ESM) solving TSP, having matrix of distances represented at Table 8.

Table 9 Comparison of working time and best results for ESM & GA presented by Arabi [30]

Towns Visited	***GA Worse***	***GA best***	***GA Average***	***GA work time, sec***	***GA result***
8	*141*	*92*	*98*	*0,52*	*84*
9	*-*	*-*	*-*	*0,55*	*82*
10	*-*	*-*	*-*	*0,54*	*112*
11	*-*	*-*	*-*	*0,58*	*111*
12	*187*	*123*	*132*	*0,57*	*145*

In Table 9 are presented results obtained by Arabi when using GA parameters with 20 individuals in population and 50 generation. Since ESM is not object of study for this problem it was excluded.

Table 10 TSP computational results [31]

Criterion	***GA***	***HS***	***PSO***	***QA***	***SA***	***TS***	***2-OPT***
Mean value	*2,37*	*2,15*	*2,97*	*2,02*	*1,08*	*1,21*	*1,68*
Standard Deviation	*0,82*	*1,21*	*0,92*	*1,27*	*0,69*	*0,44*	*1,66*
Solution Time	*1,25*	*1,34*	*1,09*	*1,26*	*0,95*	*0,67*	*0,44*

Antosiewicz, Koloch and Kamiński [31] compared algorithms applied to TSP, though setting as stopping time 100 seconds and obtain the table presented in

Table 10 which comprises mean value, standard deviation and solution times. Test columns represent algorithms, GA, Harmony Search (HS), Particle Swarm Optimization (PSO), Quantum Annealing (QA), SA, TS and greedy (2-OPT). Algorithms were initialized with a random generated solution and were made 10 executions.

"The best quality solutions were generated using simulated annealing. The performance of tabu search was also satisfactory and importantly – guaranteed stable "run to run" optimization results. Both of these are algorithms which process a single solution in each iteration, in contrast to other algorithms which process a population of solutions in each iteration. Also these are the algorithms which, during the designated time span transverse the largest number of trial solutions. (…) With regard to solution time, among metaheuristic algorithms, tabu search is the best one, with swarm optimization and simulated annealing in second place." [31]

Table 11 Results of TSP Algorithms Execution [32]

Problem Number of Cities	***Times***	***ACA***			***GA***			***TSPLIB***
		Best	*Average*	*Worst*	*Best*	*Average*	*Worst*	
30	*20*	*423.74*	*429.7*	*432.46*	*423.74*	*456.68*	*502.57*	*423.74*
	50	*423.74*	*428.76*	*432.45*	*423.74*	*425.7*	*439.84*	
48	*20*	*33780*	*35595*	*36534*	*37880*	*38833*	*38894*	*33522*
	50	*33780*	*35533*	*36534*	*35633*	*38541*	*42458*	
51	*20*	*426*	*428.4*	*431.42*	*495*	*512*	*536*	*426*
	50	*426*	*428*	*432.01*	*494*	*514*	*554*	

Li and others [32] studied GA and Ant Colony Algorithm (ACA) applied to TSP and obtained results presented in Table 11. First column represents number of cities of the problem, second column represents number of executions. ACA and GA best, average and worst objective function results are presented also, as well as TSPLIB which represents the best objective function for the problem. Is important to refer that TSPLIB is a library of sample instances for TSP [33].

These results were obtained by applying to GA a crossover rate of 0.75% and a mutation rate of 0.05%. Li [32] concluded that "Genetic Algorithms have rapid global searching capability, but the feedback of information in the system has not been utilized, sometimes leading to do-nothing redundant iteration and inefficient solution. Therefore, if the size of the cities are more than 30, genetic algorithms searching capability will gradually decline to a certain extent, when the number of the cities are too big, it cannot obtain the optimal solution in finite iteration, because iterative times are too long and unbearable. Here ant colony algorithm is better than genetic algorithm, and it can reach the optimum value.

When the size of cities is too big, ant colony algorithm may appear stagnation. Thereby it cannot obtain optimum value”

2.5 Value Analysis

A product or service value is the perceived value by the customer point of view. The service value can imply the document viability or unviability. In order to understand the perceived value there is a simple example. There are two houses, one of them next to a bus station, and the other with a bus station at 10 km. Assuming they have the same characteristics and commercial values, two different persons can evaluate them differently, for instance a person who always uses the car to work would not value the house as much as a person who travels by bus in a daily basis. This is why the value analysis is so important. It is necessary to evaluate the **target** of the service/product in order to add enough value to make the service/product attractive and **“profitable”,** whether talking about economic or knowledge value when analysing costs versus benefits of our **offer** as well as the **differentiation** from the alternatives.
Analysing the perceived value in a longitudinal perspective. There are four main phases:

- Pre-purchase – there is an attempt to understand what would be the perceived value for the clients;
- At the point of trade or experience – value perceived by the client at the moment of the purchase/acquisition;
- Post-Purchase – at this phase, the client already experienced the purchase/acquisition and there is an effective use value;
- After use/experience – point of disposal of the purchase/acquisition, at this point the purchase/acquisition lost perceived value for the clients.

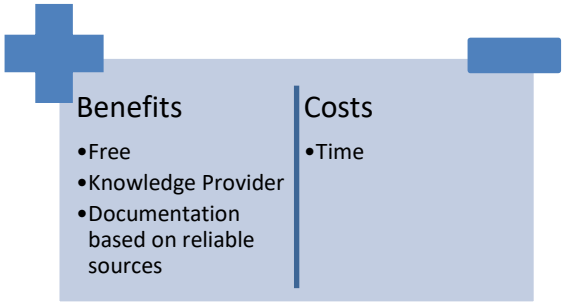

Figure 13 Benefits versus Costs

The present document is intended to bring customer value, by providing knowledge to all interested with a quality product, based in reliable sources, and free, costing only the time to analyse the documentation. In this way, is logical to infer that the value perceived will be positive, because the benefits supplant the costs (Figure 13). Though the free nature and first goal of knowledge provider, this product can contribute to help software development based in metaheuristics and can generate money for the clients. This is a win-win negotiation, once the knowledge providing ability of the present document is two way based, providing value for both author and customers.

2.5.1 Value Proposal

The present document aims to contribute to the scientific community, by the presentation of a software able to extract metrics of computational behaviour when applying metaheuristics into different optimization problems approaches. This is a free service, having as main goal to increase knowledge sources, having a free price level.

Koen, Bertels and Kleinschmidt [34] states that innovation process may be divided into three main parts, front end of innovation, new product development process, and commercialization. **Front End** is often characterized as a three degree stages. First stage is pre-work done in order to discover new opportunities. Second stage is the scoping stage, which comprehends the assessments of marketing and technical merits of the project. In the end, the third stage results in a detailed business case.

The front end of innovation includes planning and answering questions that can determine document viability, this is why this phase is so important. In order to simplify this planning and analysis, Koen and his team surfaced with a new model, **New Concept Development Model** (NCD).

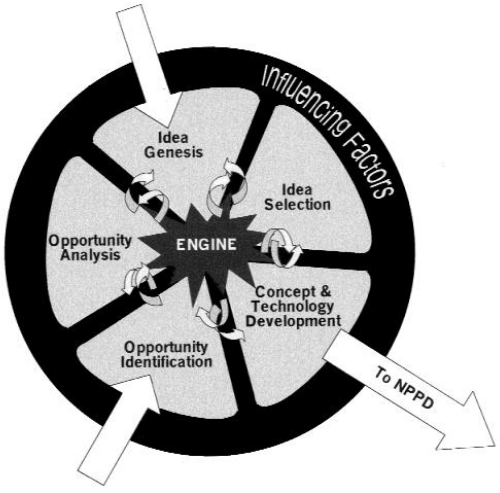

Figure 14 New Concept Development Model (NCD) [35]

As stated by Koen [36] and presented in Figure 14, NCD model compares the product development to engine wheel, which have five main elements that flow, articulate and iterate on each other by exchange of ideas that are influenced by external factors, which is why they are represented in the rim, external part of the wheel. As external influencers there are organizational capabilities, competitor threats, trends, regulatory changes and others. The arrows pointed to the interior of the wheel signals the initial phase of the development process, that can begin by opportunity identification, this phase is the better phase to initiate this process, or by the idea genesis. The exiting arrow represents the result of the process in the form of New Product and Process Development (NPPD).

The inner wheel elements are:

- Opportunity Identification – this is the point where are defined the opportunities to pursue;
- Opportunity Analysis - an intensive analysis of the opportunities identified in the previous phase. “This element may be part of a formal process or may be occurring iteratively in reaction to opportunities identified, such as “what-if” scenarios. Hard, quantifiable templates, which would be used in the NPPD, are typically not applied in this element. Both competitive intelligence and trend analyses are used extensively in this element” [35];
- Idea Genesis – this element refers to the idea construction based in the identified opportunity;
- Idea Selection – This phase is characterized by the analysis and selection of the ideas;
- Concept & Technology Development – “The final element of the model involves the development of a business case based on estimates of market potential, customer needs, investment requirements, competitor assessments, technology unknowns, and overall project risk. The level of formality of the business case varies according to the nature of the opportunity (e.g., new market, new technology and/or new platform), level of resources, organizational requirements to proceed to the NPPD and the business culture (formal, informal or hybrid). In some organizations, this is considered the initial stage (i.e., Stage 0) of the NPPD process.” [35].

Applying NCD to the present document, is fair to say that the beginning is set at the idea genesis and selection, since the idea is to explore the metaheuristics application on optimization problems and the attempt to set some relation between the best solution in the best computational time (CT), and the different algorithms in order to infer if is there any tendency.

Towards the idea evolution, there was the need of a concept. It was defined to produce a software application that allows users to parametrize the initial state and optimization problems variables, and present to the user computational results of the selected algorithms applied to those problems, presenting computational data able to allow inferences about metaheuristics performance. These inferences, besides contributing to scientific knowledge basis, can even be applied into organizations. For example TSP conclusion can help if there is the need to schedule a sales department routing and KP can help in the logistics department when packaging orders.
Analysing the concept and researching forward, were not found evidences of an analysis applying these four specific algorithms and their comparison when applied to a specific optimization problem, adding a comparison with other optimization problems performance. It is also aimed to use C-Sharp language, there are more computational studies using *PROLOG, Java* and *Eclips*e, so this could be a viable idea and concept.

2.5.2 Canvas Analysis

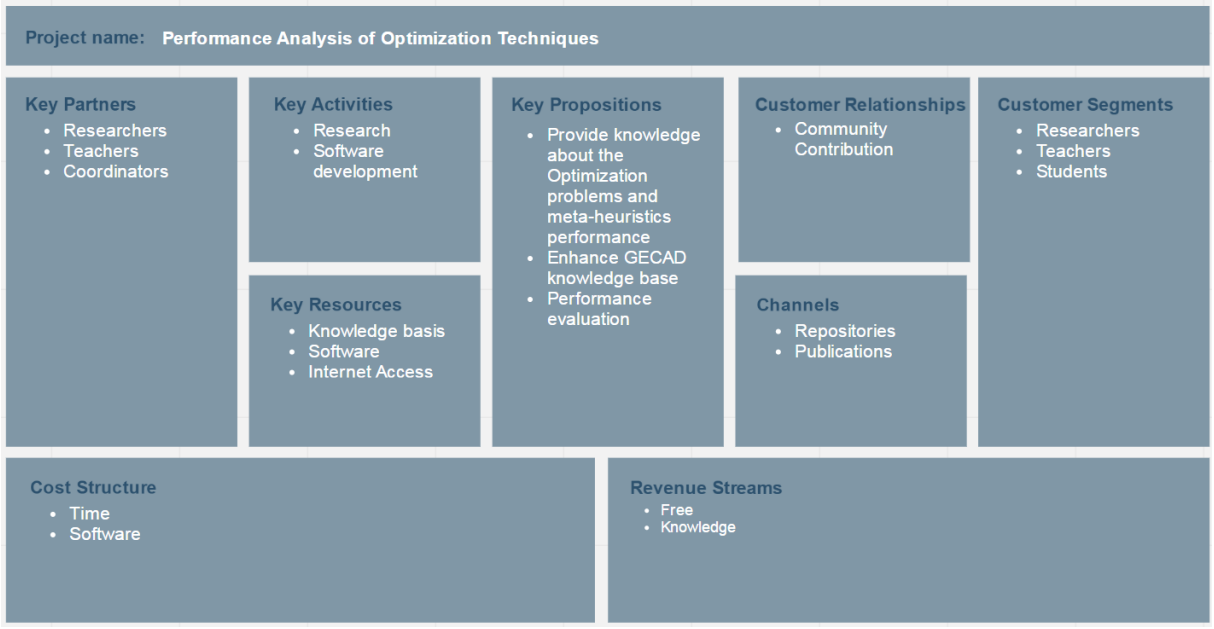

Figure 15 Canvas Analysis [37]

Figure 15 exposes a Canvas Analysis. This analysis allows to reflect and produce an overview of the business, approaching several keys from the business models.

