Universidad de La Laguna

Masters Degree Thesis

On the healthy and balanced menu planning automatisation

Planificación automática de menús saludables y equilibrados

Alejandro Marrero Díaz

La Laguna, XX March 2019

Mr. **Eduardo Manuel Segredo González**, con N.I.F. 78-564-242-Z Part-time lecturer at Departamento de Ingeniería Informática y de Sistemas de la Universidad de La Laguna, as mentor

Mrs. **Coromoto Antonia León Hernández**, with N.I.F. 78-605-216-W Full-time lecturer at Departamento de Ingeniería Informática y de Sistemas de la Universidad de La Laguna, as mentor

C E R T I F Y

That this report named:

"*On the healthy and balance menu planning automatisation.*"

has been done under they supervision by Mr. **Alejandro Marrero Díaz**, with N.I.F. 78-649-404-F.

And to be recorded, in compliance of the current legislation and timely effects XX March 2019.

Acknowledgements

License

Abstract

With the raise of diseases related with unhealthy lifestyles such as heart-attacks, overweight, diabetes, etc. Encouraging healthy and balanced patterns in the population is one of the most important action points for governments around the world. Furthermore, it is actually even a more critical situation when a high percentage of patients are child and teenagers whose habits consists merely in eating fast or ultra-processed food and a sedentary life.

The development of healthy and balanced menu plans became a routine task for physicians and nutritionists, and it is at this point that computer science has taken an important role. Discovering new approaches for generating healthy and balanced as well as inexpensive menu plans will play a part in banish of diseases from actual and new generations.

In this Master Thesis, a evolutionary algorithm have been compared with other state-of-art evolutionary algorithm for solving the Menu Planning Problem. In order to evaluate the performance of the developed algorithm, an exhaustive experimental research was made. Firstly, we focused on evaluate the parameter tuning of the algorithm so afterwards the best configuration found could be compared with other algorithms.

Keywords: menu-plan, computer-science

Contents

List of Figures

List of Tables

Chapter 1

Motivation

1.1 Description of the Master Thesis

In this Master Thesis, the main intention is to develop and evolutionary algorithm for solving the well-known *Menu Planning Problem (MPP)*. The MPP is an optimisation problem which is based on design menu plan under some restrictions. Although there are a lot of different kind of algorithms for solving such a problem, a high percentage of published papers use evolutionary algorithm approaches or any other type of meta-heuristics due its large benefits like robustness, reliability, global-search ability and its simplicity [12–14, 19, 22].

Concretely, this Master Thesis will be focused on solving the MPP formulation proposed in [18] with an evolutionary algorithm called *Multi-objective Evolutionary Algorithm based on Decomposition* and compare its performance with other state-of-art algorithms such as *Nondominated Sorting Genetic Algorithm II (NSGA-II)* and *Strength Pareto Evolutionary Algorithm 2 (SPEA 2)*.

1.2 State of the art

The Menu Planning Problem *(MPP)* is a well-known NP-Problem which has been trying to computerise since 1960 [20]. In essence, the MPP is to find a set of dishes combination which satisfies some restrictions of budge, variety and nutritional requirements for a n days sequence. In addition, it can include other constraints such as user preferences, cooking time or the number for meals each day. Even though there is not consensus about the number of objectives that a MPP's formulation may have, in almost every formulation the cost of the menu plan is considered as one of the main objectives to optimise [19, 20] but it also supports other objective functions like maximising the variability or minimising the cooking time.

Furthermore, the MPP can be studied as a multi-objective problem [21] if the amounts of nutrients requirements and cost of the meals are considered independent objectives. This approach leads to reduce the MPP to a Multi-dimensional Knapsack Problem *(MDKP)* where the maximum amount of each nutrient define the limit of the multiple dimensions. However, the MPP is also studying as a single-objective problem where mainly research define the objective function as the total cost of the meals. For instance, a single-objective approach for the MPP is in [19] where the authors proposed an evolutionary approach to solving the 5-day Single-Objective Menu Planning Problem composed by three meals daily, using as a function to minimise the total cost of the designed menus. In addition, the set of constraints that the researchers defined to this problem are moderately different from the usual constraints set for the typical MPP. In this occasion, the authors set the student age group, the school category, school duration time, school location, variety of preparations, the maximum amount to be paid for each meal and finally, the lower and upper limits of macro-nutrients as the constraints set to be satisfied for each solution to be considered feasible. Within this research, the authors used the generic Genetic Algorithm (GA) for the computational experiments. The results obtained from the generic GA where compared with a Greedy-based approach. The results prove that the GA outperforms the Greedy-based approach when the limit values of the meals are fixed at R$ 2.00 for breakfast, R$ 4.00 for lunch and R$ 2.00 for the snack. (BRL - R$ 1.0 USD - $ 0.31).

In the other hand, in the paper [10], the authors refer to the Two-phase Cooking N-day Menu Planning Problem where the objective is to maximise the preferences among the selected foods in the menu plan. The conditions which shape the set of constraints that must be satisfied are only three. The total cooking time of any day must not exceed the limit specified, only foods that which allow two-phase cooking can be selected for two-phase cooking and finally, the food cannot be repeated more times than a certain repeat constraint. In order to face this problem, the researchers used a simple greedy method prioritising the user-specified preference with the cooking time of each food.

Eventually, another study where the MPP is faced as a single-objective problem is [25]. Here, the authors set up a mathematical model to solve the MPP considering only one objective function. The model's goal is to minimise the budget provided by the government subject to the restriction of trying to maximise the variety of dishes. Furthermore, the model tries to create menus in such a way they maximise the nutritional requirements. For the computational experiments, the researchers programmed an Integer Programming algorithm in Matlab using LPSolve. Furthermore, the given results, taking into account that the optimal solution was found within one second, are better compared to other heuristics like GA.