The present document key partners are researchers because they publish articles, papers and books that allow to evaluate the state of art and produce new evidences and products; teachers who transmit knowledge in order to allow the software development, paper writing and problem analysis and solution achievement easier; and finally coordinators who review the document and supervise the development.
When approaching key activities, this document invests into Research and software development, having as resources the “Grupo de Investigação em Engenharia do Conhecimento e Apoio à Decisão” (GECAD) knowledge basis, “Instituto Superior de Engenharia do Porto” (ISEP) as well as the knowledge available online through internet access and software to develop the aimed solution. As key propositions this document purposes to achieve more knowledge about optimization problems and metaheuristics performance, in order to enhance scientific community knowledge base.
This document will be available to all community, so the customer relationship is more accurately categorized as Community contribution. This knowledge is scattered by repositories and publications.
Customer segments targeted are researchers, teachers and students, and this document will allow to provide knowledge to all these groups.
As for costs, this document will take time and software investment, though the revenue streams are free and contribute with knowledge acquisition.

2.6 Approach

In the set of optimization problems, there is a whole role of problems, widely approached in the literature. In this chapter were referred some of these problems. This document will approach problems with less complexity because it is easier to make inferences in less complex problems and only then move to more complex ones.
Having this in mind, the document developed will approach only two of the described problems, TSP and KP, other problems can be explored further more in future work.
In order to define algorithms parameterization, and since this is a scientific study that needs to account different parametrizations and compare them with others, the parametrization will be made by an experimental parameter tuning approach.

2.7 Conclusion

In this chapter it was possible to present the context, approaching the problem and focusing this document purpose. During this presentation optimization problems were addressed.

One of the most applied techniques to solve complex optimization problems, metaheuristics, were explained focusing in the algorithms that are the basis to the addressed problem. An important issue that was mentioned was the parameter tuning which is important to understand in order to develop stable and reliable solution.

In this chapter it was described the document contribution through its value analysis, which presents arguments to make it viable as well as a Canvas analysis.

3 The System Context and Execution

Following the document goal of achieving computational results, urges the need to create an application that solves the two selected problems by implementing the defined metaheuristics.
This chapter is dedicated to the construction of that application, presenting the conceptual decisions, architecture, solution design and user interface.

3.1 Architecture

The implemented system basis are the metaheuristics. In order to make correct inferences, metaheuristics must be implemented in their spirit, according to algorithms description in section 2.2.

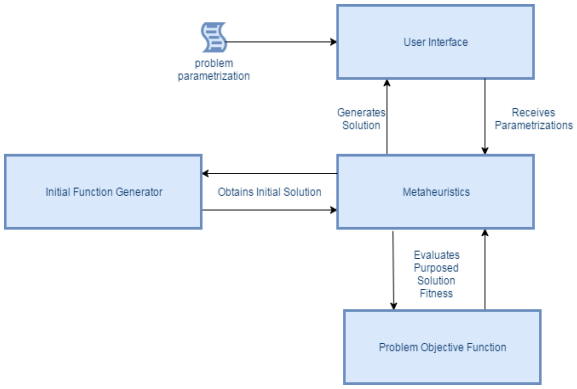

Figure 16 System Representation

Metaheuristics performance is going to be evaluated for two different problems, KP and TSP so there is the need to receive problem input parameters also. Both approached problems should allow the input of objective function, initial solution generation strategy and budget (distance budget in case of TSP and weight budget in case of KP). TSP is required to have cities that should have a name to be easily identified and coordinates in order to calculate distances. As for KP, it should have items which knapsack insertion will be evaluated, these items should have a defined weight and profit. There is the need of maintain the scheduling sample between executions, so user upload of files, manual or random parametrization of cities/items is required.
As shown in Figure 16, system receives problem parametrization and creates objects to easily transfer information into metaheuristics module, because of this transferring function, these objects are named Data Transfer Objects (DTO). Metaheuristics module contains metaheuristics algorithms implementation, in

order to start algorithms iterations, there is the need to set an initial solution, which is also defined by user input. After ordering cities/items according to user instructions, algorithm execution begins. Whenever algorithm requires a fitness evaluation, system will call problem objective function module, which evaluates solution according to objective function defined for the problem in hands.
After this process, metaheuristics module must return results to user interface, in order to present them to the user.

3.2 Design

The application domain model is presented in Figure 17. Naturally it is possible to create *Metaheuristics*, *Objective Functions* and problem entities, such as *Knapsack Problem* and *Travelling Salesman Problem*.

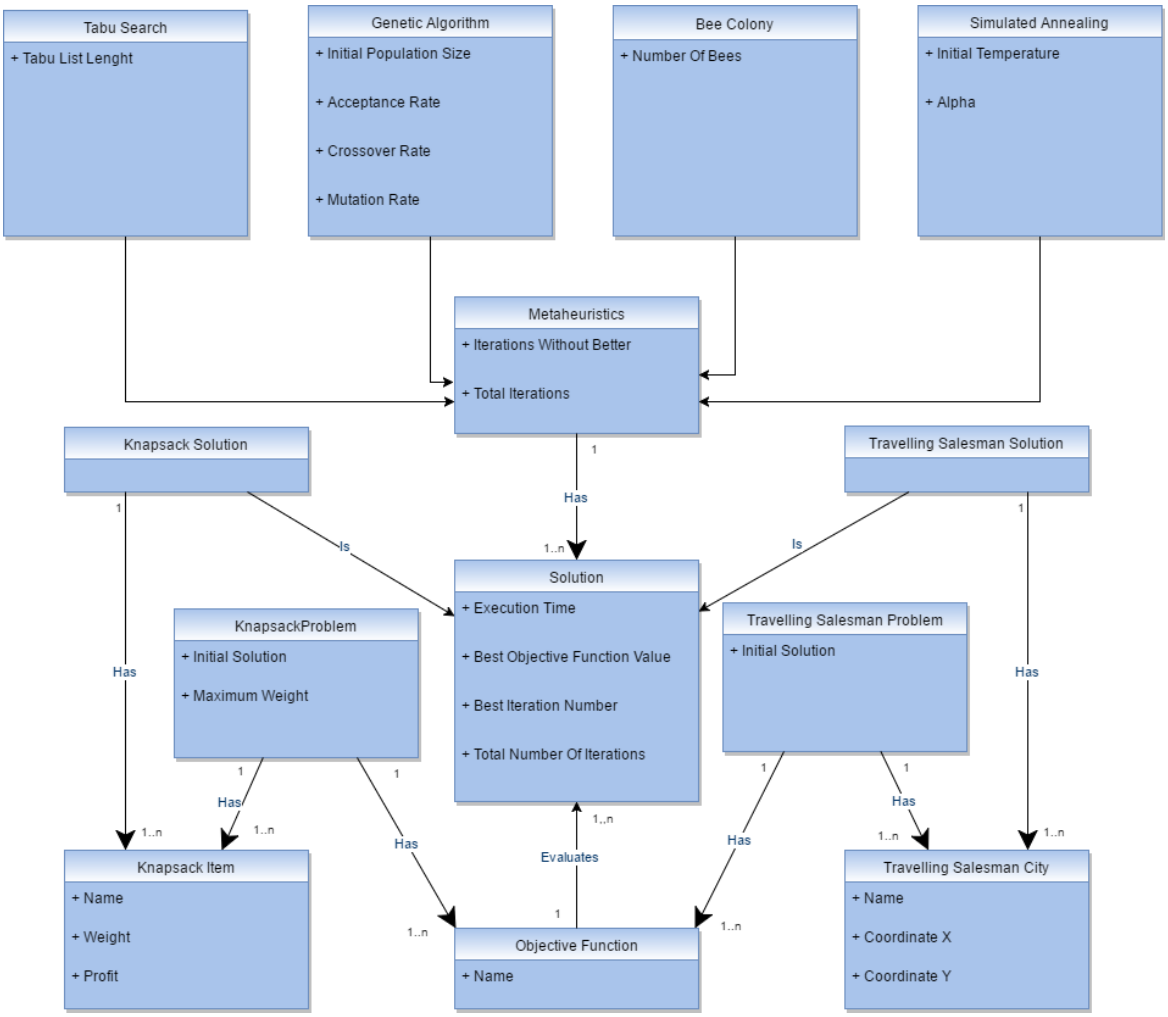

Figure 17 Application Domain Model

Metaheuristics entity is the class dedicated to algorithms implementation. As referred in metaheuristics chapter, algorithms must have a stopping criteria that will define their end. One stopping criteria that can be applied to all algorithms is the total number of iterations (*Total Iterations*) and total number of iterations without a better value for the objective function (Iterations Without Better).
KP entity (*Knapsack Problem*) has items (*Knapsack Item*), which are identified by their *name*, *weight* and *profit*. Metaheuristics algorithm receives these items in order to process the solution, thought which is the first order of the items? This order is defined as *Initial Solution* and will determine if the items will be sort randomly or by their properties, such as profitability or weight in ascending or descending order. KP entity also has an *Objective Function*, which is characterized by a name, which defines minimization or maximization function to be applied.
Similarly to KP entity, there is a TSP entity (*Travelling Salesman Problem*), which has an *Initial* Solution, a set of cities that the salesman should visit and a *Budget*. The cities that can be visited by the salesman, are another identified entity, *Travelling Salesman* City, and is characterized by their Name and their coordinates (*Coordinate X* and *Coordinate Y*) that allow to calculate distances between them. These calculations are made according to an objective function, so *Travelling Salesman Problem* has an *Objective Function*.
As for the solution (*Solution*), it will have an execution time (*Execution Time*) that will allow to infer the computational cost of the algorithm solving the problem. *Best Objective Function Value* reflects the best value found by the algorithm during execution, there are also a record of *Best Iteration* Number (iteration where the best objective function value was found) and total number of iterations, this will allow to understand how many iterations could be avoided for example.
For each problem, there are different types of solutions, so it is needed to separate problem solution into KP Solution (*Knapsack Solution*), which will contain the set of knapsack items included in the knapsack, and TSP Solution (*Travelling Salesman Solution*), that will contain the ordered set of cities to visit.

3.3 User Interface

Lauesen [38] defines user interface as a part of the system seen, heard and felt by the user. It is very important to develop a system that has a good interaction, usability – as ISO 9241 defines, usability is an “extent to which a system, product or service can be used by specified users to achieve specified goals with effectiveness, efficiency and satisfaction in a specified context of use” [39].