As can be seen, there is a certain variety within the optimisation methods for solving the single-objective MPP approach. Despite that, Evolutionary Computing *(EC)* techniques, such as GA, are mostly cited in the related bibliography as a good choice [19–21].

Therefore, this Master Thesis is focused on develop, analyse and compare an evolutionary algorithm for solving the Menu Planning Problem with other evolutionary algorithms focusing on a real world case.

Chapter 2

Introduction

2.1 Optimisation Problems

An Optimisation Problem (OP) is a problem which has a score function and bounds where the main task is to find a input that optimises the score function. Optimisation problems can be categorised as *discrete optimisation problem (DOP)* or *continuous optimisation problem (COP)* whether the variables of the problem are discrete or continuous. Formally speaking, an OP can be described as follows:

$$min\ f(x), x \in \chi, \quad s.t.\, \Omega$$

where $\chi \subset \mathbb{R}^n$ is the search space defined over a set of n decision variables $x = (x_1, x_2, ..., x_n)$, $f : \chi \rightarrow \mathbb{R}$ is the score function and Ω is the restrictions set in x. This is the definition for a minimisation DOP, although it would be equivalent for a maximisation DOP changing $min\ f(x)$ by $max\ f(x)$. The same happens for a COP, if in addition the decision variables are set over $\mathbb{Z}^n$ instead of $\mathbb{R}^n$.

Furthermore, optimisation problems may have more than one score function and they are called *Multi-Objective Optimisation Problems (MOOP)*. This is the primary field of study in this work so, hereinafter all the references to optimisation problems in this work will be to MOOP.

2.1.1 Multi-Objective Optimisation Problems

Multi-Objective Optimisation Problems (MOOPs) are optimisation problems which have two or more objective functions to optimise and those objective functions can take opposite directions (thinking about directions as *minimise or maximise*). Besides, a MOOP can discrete or continuous considering whether the variables of the problem are discrete or continuous.
Formally speaking, a MOOP can be described as finding a vector x inside the problem's search space χ in such way that optimises the vector of objective

functions $f(x)$ [6]:

$$\begin{aligned} min\ f(x) = (f_1(x), f_2(x), ..., f_k(x)),\ x\ \in \chi \\ g_i(x) \leq 0,\ i = 1, 2, ..., q. \\ h_i(x) \leq 0,\ i = 1, 2, ..., p. \end{aligned}$$

where $x = (x_1, x_2, ..., x_n) \in \mathbb{Z}^n$, are the objective functions to optimise $f_i : \mathbb{Z}^n \rightarrow \mathbb{R},\ i = 1, ..., k$ being n the number of decision variables and $g_i : \mathbb{Z}^n \rightarrow \mathbb{R},\ i =\ 1, ..., q$ and $h_i : \mathbb{Z}^n \rightarrow \mathbb{R},\ i = 1, ..., p$ are the problem's restriction functions.

Moreover, the standard method for solving a MOOP is the well-known *Pareto method* [6]. The Pareto method is based on the non-dominance principle [6,26]. On the one hand, dominance means that given two solutions for a MOOP, one solutions dominates the other one when it has as least the same quality for every objective and, it has strictly more quality for one of them than the other solution. Formally, this can be expressed as follows [6]:

$$\begin{aligned} A \succeq B \Leftrightarrow \forall i \in \{1, 2, ..., n\}\ a_i \leq b_i, \\ y\ \exists i \in \{1, 2, ..., n\},\ a_i < b_i \end{aligned}$$

On the other hand, it is the direction conflict between objectives which leads to solutions with trade-offs between those objectives. So, at this point it is where the *non-dominance* appears. The non-dominance refers the situation where a solution it is not dominated by any other solution of the problem. That means that it can not be found any other solution to the problem which increases the quality of any objective without irredeemably decreases the quality of another one. Non-dominated solutions may be found at the limits of the search space (χ) and are those which shape the Pareto set.

2.2 Evolutionary Algorithms

Nowadays, there are many methods for solving MOOPs but they can be classified merely in two types of methods: *approximated methods and exacts methods.* The different categories of exact and approximated methods can be seen down below.

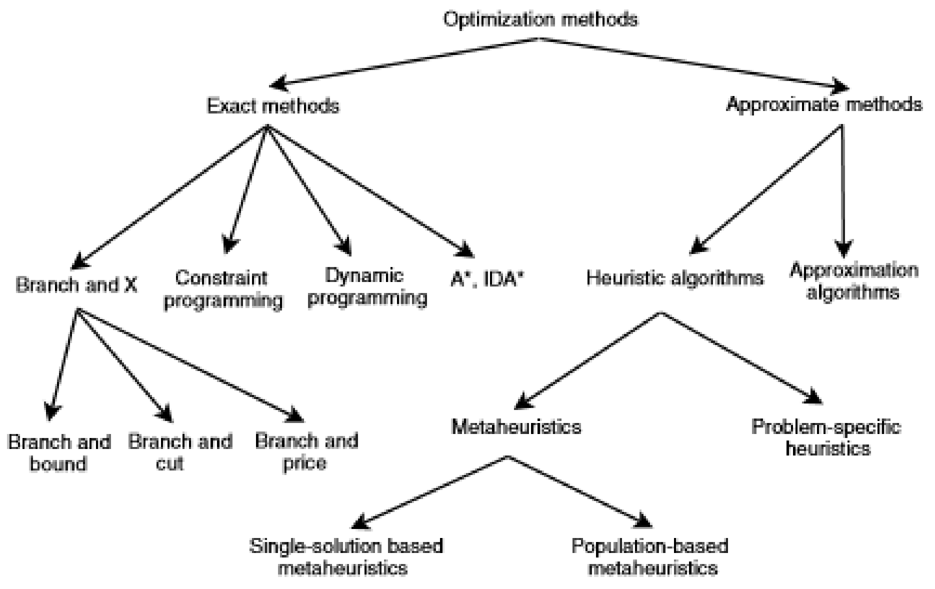

Figure 2.1: Optimisation methods

On the one hand, *exacts methods* are those which ensure that, if there is an optimal solution to the facing problem they will be able to find it. However, even though these methods guarantee reaching the optimal solution they have a important drawback on its performance. Assuring the optimal solution implies increasing the computational work and hence more time to obtain the solution.

On the other hand, *approximated methods* are very popular nowadays even though they do not guarantee reaching the optimal solution for a problem. Nevertheless, approximated methods can obtain high quality solutions in an assumable time due they set a balance between computational performance and solution quality.

Approximated methods can be divided in two categories: *heuristic algorithms* and *meta-heuristics algorithms.* However, this work it is focus primarily in *meta-heuristics algorithm* and even more specifically in the field of *evolutionary algorithms.*

Evolutionary algorithms (EA) develop the metaphor of natural evolution, the survival of the fittest individual [8]. This is, given a population of individuals in some environment with limited resources, the competition for surviving causes natural selection and the fittest individuals are more likely to survive and reproduce. Nevertheless, there are several variants of EA and they can be classified as:

1. Genetic Algorithms [2, 23, 28].
2. Evolutionary Strategies [4, 11].
3. Differential Evolution [1, 9, 27, 30].

Moreover, there are a specific group of evolutionary algorithms for solving multi-objective problems known as *Multi-Objective Evolutionary Algorithms (MOEAs)*. MOEAs can be classified in many different subgroups considering the main approach underlying the algorithm [31]:

- MOEAs based on decomposition, i.e. MOEA/D [17, 29].
- MOEAs based on preference, i.e. NSGA-II [5].
- Indicator-based MOEAs, i.e. IBEA [32].
- PSO-based approaches, i.e. SPEA-2 [15].

Considering the field of optimisation problems, the natural evolution metaphor is develop as follows:

1. The problem to solve and its bounds is the environment with limited resources.
2. A set of random initial solutions for the given problem are the first individuals at generation zero.
3. The population of individuals reproduce between each other applying genetic operators to generate offspring. Commonly combination and mutation.
4. At each generation, the individuals within a population compete and the fittest individuals (the better quality solutions) survive.
5. Steps three and four are repeated until reaching the stop condition.

Generally, the before metaphor can be shown as a pseudocode [8]:

Algorithm 1 Pseudocode of an EA.

```
1: INITIALISE population with random candidate solutions
2: EVALUATE each candidate
3: while not StopCriteria satisfied do
4:     SELECT parents
5:     RECOMBINE pairs of parents
6:     MUTATE the result offspring
7:     EVALUATE new candidates
8:     SELECT individuals for the next generation
9: end
```

2.3 Menu Planning Problem Formulation considered

In this particular case, a new formulation of the Menu Planning Problem proposed in [18] for school cafeterias is considered. The authors defined two objectives: meal cost and variety of dishes. On the one hand, as usual in MPP, one goal is to minimise the total cost of the meal plan generated. Since the meal plan is designed for school cafeterias, the authors considered three meals in each menu: first course, second course and dessert. Formally, the meal plan cost can be defined as follows:

$$min\ C = \sum_{i=1}^{n} c_{fc_i} + c_{sc_i} + c_{d_i}$$

where C is the total cost of the menu plan and $c_{fc_i}, c_{sc_i}, c_{d_i}$ represent the cost of the first course, second course and dessert in n days.

On the other hand, an assorted menu plan is a must for children in order to avoid them to annoy about the food. For that reason, the second objective is to minimise the level of repetition of dishes and food groups in a certain menu plan.

$$min\ L_{Rep} = \sum_{i=1}^{n} v_{table_i} + \frac{p_{fc}}{d_{fc_i}} + \frac{p_{sc}}{d_{sc_i}} + \frac{p_{ds}}{d_{ds_i}} + v_{FG_i}$$

Where L_{Rep} is the level of repetition to minimise, v_{table_i} represents the compatibility between the courses $c_{fc_i}, c_{sc_i}, c_{d_i}$ for day n, p is a penalty constant for every kind of course and d stands for the number of days since the course was repeat for the last time. Finally, v_{FG_i} is the penalty value for repetition of food groups in the last five days.

Additionally, in order to consider a menu plan as feasible, it must keep to some restrictions about a set nutritional requirements (N). The nutritional requirements considered in this formulation are the follow:

- Folic acid.
- Calcium.
- Energy (Kcal).
- Phosphorus.
- Fat.
- Iron.
- Magnesium.
- Potassium.
- Protein.
- Selenium.
- Sodium.
- Vitamin A.
- Vitamin B1.
- Vitamin B2.

- Vitamin B6.
- Vitamin B12.
- Vitamin C.
- Vitamin D.
- Vitamin E.
- Iodo.
- Zinc.

There is also a vector R in which the minimum and maximum amount of each nutritional requirement is stored. So formally speaking, a menu plan is feasible only if

$$\forall n \in N \ : R_{min_n} \leq I_n \leq R_{max_n}$$

where I_n is the amount of the n-nutritional requirement in the menu plan.

Lastly, the set G of food groups considered for the available meals is:

- Meat.
- Cereal.
- Fruit.
- Dairy.
- Fish.
- Vegetable.
- Shellfish.
- Legume.
- Pasta.
- Others.

Chapter 3

Algorithms

At this point, it will be introduced the evolutionary algorithms compared in this thesis. With a view to have variety of MOEAs, every algorithm it is based on a different MOEA approach [31]:

- Based on decomposition: Multi-objective Evolutionary Algorithm Based on Decomposition.
- Based on preference: Non-dominated Sorting Genetic Algorithm II.
- Based on PSO: Strength Pareto Evolutionary Algorithm 2.