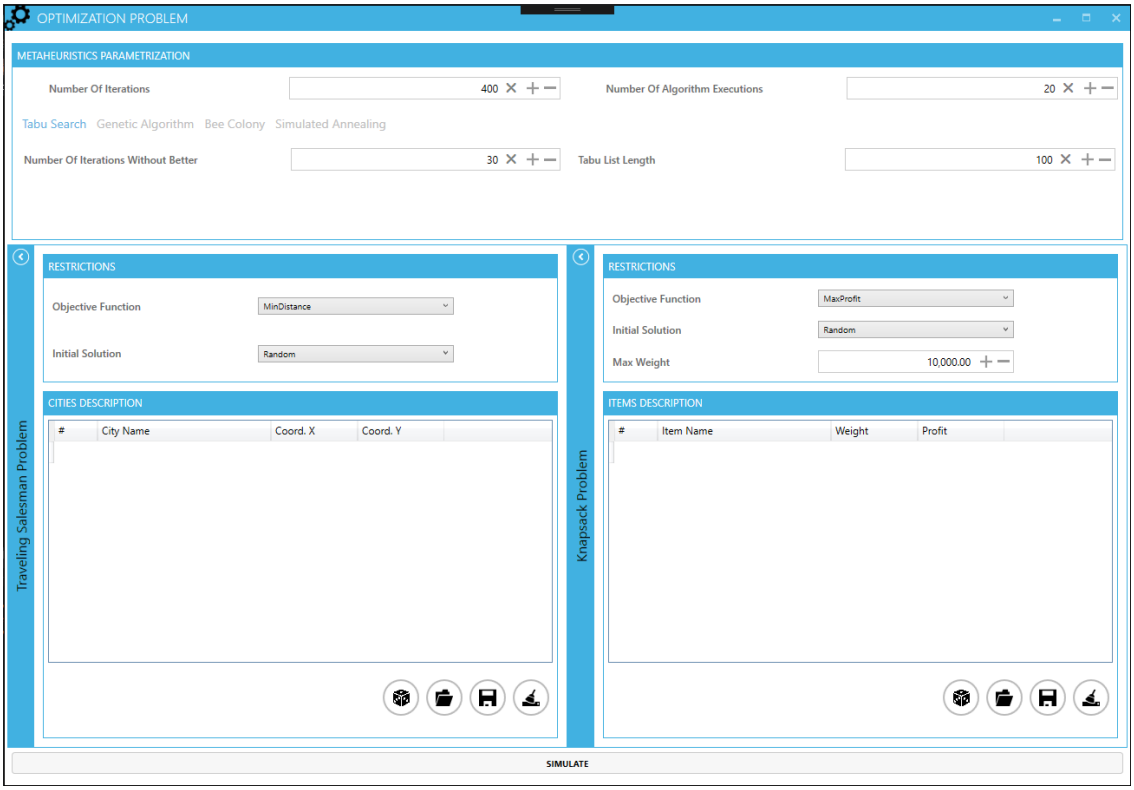

Figure 18 Main Window

User interface project was developed in Windows Presentation Foundation due to “set of application-development features that include Extensible Application Mark-up Language (XAML), controls, data binding, layout, 2-D and 3-D graphics, animation, styles, templates, documents, media, text, and typograph” [40].
This project requires as input user parametrization of algorithms and problems, and as output it should present computational results, so a user interface is needed to allow both actions.
Project main window will be dedicated to parametrization. There are three main areas for parametrization, Metaheuristics Parametrization, TSP Parametrizations and KP Parametrization.
Each algorithm, as represented in the domain model, has specific parameters. These have a dedicated separator in the main screen, “Metaheuristic Parameterization”. In Figure 18 it is possible to see this section. It comprises parametrizations that are linear to all algorithms and also parametrization that is specific of each algorithm.

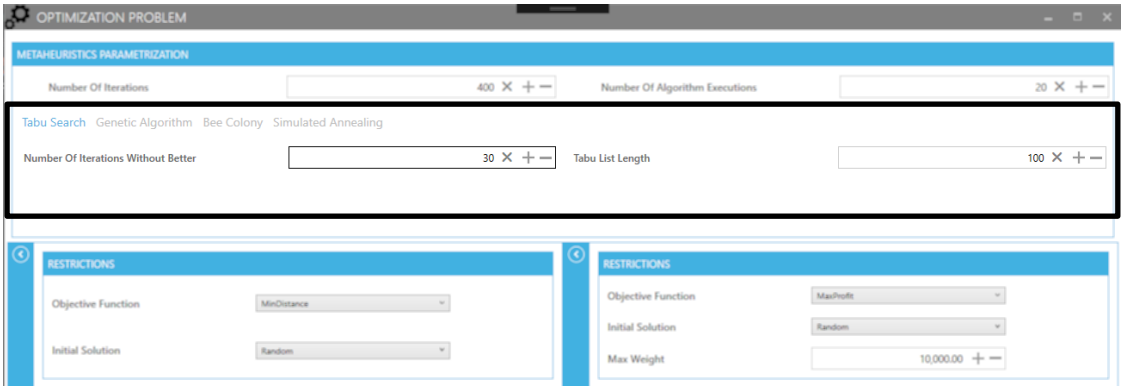

Figure 19 TS Parametrization

Generic parametrization includes “Number of Iterations” of the algorithms until returning the best solution found and, “Number of Algorithm Executions”. Number of algorithms execution is a very important parametrization. Since the project is based into computational results, it is important to be able to have several run results for the same parametrization. This will allow to evaluate algorithm performance in different runs and get an average result, avoiding to infer a conclusion only based in a single run.
To present specific parametrization, a tab menu was included. As showed in Figure 19, TS parametrization includes a numeric box to set the number of iterations without a better and another to set tabu list length.

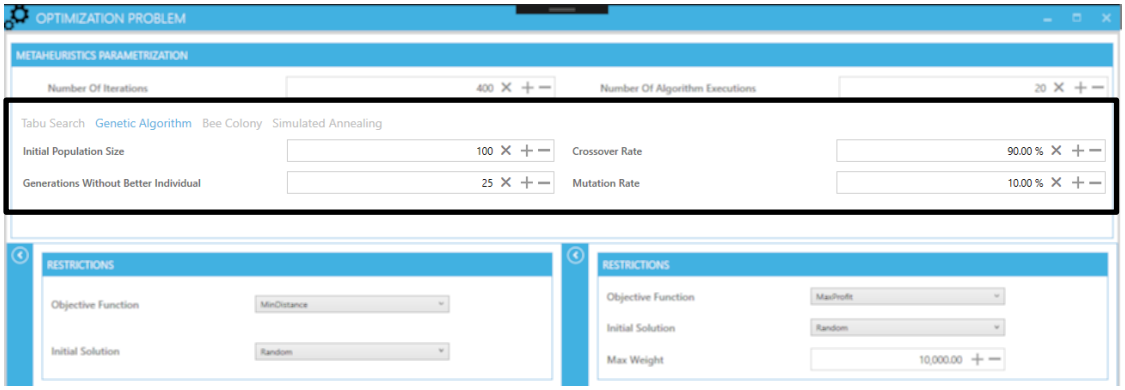

Figure 20 GA Parametrization

Figure 20 presents GA parametrization, which includes a field to set initial population size, percentage fields for acceptance, crossover and mutation rates and finally a second stop criteria, number of generations without a better solution.

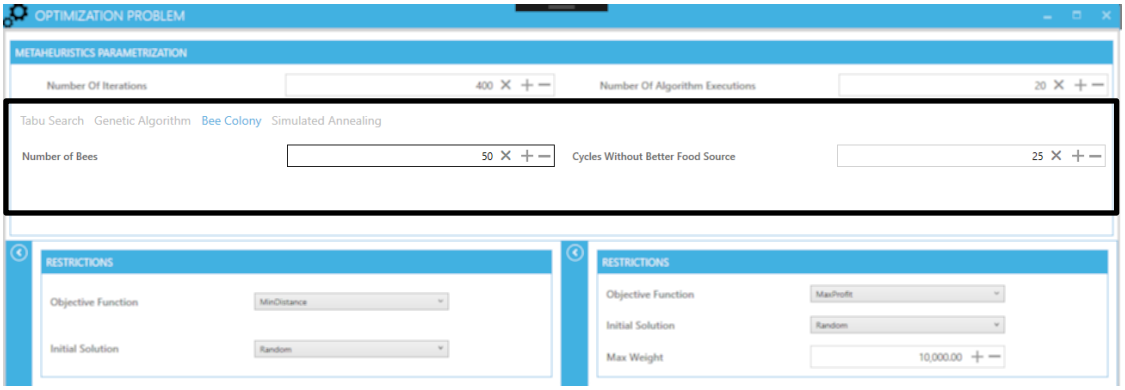

Figure 21 BC Parametrization

BC algorithm has two specific parametrizations, number of bees considered as well as number of cycles without better food source (Figure 21).

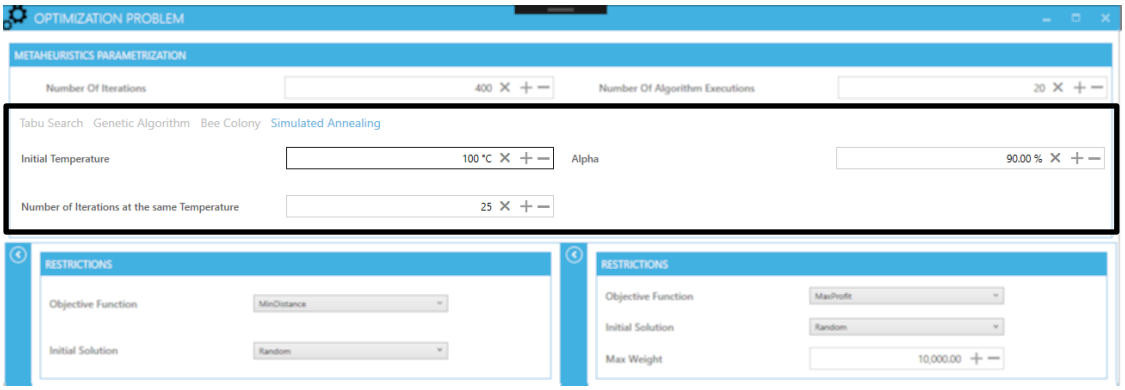

Figure 22 SA Parametrization

Finally, algorithm SA parametrization includes initial temperature, alpha and number of iterations at same temperature, as shown at Figure 22.
As presented in Figure 18 there is an area dedicated to TSP and KP parametrization, which includes dropdown lists for selecting initial solutions and objective function. Initial solution comprises the possibility to generate a specific order to initialize the problem solving process. Particularly, for the KP problem there is another field to set the maximum weight supported by the knapsack.
User is also presented with the possibility to set the list of cities or items manually, upload them from an excel file or generate them randomly by defining only the number of cities/items to generate. In order to maintain some update made to a list or a list created, the user can save the list of cities by selecting the save button.

The “Simulate” button initializes the process. If the user has data in both KP and TSP problem parametrization it will run both; else it will only run the problem parametrized.
All these pieces presented above are part of the metaheuristics and problem settings, when the user runs the program. Results will be presented in a new board (which can be viewed by selecting the flag results centred in the right of the screen).