3.1 Multi-objective Evolutionary Algorithm Based on Decomposition

Multi-objective Evolutionary Algorithm Based on Decomposition (MOEA/D) is an evolutionary algorithm for multi-objective optimisation proposed by Qingfu Zhang and Hui Li in 2007 [29]. The underlying idea behind this algorithm is to decompose a multi-objective optimisation problem into a number of scalar optimisation sub-problems and optimises them simultaneously. It also harnesses in the well-known feature of Pareto optimal solutions to a MOP, which sustains that an optimal solution for a scalar optimisation problem with an objective function as the aggregation of all the f_i could be the same as the Pareto optimal solution for the MOP [29].

The decomposition approach of MOEA/D takes place where the algorithm decomposes a MOP into N sub-problems and simultaneously optimises every single sub-problem at each generation. Furthermore it establish some relations between sub-problems and organised them in neighbourhoods. These neighbourhoods are shaped by sub-problems which coefficient vectors are very similar to each other and every single sub-problem is optimised based on its neighbouring sub-problems information. Therefore, the optimal solution for two neighbouring sub-problems should be very similar [29].

On the other hand, the process of decompose a MOP into N sub-problems can be done from some different approaches. However, as the authors referred [29] it this paper the MOEA/D uses the Tchebycheff Approach [17] to decompose a MOP. Formally, the Tchebycheff Approach it is defined as follows:

$$min \quad g^{te}(x|\lambda, z^*) = max_{i=1}^{m}\{\lambda_i|f_i(x) - z_i^*|\}$$

where $z^* = (z_1^*, ..., z_m^*)$ is the reference point with the best solution founds so far for each sub-problem and $\lambda_i = (\lambda_{i,1}, ..., \lambda_{i,m})$ is a even spread weight vector for each sub-problem i.

In addition, MOEA/D version implemented in this paper works as follows. It takes a MOP, the population size, a stopping criterion and the number of neighbours for each neighbourhood. The number of sub-problems in this implementation are the MOP's objectives. Then, it starts by randomly generate N even spread weight vectors and compute the Euclidean distance between each others to shape the neighbourhoods, generates an initial random population and computes the reference point Z^*. After the initialisation phase, it goes into the main loop where, until the stopping criteria is not satisfied, the algorithm preforms theses steps for each individual of the population:

- Reproduction: generates a new child individual from two randomly selected neighbours *l, k*.
- Improve: maintains the new child under the limits of the problem's search space.
- UpdateZ: updates the reference point by comparing it with each new child individual.
- Update Neighbours: if the new child individual performs better than any neighbours, replaces the neighbour with the brand new individual.

Finally, MOEA/D returns the PF's points found.

Concretely, the algorithm can be outlined as follows:

Algorithm 2 MOEA/D.

```
1: SetRandomWeightVectors()
2: EuclideanDistance()
3: GenerateRandomPopulation()
4: InitializeZ()
5: while not StopCriteria satisfied do
6:     for all sub-problem do
7:         l,k = getRandomNeigbours()
8:         child = reproduce(l, k)
9:         child = improve(child)
10:        updateZ(child)
11:        updateNeighbouringSolutions(child)
12: end
```

3.2 Nondominated Sorting Genetic Algorithm II

Nondominated Sorting Genetic Algorithm II, as well known as *NSGA-II* was proposed in 2002 by K. Deb and A. Pratap and S. Agarwal and T. Meyarivan to mitigate the major difficulties of nondominated sorting MOEAs [5]. In fact, this algorithm is an improvement of the previously algorithm is an improvement of the previously suggested algorithm NSGA [16] algorithm NSGA [24] in 1994 by N. Srinivas and Kalyanmoy Deb.

The main improvements of NSGA-II over NSGA is a fast nondominated sorting approach with complexity $\mathcal{O}(MN^2)$ that replaces the previous one which has complexity $\mathcal{O}(MN^3)$ [5](considering M the number of objectives and N the population size) and the selection operator of NSGA-II which comes to solve the lack of elitism of the previous NSGA version.

The fast nondominated sorting procedure starts by computing the domination count n_p which is the number of solutions that dominates p and next, the set of solutions dominated by p called S_p. Then the procedure continues identifying all Pareto Fronts and ranking the solutions in different fronts by its n_p [5]. An example of the fast nondominated sort procedure can be seen at the following pseudocode.

Algorithm 3 Fast Nondominated Sort.

```
 1: for all individual p in population do
 2:     $S_p = \emptyset$
 3:     $n_p = 0$
 4:     for all individual q in population do
 5:         if $p \prec q$ then
 6:             $S_p = S_p \cup \{q\}$
 7:         else
 8:             if $q \prec p$ then
 9:                 $n_p = n_p + 1$
10:     if $n_p = 0$ then
11:         $p_{rank} = 1$
12:         $F_1 = F_1 \cup \{p\}$
13: $i = 1$
14: while $f_i \neq \emptyset$ do
15:     $Q = \emptyset$
16:     for all $p \in F_i$ do
17:         for all $q \in S_p$ do
18:             $n_q = n_q - 1$
19:             if $n_q = 0$ then
20:                 $q_{rank} = i + 1$
21:                 $Q = Q \cup \{q\}$
22:     $i = i + 1$
23:     $F_i = Q$
24: end
```

NSGA-II algorithm is quite simple and it can be seen in the pseudocode down below.

Algorithm 4 NSGA-II.

```
1: P = CreateInitialPopulation(N)
2: FastNondominatedSorting(P)
3: while not StopCriteria satisfied do
4:     BinaryTournamentSelection(P)
5:     Q = CreateOffspring(P)
6:     R = Combine(P, Q)
7:     FastNondominatedSorting(R)
8:     P = SelectNIndividuals(R, N)
9: end
```

3.3 Strength Pareto Evolutionary Algorithm 2

Likewise *NSGA-II* is an improvement of its predecessor NSGA, the *Strength Pareto Evolutionary Algorithm 2* was published in 2001 by Eckart Zitzler, Marco Laumanns and Lothar Thiele as a new version of SPEA algorithm proposed in 1999 by Zitzler and Thiele [7]. Essentially, the SPEA 2 differs with SPEA in a fine-grained fitness assignment strategy, a density estimation technique, and an enhanced archive truncation method [7]. The basis of both SPEA and SPEA 2 algorithm are that they uses a standard population (P) and also an *archive* $(\overline{P})$ or external population and follows theses steps:

- Create an initial random population of size N and an empty archive.
- Then all nondominated individuals are sent to the archive.
- If the size of the archive increase over the limit $\overline{N}$, new archive individuals are deleted preserving the nondominated front.
- Both population and archive individuals are evaluated and a fitness value is assigned to each of them.
- After the evaluation, the mating selection phase comes.
- When the parents are selected, genetic operators are applied to generate offspring and replace the old population.