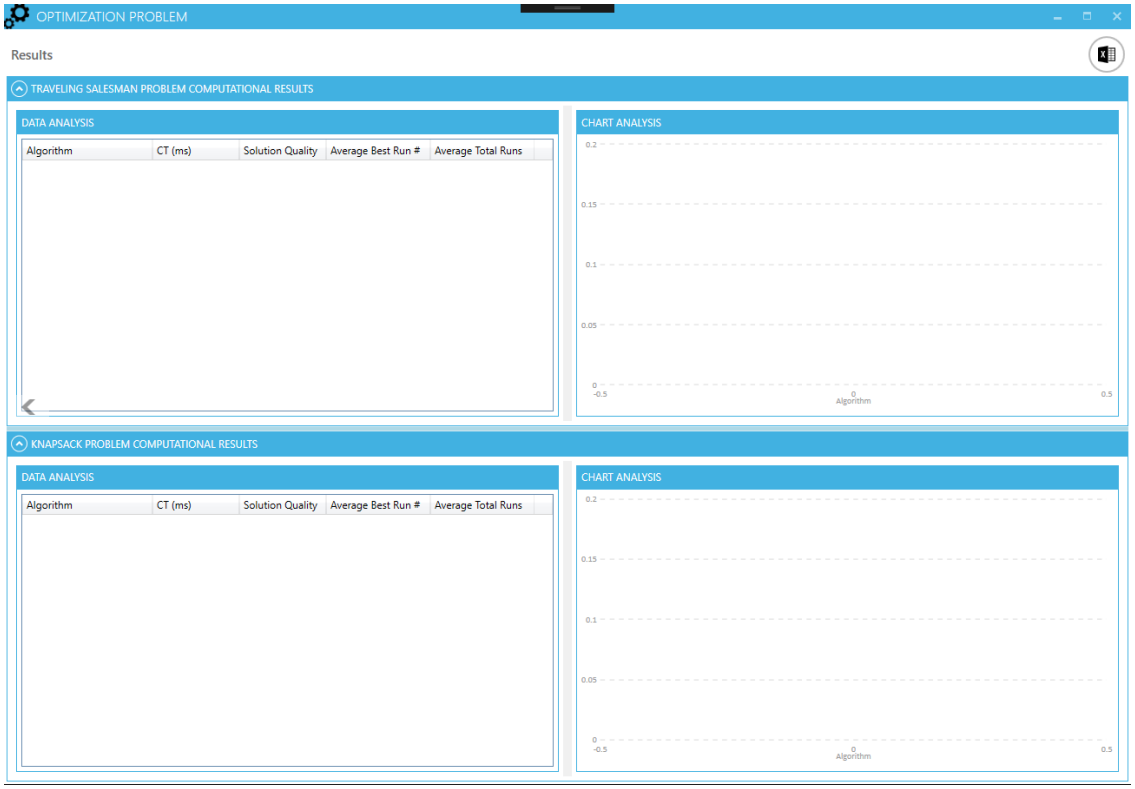

Figure 23 Results Main Window

Considering the possibility to scale the application and add more problems or algorithms to it, the application is organized with tabs and resizable windows, so that enhancements can be easily and simply accommodated.
Results main window, Figure 23, presents the results separated by problem sections. These sections are resizable and collapsible, allowing the user to focus on the pretended area.

DATA ANALYSIS

Algorithm	CT (ms)	Solution Quality	Average Best Run #	Average Total Runs
Simulated Annealing	102	29098.59	18	30
Tabu Search	107.5	28563.98	14	30
Genetic Algorithm	1099	27979.32	10	16
Bee Colony	398	28501.86	1	6

Figure 24 Data Analysis Section

There are several metrics that could be presented, though the focus was to present core fields to allow results analysis and inferences, such as Algorithm name, CT, SQ, average best run number and total number of runs (Figure 24). Results such as best solution found are presented when the user chooses to export results to an excel file by selecting the export to excel button in the right top of the window.

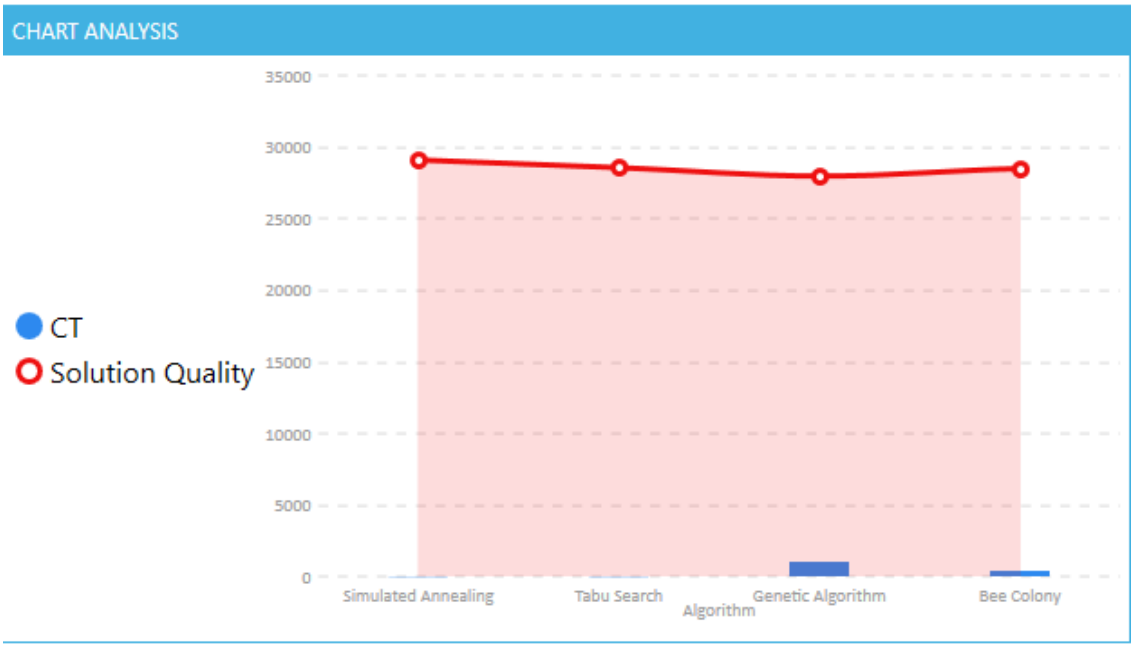

Figure 25 Chart Analysis

A chart analysis is presented (Figure 25) to allow a chart comparison, though this is only a visual enhancement, since the results must be analytically calculated to allow a correct and specific inference.

3.4 Conclusion

In order to solve the presented problem, it is important to obtain metrics about metaheuristics runs applied to the optimization problems selected, both problem and metaheuristics parametrizations are very important as well as results presentation. To achieve this goal, a complete prototype comprising all these variables is of extreme importance.

Over this chapter it was possible to understand how this prototype was developed and its organization. System architecture and design was presented, allowing a better understanding of development details, as well as the developed user interface.

4 Computational Results

Outputs produced by the prototype are only valid if they have meaning. Getting this meaning is done by the analytical and mathematical evaluations of several aspects, and producing results that are comparable among them. Comparing these values will allow inference of some conclusions on how the system performed under a certain context.

It is the goal of this chapter, the exposure and explanation of each of the evaluation aspects and how the mathematical analysis is done, providing a clear and comprehensive overview of the system and, in specific, of how each algorithm performed, by evaluating and matching each with other.

In order to be comparable, evaluation metrics must be obtained in the same environment. This allows to prevent differences due to number of cores, clocks and others.

Execution tests are going to be run in the same computer, a Hewlet-Packard ProBook 650 G1 with an Intel® Core™ i7-4712MQ CPU @ 2.30 GHz processor, 16 GB of memory RAM and Windows 10 Professional as the operative system.

4.1 Evaluation Metrics

The evaluation of metaheuristics will be made by experimentations technique. An application applies metaheuristics to optimization problems, according to some defined stopping criteria, and will collect data samples. Since the goal is to analyse algorithms between them, there is the need to obtain computational results, not necessarily with best results for the problem in hands but the best result achieved by algorithms and best CT. Only through empirical experimentation is possible to obtain metrics and evaluate efficacy, **SQ** (objective functions were defined in section 2.2.3 and 2.2.4)**,** and efficiency, **CT**.

In this document, evaluation will be made by SQ and also by weights application to SQ and CT. SQ will value 60% and CT 40% because it is very important to find solutions quickly but it is more important to find better solutions. That is why these proportions are so close, benefiting SQ.

As for SQ, it is not enough to use only the best solution evaluation, so the solution in evaluation will be the average solution among all executions of the algorithms.

$$0.40 * \mathrm{CT} + 0.60 * \frac{\sum_{i=1}^{n} r_n}{n} \quad (6)$$

Equation 6 will be applied, and will be called as hybrid, where CT is represented in milliseconds, and SQ, being the average of objective function results, is represented by the sum of all solution results (*r*) obtained in *n* executions.
Since there is an implementation of two optimization problems, for each problem, it will be analysed each of the hypothesis:
- H0: All algorithms are equal
- H1: There is at least one algorithm that steps out with better solutions

Algorithms performance analysis implies some executions in order to obtain metrics able to infer conclusions about results. Having this in mind, results tend to a normal distribution as the number of observations increase. Since the aim is to compare a group of four algorithms, ANOVA (Analysis of Variance Methods) is the most appropriate test to use.
This study will include five benchmark problem instances of each TSP and KP, with different number of cities/items.

4.2 Parametrizations

As already approached on Chapter 2, parametrizations are very important when evaluating performance and can be decisive when determining algorithms success.

Table 12 TSP Algorithms Parametrization

Metaheuristics	***Parameter***	***Value***
TS	*Number of Iterations Without Better*	*100*
	Tabu Search List Length	*30*
GA	*Generations Without Better Individual*	*25*
	Initial Population Size	*100*
	Crossover Rate	*90%*
	Mutation Rate	*10%*
SA	*Number of Iterations at Same Temperature*	*25*
	Initial Temperature	*100*
	Alpha	*90%*
BC	*Number of Iterations without Better*	*25*
	Number of Bees	*50*
	Number of Food Sources	*50%*Number Of Bees*

TSP computational results will have on their basis parametrization defined in Table 12. In order to define the presented parametrization, several experiments were made and these were the parameters with better results to the majority of the travelling salesman analysed executions.

Algorithms will share stopping criteria, each algorithm will not exceed 400 iterations, and each algorithm will be ran 20 times independently in order to obtain a mean value for each result.
As already stated, KP goal is to select a subset from a group of items to get together inside a knapsack maximizing their profit and respecting knapsack weight limitations. The benchmark problems for this problem comprises less number of items when compared to TSP problem number of cities. Having this in mind, urges the necessity of setting different parametrization.

Table 13 Algorithms Parametrization

Metaheuristics	***Parameter***	***Value***
TS	*Number of Iterations Without Better*	*200*
	Tabu Search List Length	*3*
GA	*Generations Without Better Individual*	*50*
	Initial Population Size	*50*
	Crossover Rate	*90%*
	Mutation Rate	*10%*
SA	*Number of Iterations at Same Temperature*	*200*
	Initial Temperature	*100*
	Alpha	*95%*
BC	*Number of Iterations without Better*	*50*
	Number of Bees	*25*
	Number of Food Sources	*50%*Number Of Bees*

After experimental tuning, parametrization set to analyse this KP instance was the presented in Table 13. Similarly to TSP, KP algorithms will share a stopping criteria, each algorithm should not exceed 3000 iterations, and each algorithm will be ran 20 times independently in order to obtain a mean value for each result.

4.3 Test Results

This section gathers all results obtained by executing the developed application with the previously defined constraints and stopping criteria.
As already approached in the previous section, the instances of the chosen problems are from benchmark problems (“sets of benchmark problem instances allow researchers to compare the efficiency of their optimization solvers in common, usually publicly available, problem sets” [41]) and are presented in the appendix section.

Since the goal is to compare algorithms performance between them, getting optimal solutions for these problems is not relevant to this research.

Table 14 TSP Computational Results

Algorithm		Att48	Berlin52	KroE110	Tsp225	Att532
TS	***CT***	261,05	190,05	472,10	181,10	635,2
	SQ	147300,54	22169,54	166674,93	10299,90	977407,79
	Hybrid	88484,74	13377,74	100193,80	6252,38	586698,75
SA	***CT***	551,4	229,35	670,10	164,40	659,7
	SQ	140262,70	21803,24	169021,07	10290,74	975239,85
	Hybrid	84378,18	13173,68	101680,68	6240,20	585407,79
BC	***CT***	7714,75	8136,9	7914,95	8311,50	26761,2
	SQ	146484,26	21472,52	180029,41	10264,50	971687,84
	Hybrid	90976,46	16138,27	111183,63	9483,30	593717,18
GA	***CT***	16331	3100,9	34604,10	2954,10	11147,35
	SQ	130369,36	21968,10	155644,07	10299,90	976588,24
	Hybrid	84754,02	14421,22	107228,08	7361,58	590411,88

Table 14 summarizes TSP mean values of the 20 parametrized runs. Columns represent the problem and rows represent computational results for CT, SQ and hybrid version (formula from section 4.1 applied), per algorithm.

Table 15 KP Computational Results

Algorithm		P01	P02	P05	P07	P08
TS	***CT***	0,4	0,3	0,6	1	2,3
	SQ	171	24	35	392,2	3288430,3
	Hybrid	102,76	14,52	21,24	235,72	1973059,1
SA	***CT***	0,05	0,1	0,1	0,35	1,05
	SQ	170,8	24	35	392,05	3277544,55
	Hybrid	102,5	14,44	21,04	235,37	1966527,15

BC	***CT***	8,45	5,35	7	11,95	28,45
	SQ	170,60	24	35	391,75	3303290,05
	Hybrid	105,74	16,54	23,8	239,83	1981985,41
GA	***CT***	19,85	11,35	22,4	66,65	257,85
	SQ	171	24	35	393,25	3307225,85
	Hybrid	110,54	18,94	29,96	262,61	1984438,65

Table 15 presents obtained results when running algorithms against KP problem.

4.3.1 SQ Analysis

SQ corresponds to the result from the minimization, as already explained in previous sections algorithms may not find the best solution possible but finds an acceptable solution in the available time.

Table 16 TSP SQ Summary

Algorithm	Number of instances	Mean	St. Dev	95% CI
BC	5	265988	401517	(-117150; 649125)
GA	5	258974	406263	(-124164; 642112)
SA	5	263324	404110	(-119814; 646461)
TS	5	264771	404625	(-118367; 647908)

Table 17 KP SQ Summary

Algorithm	Number of instances	Mean	St. Dev	95% CI
BC	5	660782	1477207	(-735806; 2057371)
GA	5	661570	1478967	(-735019; 2058159)
SA	5	655633	1465693	(-740955; 2052222)
TS	5	657811	1470561	(-738778; 2054399)

Table 16 summarizes each algorithm values for TSP, while

Table 17 presents KP values. Summary metrics include the number of different instances ran per algorithm, mean values of all solutions obtained, standard deviation and interval with 95% of confidence.

Table 18 TSP SQ Analysis of Variance

Source	DF	SS	MS	F-Value	P-Value
Algorithms	3	140482230	46827410	0,00	0,999
Error	16	2,61317E+12	1,63323E+11		
Total	19	2,61331E+12			

Table 19 KP SQ Analysis of Variance

Source	DF	SS	MS	F-Value	P-Value
Algorithms	3	112598972	37532991	0,00	0,999
Error	16	3,47212E+13	2,17007E+12		
Total	19	3,47213E+13			

The main question is, are these results equivalent? ANOVA analysis are presented in Table 18 for TSP and in Table 19 for KP, in which:

- “SS” means “Sum of Squares” and “represents a measure of variation or deviation from the mean. It is calculated as a summation of the squares of the differences from the mean. The calculation of the total sum of squares considers both the sum of squares from the factors and from randomness or error” [42]
- “df” stands for “degrees of freedom” and represent the information in samples, more samples will increase df, and less samples will decrease df values. Total df is given by N-1, in this case, since there are 20 samples, it would be 19. Total df equals the sum of df between groups (number of algorithms - 1) and within groups (N-number of algorithms).
- “MS” corresponds to “Mean Squares” and “represent an estimate of population variance. It is calculated by dividing the corresponding sum of squares by the degrees of freedom.” [42]
- “F” represents the ratio between MS between groups and MS within groups and indicates if results are or not significant
- “P-Value” is a probability and is used to measure evidence against the null hypothesis
- “F-crit” is the critical values for the “F” distribution with defined alpha, in this case is 0.05

It is possible to conclude that, in neither of the instances, p-value is less than defined alpha of 0.05, since in both problems p-value assumes value near 1, so there is not significant difference that allows to reject that all algorithms are equal. There is not a single algorithm that steps up between the groups of four algorithms analysed when considering their SQ.

4.3.2 Hybrid Analysis

Table 20 TSP Hybrid Summary

Algorithm	Number of instances	Mean	St. Dev	95% CI
BC	5	164300	244191	(-66453; 395053)
GA	5	160835	244023	(-69918; 391588)
SA	5	158176	242526	(-72577; 388929)
TS	5	159001	242845	(-71751; 389754)

Table 21 KP Hybrid Summary

Algorithm	Number of instances	Mean	St. Dev	95% CI
BC	5	396474	886328	(-441490; 1234438)
GA	5	396972	887421	(-440992; 1234936)
SA	5	393380	879416	(-444584; 1231344)
TS	5	394687	882337	(-443277; 1232651)

Table 20 and Table 21 summarizes each algorithm values for TSP and KP, respectively.

Table 22 TSP Hybrid Analysis of Variance

Source	DF	SS	MS	F-Value	P-Value
Algorithms	3	110861464	36953821	0,00	0,999
Error	16	9,47876E+11	59242236511		
Total	19	9,47987E+11			

Table 23 KP Hybrid Analysis of Variance

Source	DF	SS	MS	F-Value	P-Value
Algorithms	3	41063105	13687702	0,00	0,999
Error	16	1,24999E+13	7,81246E+11		
Total	19	1,25000E+13			

Analysing Table 22 and Table 23 values, similarly to SQ results, P-value is near 1, so p-value is higher than alpha of 0.05, so is not possible to discard H0. It is not possible to conclude that algorithms results, considering hybrid combination of 40% of CT and 60% of SQ, are not equal and that there is an algorithm that has better results than the other because null hypothesis cannot be discarded.

4.4 Conclusion

This chapter firstly presents an introduction referring the importance of achieving measurable and comparable results, and presenting execution environment.
Evaluation metrics were exposed, explaining which metrics and statistical model were used to evaluate results.
The last section is dedicated to analyse results obtained presenting a synthesis of all achieved means and two chapters, one dedicated to SQ results analysis and a section with hybrid analysis.

5 Conclusions and Implications

This document presented optimization problems and a specific solving approach, metaheuristics.

Metaheuristics are a valuable tool to solve optimization problems. Besides the four considered in this document, which comprised TS, SA, BC and GA, there is a wide range of others and there are still many incoming.

Metaheuristics parameter tuning and control is, by itself, an optimization problem because if comprises a group of variables that lead to an increase or decrease of SQ and CT. This parameterization problem was synthetically presented along this document as well as their impact on the final solution.

Since this problem is widely studied and approached, the state of the art section presented some metaheuristics performance studies and their conclusions. A value analysis was made and transmits the importance of finding an algorithm that is the best in optimization problem resolution, also as the contribution this paper can bring to industry and scientific community.

Having as aim demonstrating if it was possible or not to select one of four metaheuristics to solve optimization problems more efficiently and gathering best solutions, a prototype was assembled to allow results gathering and analysis. In this document it is possible to find the system context and execution, gathering information about architecture and design decisions as well as user interface presentation.

Once performance is not the only metric available and important to make decisions (there can be an algorithm highly performant that provides the worst solution possible) there is a section dedicated to explain all metrics used, SQ and a combination of SQ with CT, and also present statistical model used to take inferences and answer the problem in hands.

Problems were separated in two, due to their singularity, and results were presented and analysed with ANOVA statistical model applied to efficacy, efficiency and a combination of both.

5.1 Achievements

It was not possible to prove either that all algorithms are equivalent or that some of them is better in the majority of the cases, considering both problems. It is only possible to state that CT of the TSP problem is significantly different between algorithms, being TS in the top of performance. This can be explained because, ignoring SQ, TS is the simplest algorithm of the four algorithms considered. Though, when applying the hybrid analysis formula, algorithms results are very similar.

It is important to refer that either problem or parametrization influence results. Different parametrization can lead better or worst results. Having this in mind, is not possible to confirm neither if the algorithms are equivalent or not.

5.2 Limitations and Further Research

This document approached only four algorithms and two optimization problems. This range could be enlarged including other algorithms and more optimization problems. Having in mind this problem investigation evolution, user interface was conceived with additional flexibility in order to allow these new algorithms and problems incorporation (as referred in section 3.3).
Parameter tuning and control has an enormous influence in metaheuristics performance. Since parameter definition was made through experimental trials, it is important to study other parameters combination and also their impact in the results.

5.3 Final Appreciation

Embracing this challenge was very demanding because it required dedication and self-organization in order to combine research with work and family responsibilities and dedication at the same time, though at the end it was very positive because it allowed a knowledge enrichment and evolution.
It was very rewarding to carry out this research and overcome difficulties, it allowed a significant growth and arousing of interest in this area and their involvement.

6 Bibliography

[1] M. Chiarandini, L. Paquete, M. Preuss and E. Ridge, "Experiments on Metaheuristics: Methodological," [Online]. Available: https://www.cs.ubc.ca/~hutter/EARG.shtml/earg/papers08/ChiEtAl07.pdf. [Accessed 09 2017].

[2] C. H. Papadimitriou and K. Steiglitz, Combinatorial Optimization, New York: Wiley, 1982.

[3] E.-G. Talbi, Metaheuristics from design to implementation, University of Lille – CNRS – INRIA: y John Wiley & Sons, Inc., Hoboken, New Jersey, 2009.

[4] R. A. Sarker and C. S. Newton, Optimization Modelling: A Practical Approach, CRC Press, 2007.

[5] A. Madureira, "Aplicação de Meta-Heurísticas ao Problema de Escalonamento em Ambiente Dinâmico de Produção Discreta," (PhD thesis), Universidade do Minho, Braga, 2003.

[6] I. A. S. Pereira, "Aspectos de Aprendizagem em Optimização," ISEP, Porto, 2009.

[7] Wikipedia, "Travelling salesman problem," [Online]. Available: https://en.wikipedia.org/wiki/Travelling_salesman_problem. [Accessed 10 2016].

[8] S. Dasgupta, C. Papadimitriou and U. Vazirani, Algorithms, McGraw-Hill Education; 1 edition, 2006.

[9] G. Zäpfel, R. Braune and R. Braune, Metaheuristic Search Concepts, Springer-Verlag Berlin Heidelberg, 2010.

[10] G. Zäpfel and R. Braune, Metaheuristic Search Concepts: A Tutorial with Applications to Production and Logistics, Springer Science & Business Media, 2010.

[11] A. B. G. S. Santos, "Análise de Desempenho de Técnicas de Optimização no problema de escalonamento," Instituto Superior de Engenharia do Porto, Porto, 2015.

[12] L. W. Miller, R. W. Conway and W. L. Maxwell, Theory of Scheduling, (Reprint Edition)Dover Books of Computer Science, Dover Publications, 1967.

[13] I. A. S. Pereira, Sistema Inteligente para Escalonamento Assistido por Aprendizagem, Vila Real: Universidade de Trás-os-Montes e Alto Douro, 2014.

[14] F. Glover, "Future paths for integer programming and links to artificial intelligence," *Comput. Oper. Res.,* pp. Volume 13, Issue 5, Pages 533-549, 1986.