Although those are the foundations of SPEA and SPEA 2, SPEA 2 has two improvements in *fitness assignment* and *environmental selection.* On the one hand, in an effort to avoid that individuals dominated by the same archive individuals have the same fitness, each individual **i** in $\overline{P}$ and P have a strength value $S(i)$ indicating the number of individuals it dominates [7].

$$S(i) = |\{j | j \in P_t + \overline{P_t} \wedge i \succ j\}|$$

After computing the strength of each individual, a *Raw fitness value (R)* is calculated for each individual.

$$R(i) = \sum_{j \in P_t + \overline{P_t}, j \succ i} S(j)$$

In the case where most individuals do not dominate each other, R it is not enough so an adaptation of the k-the nearest neighbour algorithm is included for additional density information. In this particular case, authors use k as the result of the square root of the sample size, so $k = \sqrt{N + \overline{N}}$. Then, the density of each individual is calculated as follows.

$$D(i) = \frac{1}{\sigma_i^k + 2}$$

Finally, the fitness value (F) of each individual is defined as the sum of its raw fitness plus its density.

$$F(i) = R(i) + D(i)$$

On the other hand, the archive updating procedure of SPEA 2 is slightly different from SPEA. Primarily, it ensures two aspects [7]:

- The number of individuals in the archive maintain is regular.
- The truncation method prevents boundary solutions being removed.

The environmental selection begins by copying all nondominated individuals from archive and population which have a fitness value F lower than one to the archive for the next generation.

$$\overline{P_{t+1}} = \{i | i \in \overline{P_t} + \overline{P_t} \wedge F(i) < 1\}$$

After finishing this step, if the archive is fully filling, the environmental selection is completed. Under other conditions, new individuals are added to the archive if it is too small or deleted in other case to fit the size $\overline{N}$.

All the previous description of SPEA 2 algorithm can be outlined in the following pseudocode.

Algorithm 5 SPEA 2.

```
1: P = CreateInitialPopulation(N)
2: $\overline{P}$ = CreateEmptyArchive();
3: while not StopCriteria satisfied do
4:     ComputeFitness(P,$\overline{P}$)
5:     EnvironmentalSelection();
6:     BinaryTournamentSelection();
7:     Recombination();
8:     Mutation();
9: end
```

Chapter 4

Experimental Evaluation

In this chapter, the experimental evaluation of MOEA/D will be introduce. In order to achieve an accurate comparison of MOEA/D with the results obtained in [18] by other algorithms mentioned before (NSGA-II and SPEA-2), the algorithm and the experimental evaluation were development through the Metaheuristic-based Extensible Tool for Cooperative Optimisation (METCO) [1] proposed in [16]. In addition, the experiment were executed on a Debian GNU/Linux computer with four AMD Opteron processors at 2.8GHz and 64 GB RAM and each run was repeated 25 times. Furthermore, following the evaluation procedure done by the authors in [18], the *hypervolume (HV)* [3] was the metric selected to compare the different configurations of MOEA/D and the *Shapiro-Wilk, Levene, ANOVA or Welch* test were considered for results which follow a normal distribution or *Kruskal-Wallis* test otherwise. However, in this thesis the number of individual evaluations was decresead from 4e8 evaluations to 1e8 evaluations in order to obtain faster results.

4.1 Instances

For this evaluation, a total number of 67 different courses were available group together in three different files:

- l_{st}: 19 starters.
- l_{mc}: 34 main courses.
- l_{ds}: 14 desserts.

Besides, the structure of every files is an CSV file with the following fields:

- Name of the course.
- Price of the course.

[1]Available at: https://github.com/PAL-ULL/software-metco

- Binary list of different allergens in case whether the course contains or not the allergen.
- Incompatibilities.
- Amount of the different nutrients.
- Food groups which belongs the course.

4.2 Parameter Setting

In this preliminary experiment, the goal was to find which values of the MOEA/D [29] parameters form the better configuration facing the MPP formulation considered in this thesis. The list of MOEA/D's parameters is:

- Population size.
- Neighbourhood size.
- Mutation probability that was set at 0.05.
- Crossover probability which was set at 1.

At this point, it must be say that MPP takes one parameter which defines the number of different days that in this preliminary experiment was set at **20 days**. The comments of the MOEA/D's authors in [29] about the performance of the algorithm with very small or large neighbourhood size and the effect of the population size on its performance were took into account when designing this experiment. However, a wide range of values were considered. Then, five different values for the population size and neighbourhood size were set in order to obtain 25 different configurations of MOEA/D. The values are:

- Population size: 25, 80, 140, 190, 250.
- Neighbourhood size: 0.4, 0.3, 0.25, 0.2, 0.16 of the total population size.

Table 4.1 shows the ranking of all MOEA/D configuration for a 20-days MPP related to the hypervolume values obtained at the end of the executions. The ranking *(R)* was calculated considering the number of times that one configuration outperforms other configurations *(W)* and the number of times that it was outperformed by other configurations *(L)*:

$$R = W - L$$

Configuration A statistically outperforms configuration B if the p-value, obtained after performing a pairwise comparison of both approaches by following the statistical testing procedure described at the beginning of this chapter, is lower than the significance level $\alpha = 0.05$, and if at the same time, A provides a higher mean and median of the hypervolume at the end of the runs [18].