[15] M. Pirlot, General Local Search Method, European Journal of Operational Research, 1996.

[16] F. Glover, M. Laguna and R. Marti, "Principles of Tabu Search," Universitat de València.

[17] J. Brownlee, Clever Algorithms: Nature-Inspired Programming Recipes, Melbourne: Lulu, 2012.

[18] V. Ryan, "Annealing Metals," World Association of Technology Teachers, [Online]. Available: http://www.technologystudent.com/equip1/heat3.htm. [Accessed 09 2017].

[19] C. Blum and A. Roli, "Metaheuristics in combinatorial optimization: Overview and conceptual comparison", ACM Computing Surveys, 2003.

[20] M. Silva, D. Falcão and I. Silva, "Meta-heurísticas -Análise computacional em duas plataformas para a resolução do problema HT em Máquina Única," ISEP, Porto.

[21] K. E. Parsopoulos and M. N. Vrahatis, Particle Swarm Optimization and Intelligence: Advances and Applications, New York: IGI Global, 2010.

[22] M. M. Millonas, Swarms, Phase Transitions, and Collective Intelligence, Santa Fe: Santa Fe Institute, 1993.

[23] D. Karaboga, "An Idea Based on Honey Bee Swarm for Numerical Optimization," Erciyes University, 2005.

[24] J. H. Holland, Adaptation in Natural and Artificial Systems, Michigan: University of Michigan Press, 1992.

[25] M. Mitchell, "Genetic Algorithms: An Overview," Santa Fe Institute, Santa Fe, 1995.

[26] J. McCullock, "Mutation," [Online]. Available: http://mnemstudio.org/genetic-algorithms-mutation.htm. [Accessed 20 11 2016].

[27] D. H. Wolpert and W. G. Macready, "No Free Lunch Theorems for Optimization," *IEEE Transactions on Evolutionary Computation,* pp. VOL. 1, NO. 1, 4 1997.

[28] A. E. Eiben and J. E. Smith, Introduction to Evolutionary Computing: Natural Computing Series, Springer, 2003.

[29] C. Neumüller, "Parameter Meta-Optimization of Metaheuristic: Optimization Algorithms," Fachhochschul - Masterstudiengang, Austria, 2011.

[30] B. H. Arabi, "Computer Modelling and Simulation (UKSim)," in *UKSim-AMSS 18th International Conference*, 2016.

[31] M. Antosiewicz, G. Koloch and B. Kamińsk, "Choice of best possible metaheuristic algorithm for the travelling salesman problem with limited computational time: quality, uncertainty and speed," *Journal of Theoretical and Applied Computer Science,* vol. Vol. 7, no. 1, pp. 46-55, 2013.

[32] K. Li, L. Kang and W. Zhang, "Comparative Analysis of Genetic Algorithm and Ant Colony Algorithm on Solving Traveling Salesman Problem," in *Semantic Computing and Systems*, 2008.

[33] G. Reinelt, "TSPLIB," Institut für Informatik, [Online]. Available: http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/. [Accessed 07 2017].

[34] P. A. Koen, H. M. J. Bertels and E. Kleinschmidt, "Managing the Front End of Innovation—Part I: Results From a Three-Year Study," Research-Technology Management, 2004, pp. 34-43.

[35] P. Koen and e. al, "Providing clarity and a common language to the ‘fuzzy front end," 2001.

[36] P. Koen, "FEI," [Online]. Available: http://frontendinnovation.com/fei. [Accessed 01 2017].

[37] Strategyzer AG, "Strategyzer," 2017. [Online]. Available: https://strategyzer.com/. [Accessed 01 2017].

[38] S. Lauesen, User Interface Design: A Software Engineering Perspective, Essex: Pearson Education Limited, 2005.

[39] International Organization for Standardization, "ISO," 2010. [Online]. Available: https://www.iso.org/obp/ui/#iso:std:52075:en. [Accessed 09 2017].

[40] Microsoft, "Introduction to WPF," [Online]. Available: https://msdn.microsoft.com/en-us/library/aa970268(v=vs.100).aspx. [Accessed 01 2017].

[41] A. Sifaleras, Classification of Network Optimization Software Packages, Pennsylvania: IGI Global, 2015.

[42] Minitab Inc., "Minitab 17 Support," [Online]. Available: http://support.minitab.com/en-us/minitab/17/topic-library/topic-library-overview/. [Accessed 06 2017].

[43] Microsoft, "MSDN," 26 12 2008. [Online]. Available: http://msdn.microsoft.com/en-us/library/z2kcy19k(v=vs.80).aspx. [Accessed 08 07 2012].

[44] R. Anthony, Planning and Control Systems: A Framework for Analysis, Harvard: Harvard Business School Div. of Research, 1965.

[45] Microsoft, "When to Use Interfaces," 2012. [Online]. Available: http://msdn.microsoft.com/en-us/library/3b5b8ezk(v=vs.80).aspx. [Accessed 09 07 2012].

[46] C. I. Christodoulou, C. S. Pattichis and W. F. Fincham, "A Modular Neural Network Decision Support System in EMG Diagnosis," 1998.

[47] M. J. Druzdzel and R. R. Flynn, "Decision Support Systems," 2000.

[48] C. Holsapple and A. B. Whinston., Decision Support Systems: A Knowledge-Based Approach, St. Paul: St. Paul: West Publishing, 1996.

[49] R. D. Hackathorn and P. G. W. Keen, "Organizational Strategies for Personal Computing in Decision Support Systems.", MIS Quarterly, 1981.

[50] P. D. N. S. d. A. Ferreira, "As pesquisas denominadas "Estado da Arte"," [Online]. Available: http://www.fe.unicamp.br/alle/textos/NSAF-AsPesquisasDenominadasEstadodaArte.pdf. [Accessed 15 08 2014].

[51] I. E. T. R. d. Silva, "ME-DGT - Manufacturing Engineering Data Generation Tool," ISEP, Porto, 2011/2012.

[52] K. Baker, Introduction to sequencing and scheduling, New York: Wiley, 1974.

[53] J. Blazewicz, K. H. Ecker, E. Pesch, G. Smith and J. Weglarz, Scheduling Computer, New York: Springer, 2001.

[54] S. French, Sequencing and Scheduling: An introduction to the Mathematics of the, Chichester: Ellis Horwood, 1982.

[55] A. Madureira, Técnicas Emergentes de Optimização no Suporte à Tomada de Decisão, Porto: Instituto Superior de Engenharia do Porto, 2010.

[56] M. L. Pinedo, Scheduling: Theory, Algorithms, and Systems, New York: Springer-Verlag, 2012.

[57] M. L. R. Varela, "Uma Contribuição para o Escalonamento da Produção baseado em Métodos Globalmente Distribuídos," (PhD thesis), Universidade do Minho, Braga, 2007.

[58] J. A. V. d. C. P. Claro, "Uma Abordagem Orientada por Objectos para Meta-Heurísticas Multiobjectivo," Faculdade de Engenharia da Universidade do Porto, Porto, 2002.

[59] A. Osterwalder and Y. Pigneur, An ontology for e-business, Butterworth-Heinemann, 2003.

[60] A. Osterwalder and Y. Pigneur, Business Model Generation: Inovação em Modelos de Negócios, Rio de Janeiro: Alta Books, 2011.

[61] F. Glover and G. A. Kochenberger, Handbook of Metaheuristics, New York, Boston, Dordrecht, London, Moscow: Kluwer Academic Publishers, 2003.

[62] X.-S. Yang, Nature-Inspired Computation in Engineering, Springer, 2016.

[63] D. Karaboga and B. Basturk, "A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm," *Journal of Global Optimization,* vol. 39, no. 3, p. 459–471, 2007.

[64] S. Lawrence, Resource constrained project scheduling: an experimental investigation of heuristic scheduling techniques, Pennsylvania: Graduate School of Industrial Administration. Carnegie-Mellon University, 1984.

[65] J. B. E. &. Z. D. Adams, "The shifting bottleneck procedure for job shop scheduling," *Management Science,* no. 34, pp. 391-401, 1988.

[66] E. Alonso, M. D. Inverno, D. Kudenko, M. Luck and J. Noble, "Learning in Muli-Agent Systems," in *The Knowledge Engineering Review*, vol. 3, Cambridge, Cambridge University Press, 2001, pp. 277-284.

[67] D. Applegate and W. Cook, "A computational study of the job-shop scheduling instance," *ORSA Journal on Computing,* vol. 3, pp. 149-156, 1991.

[68] R. H. Storer, S. Wu and R. Vaccari, "New search spaces for sequencing problems with application to job shop scheduling," vol. 38, pp. 1495-1509, 1992b.

[69] D. F. M. Falcão, "Híper-Heurísticas com Aprendizagem," Instituto Superior de Engenharia do Porto, Porto, 2014.

[70] I. Pereira and A. Madureira, "Self-Optimization module for Scheduling using Case-based Reasoning," *Applied Soft Computing,* pp. 1419-1432, 2013.

[71] H. Fisher and G. L. Thompson, "Probabilistic learning combinations of local Job Shop scheduling rules," in *Industrial Scheduling*, Prentice Hall, 1963, pp. 225-251.

[72] G. Salvendy, Handbook of Industrial Engineering: Technology and Operations Management, New York: John Wiley & Sons, Inc., 2001.

[73] A. M. D. Madureira, Aplicações de Meta-Heurísticas em Problemas de Sequenciamento, Porto: Faculdade de Engenharia da Universidade do Porto, 1995.

[74] Microsoft, ".NET Development," [Online]. Available: https://msdn.microsoft.com/library/w0x726c2(v=vs.110).aspx. [Accessed 01 2017].

[75] Microsoft, "ASP.NET," [Online]. Available: http://www.asp.net/. [Accessed 05 2012].

[76] Source Making, "Design Patterns," [Online]. Available: https://sourcemaking.com/design_patterns. [Accessed 01 2017].

[77] Dictionary.com, "Dictionary," [Online]. Available: http://dictionary.reference.com/browse/database. [Accessed 07 2012].

[78] Game Theory Lab, CSA, "Optimal Solution for TSP using Branch and Bound," [Online]. Available: http://lcm.csa.iisc.ernet.in/dsa/node187.html. [Accessed 10 2016].