Even though there is not a significant control of the best ranked configuration in table4.1 with only 9 wins over 24 other configurations, the MOEA/D configuration with a population size of 140 individuals and 42 individuals per neighbourhood seems to be the better one. Thus, this is the configuration chosen for the next experiment.

Configuration	**Min.**	**1st Qu.**	**Median**	**Mean**	**3rd Qu.**	**Max.**	**W**	**L**	**Ranking**
MOEA_D_PopSize_140_Neihb_42	0.7161	0.7733	0.7839	0.7834	0.8087	0.8435	9	0	9
MOEA_D_PopSize_250_Neihb_50	0.7270	0.7635	0.7737	0.7775	0.7930	0.8213	2	0	2
MOEA_D_PopSize_80_Neihb_16	0.7306	0.7655	0.7787	0.7774	0.7945	0.8342	2	0	2
MOEA_D_PopSize_140_Neihb_35	0.7388	0.7583	0.7671	0.7728	0.7914	0.8188	1	0	1
MOEA_D_PopSize_190_Neihb_48	0.7216	0.7521	0.7670	0.7688	0.7842	0.8034	0	0	0
MOEA_D_PopSize_25_Neihb_10	0.7228	0.7563	0.7669	0.7686	0.7805	0.8250	0	0	0
MOEA_D_PopSize_25_Neihb_4	0.7363	0.7490	0.7628	0.7682	0.7817	0.8174	0	0	0
MOEA_D_PopSize_80_Neihb_13	0.7231	0.7563	0.7690	0.7691	0.7874	0.8143	0	0	0
MOEA_D_PopSize_25_Neihb_8	0.7384	0.7490	0.7725	0.7716	0.7884	0.7989	0	0	0
MOEA_D_PopSize_140_Neihb_28	0.7299	0.7583	0.7715	0.7715	0.7892	0.8005	0	0	0
MOEA_D_PopSize_80_Neihb_32	0.7256	0.7428	0.7708	0.7675	0.7840	0.8190	0	0	0
MOEA_D_PopSize_250_Neihb_62	0.7244	0.7577	0.7733	0.7702	0.7847	0.8189	0	0	0
MOEA_D_PopSize_80_Neihb_24	0.7234	0.7540	0.7710	0.7712	0.7843	0.8158	0	0	0
MOEA_D_PopSize_140_Neihb_22	0.7292	0.7504	0.7728	0.7684	0.7813	0.8231	0	0	0
MOEA_D_PopSize_250_Neihb_100	0.7328	0.7501	0.7775	0.7721	0.7925	0.8253	0	0	0
MOEA_D_PopSize_25_Neihb_6	0.7211	0.7506	0.7673	0.7706	0.7897	0.8300	0	0	0
MOEA_D_PopSize_250_Neihb_40	0.7158	0.7507	0.7737	0.7674	0.7811	0.8155	0	1	-1
MOEA_D_PopSize_190_Neihb_57	0.6755	0.7441	0.7698	0.7655	0.7809	0.8135	0	1	-1
MOEA_D_PopSize_140_Neihb_56	0.7149	0.7472	0.7704	0.7664	0.7799	0.8311	0	1	-1
MOEA_D_PopSize_250_Neihb_75	0.7078	0.7443	0.7664	0.7637	0.7765	0.8143	0	1	-1
MOEA_D_PopSize_190_Neihb_76	0.7374	0.7546	0.7694	0.7685	0.7818	0.8018	0	1	-1
MOEA_D_PopSize_25_Neihb_5	0.7299	0.7494	0.7683	0.7672	0.7802	0.8180	0	1	-1
MOEA_D_PopSize_80_Neihb_20	0.7269	0.7484	0.7676	0.7663	0.7732	0.8217	0	1	-1
MOEA_D_PopSize_190_Neihb_38	0.7224	0.7515	0.7643	0.7634	0.7780	0.8113	0	3	-3
MOEA_D_PopSize_190_Neihb_30	0.7280	0.7494	0.7581	0.7607	0.7698	0.7978	0	4	-4

Table 4.1: Ranking of all MOEA/D configurations

4.3 10-days Experiment

In this experiment, the main goal is the analyse how the problem dimension affects in the performance of the best MOEA/D configuration found so far for MPP. For that reason, other 25 independent executions of MOEA/D with 140 individuals in the population and neighbourhoods of 42 individuals were launch with a 10-days MPP. The following figure 4.1 shows an approximation of the Pareto Front after 1e8 evaluations facing a 10-days MPP. And figure 4.2 shows the evolution of the minimum, average and maximum hypervolume values for MOEA/D after 1e8 facing a 10-days MPP, with a significant difference between the maximum and minimum hypervolume values at each checkpoint.

Algorithm	**Min**	**Median**	**Mean**	**Max**
MOEA_D_PopSize_140_Neihb_42_days_10	*0.743725*	*0.78073*	*0.78353884*	*0.83404*

Table 4.2: Results of MOEA/D facing a 10-days MPP.

Finally, in Figure 4.3 the performance of MOEA/D facing a 10-days and a 20-days MPP is compared by they respective maximum hypervolume values. As it can be see, the MOEA/D algorithm rapidly increases the maximum hypervolume value for a 10-days MPP but in the long term execution the maximum value stabilizes. However, MOEA/D seems to perform better on 20-days MPP compared against the 10-days MPP due to slightly higher maximum hypervolume values and an increasing inclination after 1e8 evaluations.