7 Appendix

7.1 TSP Benchmark Problems

Problem Instance	City	Coord X	Coord Y
Att48	1	6734	1453
	2	2233	10
	3	5530	1424
	4	401	841
	5	3082	1644
	6	7608	4458
	7	7573	3716
	8	7265	1268
	9	6898	1885
	10	1112	2049
	11	5468	2606
	12	5989	2873
	13	4706	2674
	14	4612	2035
	15	6347	2683
	16	6107	669
	17	7611	5184
	18	7462	3590
	19	7732	4723
	20	5900	3561
	21	4483	3369
	22	6101	1110
	23	5199	2182
	24	1633	2809
	25	4307	2322
	26	675	1006
	27	7555	4819
	28	7541	3981
	29	3177	756
	30	7352	4506
	31	7545	2801
	32	3245	3305
	33	6426	3173
	34	4608	1198

Problem Instance	City	Coord X	Coord Y
	35	23	2216
	36	7248	3779
	37	7762	4595
	38	7392	2244
	39	3484	2829
	40	6271	2135
	41	4985	140
	42	1916	1569
	43	7280	4899
	44	7509	3239
	45	10	2676
	46	6807	2993
	47	5185	3258
	48	3023	1942
Berlim52	1	565	575
	2	25	185
	3	345	750
	4	945	685
	5	845	655
	6	880	660
	7	25	230
	8	525	1000
	9	580	1175
	10	650	1130
	11	1605	620
	12	1220	580
	13	1465	200
	14	1530	5
	15	845	680
	16	725	370
	17	145	665
	18	415	635
	19	510	875
	20	560	365
	21	300	465
	22	520	585
	23	480	415
	24	835	625
	25	975	580
	26	1215	245
	27	1320	315

Problem Instance	City	Coord X	Coord Y
	28	1250	400
	29	660	180
	30	410	250
	31	420	555
	32	575	665
	33	1150	1160
	34	700	580
	35	685	595
	36	685	610
	37	770	610
	38	795	645
	39	720	635
	40	760	650
	41	475	960
	42	95	260
	43	875	920
	44	700	500
	45	555	815
	46	830	485
	47	1170	65
	48	830	610
	49	605	625
	50	595	360
	51	1340	725
	52	1740	245
KroE100	1	3477	949
	2	91	1732
	3	3972	329
	4	198	1632
	5	1806	733
	6	538	1023
	7	3430	1088
	8	2186	766
	9	1513	1646
	10	2143	1611
	11	53	1657
	12	3404	1307
	13	1034	1344
	14	2823	376
	15	3104	1931
	16	3232	324

Problem Instance	City	Coord X	Coord Y
	17	2790	1457
	18	374	9
	19	741	146
	20	3083	1938
	21	3502	1067
	22	1280	237
	23	3326	1846
	24	217	38
	25	2503	1172
	26	3527	41
	27	739	1850
	28	3548	1999
	29	48	154
	30	1419	872
	31	1689	1223
	32	3468	1404
	33	1628	253
	34	382	872
	35	3029	1242
	36	3646	1758
	37	285	1029
	38	1782	93
	39	1067	371
	40	2849	1214
	41	920	1835
	42	1741	712
	43	876	220
	44	2753	283
	45	2609	1286
	46	3941	258
	47	3613	523
	48	1754	559
	49	2916	1724
	50	2445	1820
	51	3825	1101
	52	2779	435
	53	201	693
	54	2502	1274
	55	765	833
	56	3105	1823
	57	1937	1400

Problem Instance	City	Coord X	Coord Y
	58	3364	1498
	59	3702	1624
	60	2164	1874
	61	3019	189
	62	3098	1594
	63	3239	1376
	64	3359	1693
	65	2081	1011
	66	1398	1100
	67	618	1953
	68	1878	59
	69	3803	886
	70	397	1217
	71	3035	152
	72	2502	146
	73	3230	380
	74	3479	1023
	75	958	1670
	76	3423	1241
	77	78	1066
	78	96	691
	79	3431	78
	80	2053	1461
	81	3048	1
	82	571	1711
	83	3393	782
	84	2835	1472
	85	144	1185
	86	923	108
	87	989	1997
	88	3061	1211
	89	2977	39
	90	1668	658
	91	878	715
	92	678	1599
	93	1086	868
	94	640	110
	95	3551	1673
	96	106	1267
	97	2243	1332
	98	3796	1401

Problem Instance	City	Coord X	Coord Y
	99	2643	1320
	100	48	267
	1	155.42	150.65
	2	375.92	164.65
	3	183.92	150.65
	4	205.42	150.65
	5	205.42	171.65
	6	226.42	171.65
	7	226.42	186.15
	8	226.42	207.15
	9	226.42	235.65
	10	226.42	264.15
	11	226.42	292.65
	12	226.42	314.15
	13	226.42	335.65
	14	205.42	335.65
	15	190.92	335.65
	16	190.92	328.15
	17	176.92	328.15
	18	176.92	299.65
	19	155.42	299.65
Tsp225	20	155.42	328.15
	21	155.42	356.65
	22	183.92	356.65
	23	219.42	356.65
	24	240.92	356.65
	25	269.42	356.65
	26	290.42	356.65
	27	387.42	136.15
	28	318.92	356.65
	29	318.92	335.65
	30	318.92	328.15
	31	318.92	299.65
	32	297.92	299.65
	33	290.42	328.15
	34	290.42	335.65
	35	297.92	328.15
	36	254.92	335.65
	37	254.92	314.15
	38	254.92	292.65
	39	254.92	271.65

Problem Instance	City	Coord X	Coord Y
	40	254.92	243.15
	41	254.92	221.65
	42	254.92	193.15
	43	254.92	171.65
	44	276.42	171.65
	45	296.42	150.65
	46	276.42	150.65
	47	375.92	150.65
	48	308.92	150.65
	49	354.92	164.65
	50	338.42	174.65
	51	354.92	174.65
	52	338.42	200.15
	53	338.42	221.65
	54	354.92	221.65
	55	354.92	200.15
	56	361.92	200.15
	57	361.92	186.15
	58	383.42	186.15
	59	383.42	179.15
	60	404.42	179.15
	61	404.42	186.15
	62	418.92	186.15
	63	418.92	200.15
	64	432.92	200.15
	65	432.92	221.65
	66	418.92	221.65
	67	418.92	235.65
	68	397.42	235.65
	69	397.42	243.15
	70	375.92	243.15
	71	375.92	257.15
	72	368.92	257.15
	73	368.92	264.15
	74	347.42	264.15
	75	347.42	278.65
	76	336.42	278.65
	77	336.42	328.15
	78	347.42	328.15
	79	347.42	342.65
	80	368.92	342.65

Problem Instance	City	Coord X	Coord Y
	81	368.92	353.65
	82	418.92	353.65
	83	418.92	342.65
	84	432.92	342.65
	85	432.92	356.65
	86	447.42	356.65
	87	447.42	321.15
	88	447.42	292.65
	89	432.92	292.65
	90	432.92	314.15
	91	418.92	314.15
	92	418.92	321.15
	93	397.42	321.15
	94	397.42	333.65
	95	375.92	333.65
	96	375.92	321.15
	97	361.92	321.15
	98	361.92	299.65
	99	375.92	299.65
	100	375.92	285.65
	101	397.42	285.65
	102	397.42	271.65
	103	418.92	271.65
	104	418.92	264.15
	105	439.92	264.15
	106	439.92	250.15
	107	454.42	250.15
	108	454.42	243.15
	109	461.42	243.15
	110	461.42	214.65
	111	461.42	193.15
	112	447.42	193.15
	113	447.42	179.15
	114	439.92	179.15
	115	439.92	167.65
	116	419.92	167.65
	117	419.92	150.65
	118	439.92	150.65
	119	454.42	150.65
	120	475.92	150.65
	121	475.92	171.65

Problem Instance	City	Coord X	Coord Y
	122	496.92	171.65
	123	496.92	193.15
	124	496.92	214.65
	125	496.92	243.15
	126	496.92	271.65
	127	496.92	292.65
	128	496.92	317.15
	129	496.92	335.65
	130	470.42	335.65
	131	470.42	356.65
	132	496.92	356.65
	133	347.42	150.65
	134	539.92	356.65
	135	560.92	356.65
	136	589.42	356.65
	137	589.42	342.65
	138	603.92	342.65
	139	610.92	342.65
	140	610.92	335.65
	141	610.92	321.15
	142	624.92	321.15
	143	624.92	278.65
	144	610.92	278.65
	145	610.92	257.15
	146	589.42	257.15
	147	589.42	250.15
	148	575.42	250.15
	149	560.92	250.15
	150	542.92	250.15
	151	542.92	264.15
	152	560.92	264.15
	153	575.42	264.15
	154	575.42	271.65
	155	582.42	271.65
	156	582.42	285.65
	157	596.42	285.65
	158	560.92	335.65
	159	596.42	314.15
	160	582.42	314.15
	161	582.42	321.15
	162	575.42	321.15

Problem Instance	City	Coord X	Coord Y
	163	575.42	335.65
	164	525.42	335.65
	165	525.42	314.15
	166	525.42	299.65
	167	525.42	281.65
	168	525.42	233.15
	169	525.42	214.65
	170	525.42	193.15
	171	525.42	171.65
	172	546.92	171.65
	173	546.92	150.65
	174	568.42	150.65
	175	475.92	160.65
	176	603.92	150.65
	177	624.92	150.65
	178	624.92	136.15
	179	596.42	136.15
	180	575.42	136.15
	181	553.92	136.15
	182	532.42	136.15
	183	575.42	356.65
	184	489.92	136.15
	185	468.42	136.15
	186	447.42	136.15
	187	425.92	136.15
	188	404.42	136.15
	189	370.42	136.15
	190	361.92	150.65
	191	340.42	136.15
	192	326.42	136.15
	193	301.92	136.15
	194	276.42	136.15
	195	254.92	136.15
	196	315.92	136.15
	197	212.42	136.15
	198	190.92	136.15
	199	338.92	150.65
	200	155.42	136.15
	201	624.92	299.65
	202	318.92	321.65
	203	155.42	314.15

Problem Instance	City	Coord X	Coord Y
	204	311.92	356.65
	205	355.42	136.15
	206	318.92	314.15
	207	362.92	164.65
	208	254.92	356.65
	209	383.42	333.65
	210	447.42	335.65
	211	470.42	345.65
	212	525.42	250.15
	213	546.92	335.65
	214	525.42	261.15
	215	525.42	356.65
	216	336.42	298.65
	217	336.42	313.15
	218	293.42	136.15
	219	336.42	306.15
	220	425.92	264.15
	221	391.42	353.65
	222	482.92	335.65
	223	429.92	167.65
	224	330.92	150.65
	225	368.42	150.65
Att532	1	7810	6053
	2	7798	5709
	3	7264	5575
	4	7324	5560
	5	7547	5503
	6	7744	5476
	7	7821	5457
	8	7883	5408
	9	7874	5405
	10	7927	5365
	11	7848	5358
	12	7802	5317
	13	7962	5287
	14	7913	5280
	15	7724	5210
	16	7503	5191
	17	7759	5143
	18	7890	5130
	19	7254	5129

Problem Instance	City	Coord X	Coord Y
	20	7790	5038
	21	7142	5032
	22	7606	5009
	23	7772	4989
	24	7744	4933
	25	7846	4923
	26	7622	4917
	27	6937	4917
	28	7576	4915
	29	7783	4912
	30	7716	4909
	31	7295	4887
	32	7777	4869
	33	7700	4854
	34	7726	4833
	35	7702	4815
	36	7583	4813
	37	7654	4795
	38	7417	4788
	39	7267	4779
	40	6806	4755
	41	5259	4751
	42	7698	4745
	43	7570	4741
	44	7617	4724
	45	7752	4721
	46	7673	4718
	47	7692	4666
	48	7547	4664
	49	7259	4630
	50	5387	4623
	51	7679	4581
	52	7674	4579
	53	7631	4573
	54	7520	4572
	55	7848	4546
	56	5685	4546
	57	7832	4542
	58	6735	4509
	59	7647	4504
	60	7338	4481

Problem Instance	City	Coord X	Coord Y
	61	4602	4478
	62	4606	4468
	63	7399	4467
	64	7037	4446
	65	7458	4428
	66	7364	4427
	67	6058	4426
	68	6868	4418
	69	3832	4410
	70	6670	4401
	71	7443	4375
	72	7160	4370
	73	6139	4369
	74	7333	4335
	75	6237	4332
	76	5385	4318
	77	6911	4296
	78	6304	4294
	79	7111	4288
	80	6740	4282
	81	7698	4279
	82	7613	4275
	83	7360	4275
	84	6779	4273
	85	7207	4270
	86	6241	4268
	87	7432	4265
	88	4354	4262
	89	6589	4256
	90	7817	4252
	91	6051	4246
	92	5356	4241
	93	7554	4236
	94	7534	4227
	95	4217	4224
	96	7349	4219
	97	7128	4215
	98	3950	4215
	99	6947	4209
	100	7549	4208
	101	5168	4208