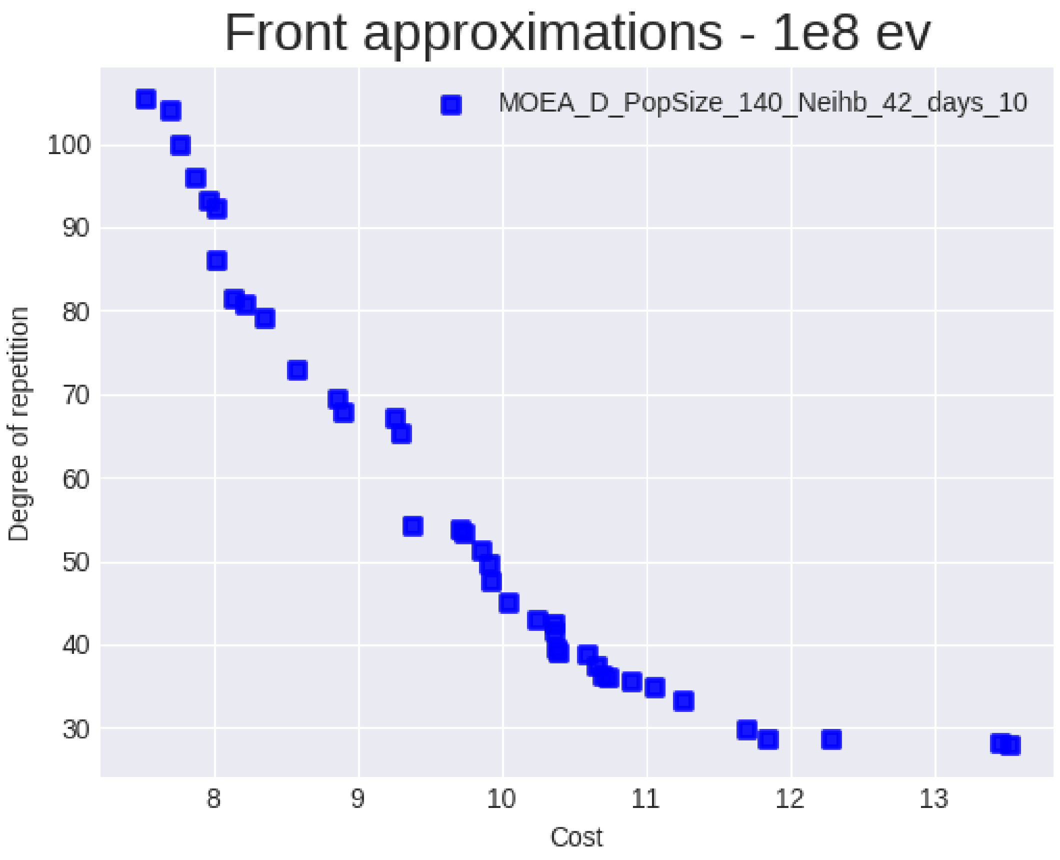

Figure 4.1: Front approximation of MOEA/D after 1e8 ev.

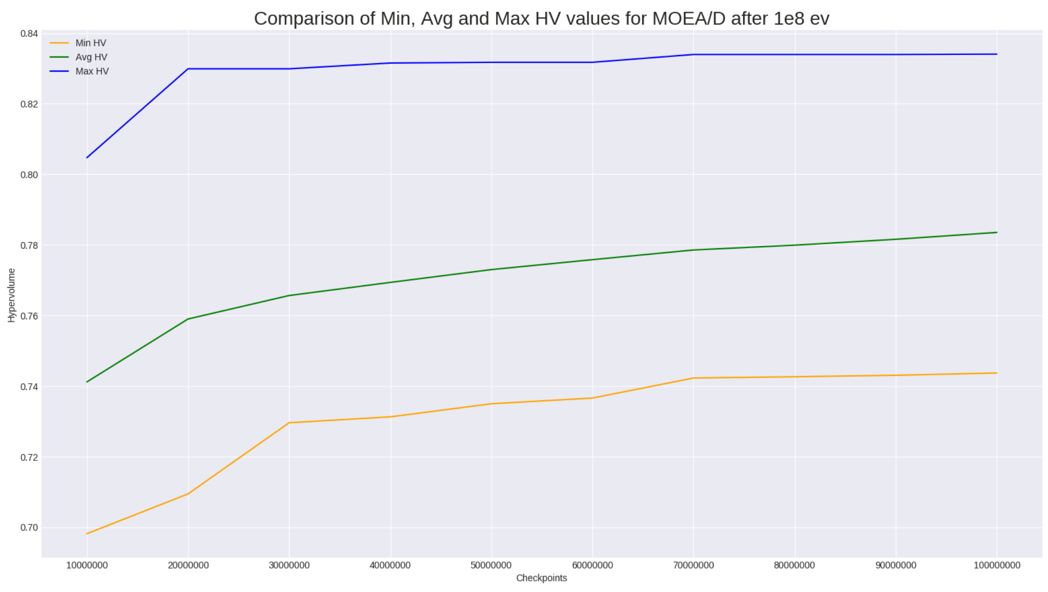

Figure 4.2: Evolution of the minimum, average and maximum HV value of MOEA/D after 1e8 ev.