Problem Instance	City	Coord X	Coord Y
	102	6524	4207
	103	5871	4202
	104	7542	4198
	105	6660	4193
	106	7216	4180
	107	6607	4173
	108	7601	4171
	109	6123	4167
	110	6450	4160
	111	6713	4154
	112	7355	4151
	113	7604	4146
	114	7541	4141
	115	7506	4138
	116	4871	4132
	117	2906	4131
	118	6488	4128
	119	6312	4126
	120	6008	4117
	121	4427	4109
	122	4679	4084
	123	5955	4081
	124	6891	4075
	125	7705	4065
	126	7562	4058
	127	4634	4054
	128	4607	4049
	129	6557	4047
	130	7344	4046
	131	5543	4042
	132	7124	4039
	133	7466	4037
	134	6259	4030
	135	6366	4002
	136	5597	3993
	137	4655	3992
	138	7805	3991
	139	3396	3990
	140	6603	3982
	141	6537	3982
	142	4342	3966

Problem Instance	City	Coord X	Coord Y
	143	7037	3965
	144	7345	3951
	145	7271	3948
	146	5336	3943
	147	5964	3935
	148	7660	3924
	149	7872	3922
	150	6567	3922
	151	6602	3920
	152	4806	3914
	153	7909	3912
	154	5926	3912
	155	7449	3911
	156	6333	3909
	157	3108	3908
	158	7844	3902
	159	5427	3894
	160	6862	3892
	161	6621	3891
	162	6150	3888
	163	7388	3879
	164	7351	3877
	165	4694	3877
	166	6340	3870
	167	6425	3867
	168	6577	3858
	169	6864	3854
	170	5706	3844
	171	4496	3844
	172	4574	3843
	173	3824	3838
	174	5803	3824
	175	5720	3823
	176	6454	3821
	177	6120	3821
	178	7988	3820
	179	6376	3819
	180	7841	3818
	181	5778	3813
	182	5457	3808
	183	5671	3807

Problem Instance	City	Coord X	Coord Y
	184	4293	3788
	185	7423	3776
	186	7342	3775
	187	5541	3769
	188	5621	3768
	189	7750	3760
	190	6327	3745
	191	7879	3743
	192	199	3743
	193	6652	3742
	194	5678	3742
	195	5207	3742
	196	7429	3737
	197	7262	3725
	198	6427	3717
	199	1851	3710
	200	6207	3700
	201	6069	3695
	202	4780	3694
	203	7603	3690
	204	5751	3681
	205	6365	3679
	206	6958	3678
	207	6317	3673
	208	5417	3673
	209	6426	3656
	210	7922	3655
	211	7331	3634
	212	5965	3624
	213	4965	3622
	214	6833	3618
	215	6798	3610
	216	7667	3608
	217	1047	3602
	218	7803	3598
	219	7370	3588
	220	952	3583
	221	7906	3580
	222	250	3578
	223	5111	3569
	224	6453	3567

Problem Instance	City	Coord X	Coord Y
	225	7492	3560
	226	6140	3558
	227	5315	3557
	228	5316	3554
	229	4232	3551
	230	7408	3534
	231	8013	3523
	232	5160	3517
	233	7141	3514
	234	5887	3508
	235	4694	3502
	236	7633	3499
	237	7919	3496
	238	1784	3494
	239	1482	3494
	240	236	3494
	241	6713	3488
	242	7696	3486
	243	536	3481
	244	317	3476
	245	5649	3472
	246	6235	3471
	247	7199	3469
	248	5540	3468
	249	5400	3461
	250	5796	3459
	251	2342	3439
	252	7494	3430
	253	7321	3429
	254	6265	3426
	255	8001	3418
	256	226	3415
	257	6148	3413
	258	5987	3402
	259	7582	3396
	260	7422	3390
	261	6623	3389
	262	7475	3388
	263	7654	3377
	264	7838	3375
	265	6570	3371

Problem Instance	City	Coord X	Coord Y
	266	4364	3362
	267	7316	3360
	268	4857	3359
	269	7533	3358
	270	5719	3352
	271	7452	3339
	272	7747	3329
	273	5841	3328
	274	3229	3312
	275	7076	3302
	276	7657	3301
	277	6360	3301
	278	525	3297
	279	5619	3291
	280	7989	3271
	281	5697	3269
	282	6050	3242
	283	7082	3235
	284	5539	3235
	285	741	3235
	286	6731	3234
	287	7453	3229
	288	7695	3220
	289	7299	3219
	290	863	3219
	291	7861	3216
	292	5960	3207
	293	4252	3206
	294	6402	3190
	295	5342	3188
	296	6656	3181
	297	7532	3175
	298	7434	3173
	299	5679	3171
	300	6518	3165
	301	4537	3143
	302	806	3123
	303	6113	3101
	304	7440	3100
	305	6204	3099
	306	7715	3086

Problem Instance	City	Coord X	Coord Y
	307	7503	3086
	308	5821	3086
	309	7131	3081
	310	7909	3080
	311	920	3065
	312	6468	3050
	313	5677	3049
	314	218	3031
	315	6881	3029
	316	5650	3023
	317	197	3021
	318	5531	3011
	319	6387	3008
	320	4458	3007
	321	6190	2985
	322	7055	2981
	323	7238	2957
	324	5930	2948
	325	7543	2929
	326	5291	2929
	327	4196	2929
	328	6617	2928
	329	4831	2917
	330	2835	2912
	331	174	2901
	332	5350	2867
	333	7346	2858
	334	6044	2848
	335	4898	2840
	336	3307	2833
	337	1918	2832
	338	7125	2823
	339	6422	2820
	340	5881	2817
	341	141	2814
	342	7851	2809
	343	4929	2803
	344	5963	2789
	345	5470	2774
	346	7458	2741
	347	1263	2734

Problem Instance	City	Coord X	Coord Y
	348	6766	2732
	349	4763	2720
	350	3461	2718
	351	7309	2717
	352	6848	2712
	353	178	2702
	354	1882	2684
	355	4584	2643
	356	3174	2627
	357	7049	2570
	358	7753	2564
	359	6597	2563
	360	4476	2555
	361	1575	2555
	362	7304	2550
	363	10	2537
	364	6800	2532
	365	5296	2520
	366	7104	2510
	367	6547	2506
	368	7267	2466
	369	3189	2411
	370	5117	2409
	371	4973	2406
	372	4488	2378
	373	7351	2376
	374	6007	2359
	375	4612	2341
	376	7015	2333
	377	3233	2329
	378	240	2327
	379	6686	2312
	380	6307	2295
	381	7448	2291
	382	7087	2274
	383	2067	2254
	384	5260	2230
	385	4174	2190
	386	36	2185
	387	7856	2181
	388	7315	2181

Problem Instance	City	Coord X	Coord Y
	389	3319	2151
	390	2126	2150
	391	7418	2139
	392	6885	2138
	393	4959	2123
	394	4996	2115
	395	5681	2109
	396	5277	2078
	397	7643	2048
	398	3390	2043
	399	8080	2039
	400	6139	2032
	401	2694	2026
	402	7152	2000
	403	7822	1992
	404	7416	1953
	405	7352	1952
	406	354	1950
	407	6493	1931
	408	7905	1921
	409	8229	1905
	410	6803	1886
	411	4012	1886
	412	4759	1883
	413	8101	1876
	414	7989	1876
	415	8063	1860
	416	8080	1835
	417	7004	1805
	418	6252	1795
	419	6826	1774
	420	7218	1773
	421	464	1773
	422	809	1766
	423	7240	1762
	424	7046	1757
	425	8098	1746
	426	7314	1739
	427	7035	1733
	428	5506	1719
	429	8184	1685

Problem Instance	City	Coord X	Coord Y
	430	6932	1683
	431	5914	1682
	432	2908	1681
	433	6496	1678
	434	8525	1664
	435	6765	1663
	436	7985	1657
	437	6854	1640
	438	7926	1627
	439	7973	1606
	440	5060	1577
	441	4056	1564
	442	5637	1558
	443	2011	1558
	444	8038	1535
	445	6651	1534
	446	552	1526
	447	6621	1513
	448	8594	1510
	449	4719	1504
	450	5472	1482
	451	8605	1479
	452	345	1476
	453	8228	1471
	454	5005	1458
	455	5114	1430
	456	5964	1421
	457	602	1395
	458	5098	1394
	459	5068	1390
	460	8292	1383
	461	6258	1354
	462	5010	1351
	463	6494	1347
	464	437	1344
	465	413	1338
	466	659	1331
	467	5840	1325
	468	6378	1314
	469	6379	1302
	470	6359	1298

Problem Instance	City	Coord X	Coord Y
	471	3245	1281
	472	450	1274
	473	478	1256
	474	5571	1255
	475	489	1254
	476	513	1247
	477	6136	1243
	478	4170	1232
	479	1721	1165
	480	893	1161
	481	5930	1151
	482	4619	1132
	483	4125	1125
	484	5139	1124
	485	572	1108
	486	4500	1093
	487	2372	1084
	488	993	1084
	489	527	1077
	490	5788	1053
	491	3719	1043
	492	4805	1033
	493	5140	1018
	494	5344	1003
	495	5532	998
	496	5069	998
	497	1595	942
	498	5666	914
	499	2260	913
	500	4244	896
	501	5596	892
	502	4569	886
	503	1072	883
	504	3499	863
	505	5136	825
	506	783	825
	507	834	757
	508	1406	750
	509	3390	698
	510	2384	695
	511	982	659

Problem Instance	City	Coord X	Coord Y
	512	1422	658
	513	1361	637
	514	1926	636
	515	1213	633
	516	1415	628
	517	1082	625
	518	1254	617
	519	5070	605
	520	1212	603
	521	1249	600
	522	3477	599
	523	1322	580
	524	1253	580
	525	1276	559
	526	2647	485
	527	1443	459
	528	1961	445
	529	1790	429
	530	1503	362
	531	5393	355
	532	5469	10

7.2 KP Benchmark Problems

#Problem Instance	Knapsack Capacity	Item	Weight	Profit
P01	165	Item-1	23	92
		Item-2	31	57
		Item-3	29	49
		Item-4	44	68
		Item-5	53	60
		Item-6	38	43
		Item-7	63	67
		Item-8	85	84
		Item-9	89	87
		Item-10	82	72
P02	26	Item-1	12	12
		Item-2	7	7
		Item-3	11	11
		Item-4	8	8
		Item-5	9	9
P05	104	Item-1	25	350
		Item-2	35	400
		Item-3	45	450
		Item-4	5	20
		Item-5	25	70
		Item-6	3	8
		Item-7	2	5
		Item-8	2	5
P07	750	Item-1	70	135
		Item-2	73	139
		Item-3	77	149
		Item-4	80	150
		Item-5	82	156
		Item-6	87	163
		Item-7	90	173
		Item-8	94	184
		Item-9	98	192
		Item-10	106	201

#Problem Instance	Knapsack Capacity	Item	Weight	Profit
		Item-11	110	210
		Item-12	113	214
		Item-13	115	221
		Item-14	118	229
		Item-15	120	240
P08	6404180	Item-1	382745	825594
		Item-2	799601	1677009
		Item-3	909247	1676628
		Item-4	729069	1523970
		Item-5	467902	943972
		Item-6	44328	97426
		Item-7	34610	69666
		Item-8	698150	1296457
		Item-9	823460	1679693
		Item-10	903959	1902996
		Item-11	853665	1844992
		Item-12	551830	1049289
		Item-13	610856	1252836
		Item-14	670702	1319836
		Item-15	488960	953277
		Item-16	951111	2067538
		Item-17	323046	675367
		Item-18	446298	853655
		Item-19	931161	1826027
		Item-20	31385	65731
		Item-21	496951	901489
		Item-22	264724	577243
		Item-23	224916	466257
		Item-24	169684	369261