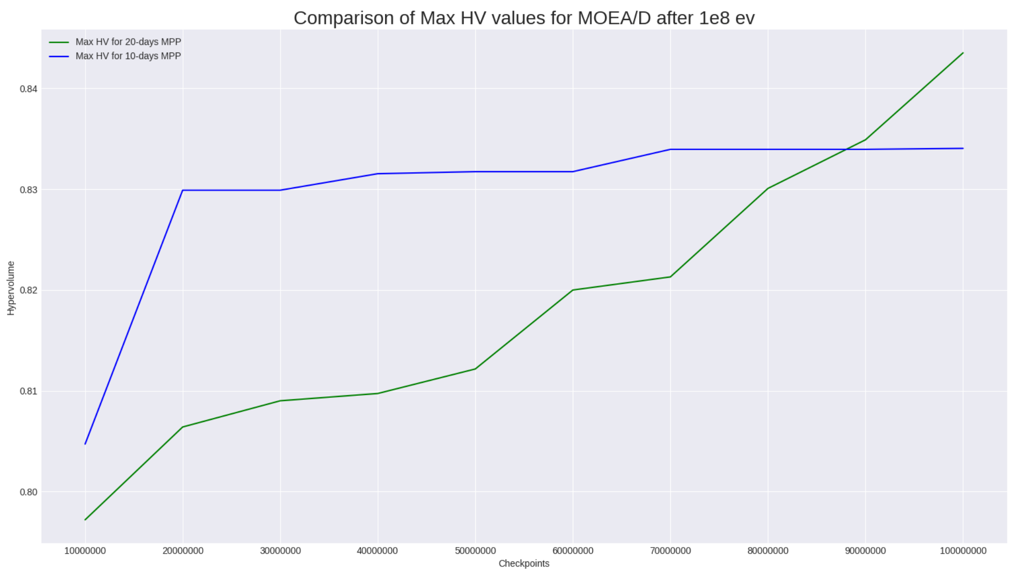

Figure 4.3: Comparison of the evolution of the maximum HV value of MOEA/D after 1e8 ev for 10-days and 20-days MPP.

4.4 Comparison with the Previous Work

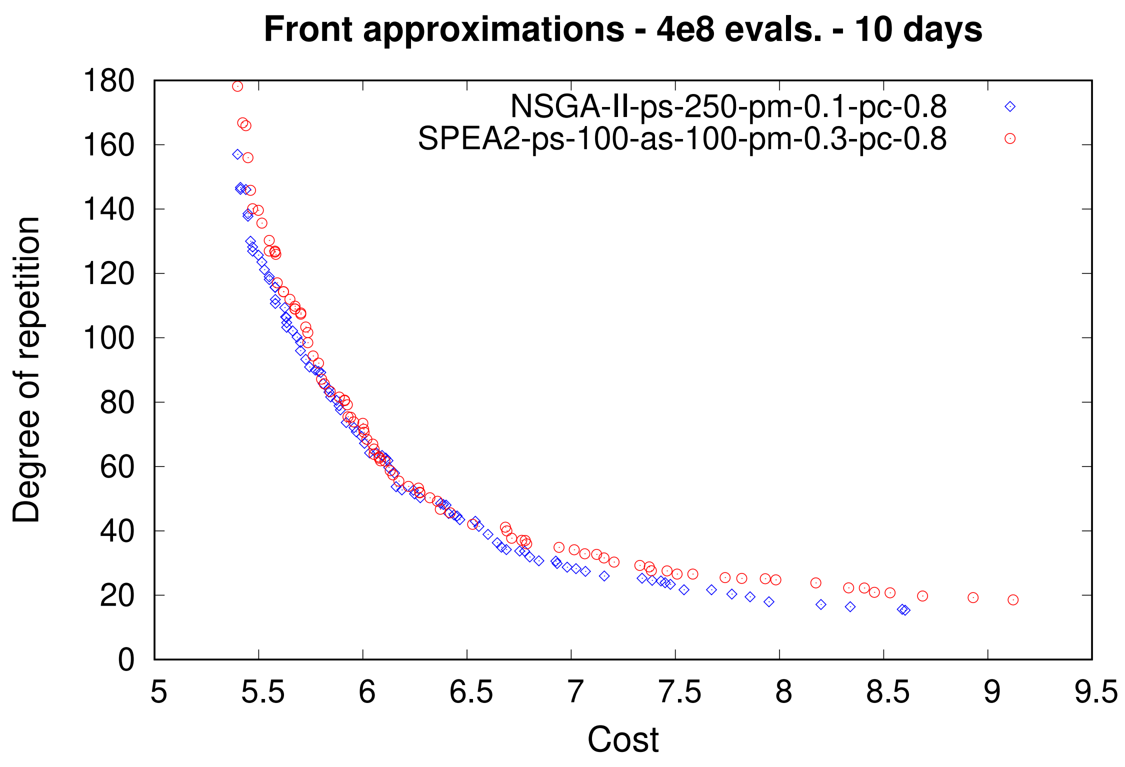

Figure 4.4: Front approximation of NSGA-II and SPEA-2 after 4e8 for a 10-days MPP.

Chapter 5

Summary and conclusions

5.1 Conclusions and future work

Bibliography

[1] The Algorithm. Differential Evolution. *Evolution*, 2006.

[2] Practical Genetic Algorithms, Second Edition, Rl Haupt, and Se Haupt. The continuous genetic algorithm. *Practical Genetic Algorithms, Second*, 2004.

[3] Anne Auger, Johannes Bader, Dimo Brockhoff, and Eckart Zitzler. Theory of the hypervolume indicator: Optimal μ-distributions and the choice of the reference point. In *Proceedings of the Tenth ACM SIGEVO Workshop on Foundations of Genetic Algorithms*, FOGA '09, pages 87–102, New York, NY, USA, 2009. ACM.

[4] Hans-georg Beyer and Hans-paul Schwefel. Evolution strategies. *Evolutionary Computation*, 2002.

[5] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. A fast and elitist multiobjective genetic algorithm: Nsga-ii. *IEEE Transactions on Evolutionary Computation*, 6(2):182–197, April 2002.

[6] Ke-Lin Du and M. N. S. Swamy. *Search and Optimization by Metaheuristics: Techniques and Algorithms Inspired by Nature.* Birkh & Basel, 1st edition, 2016.

[7] Marco Laumanns Eckart Zitzler and Lothar Thiele. Spea2: Improving the strength pareto evolutionary algorithm. 2001.

[8] A. E. Eiben and James E. Smith. *Introduction to Evolutionary Computing.* Springer Publishing Company, Incorporated, 2nd edition, 2015.

[9] C.M. Fu, C. Jiang, G.S. Chen, and Q.M. Liu. An adaptive differential evolution algorithm with an aging leader and challengers mechanism. *Applied Soft Computing Journal*, 57:60–73, 2017.

[10] Nobuo Funabiki, Shiho Taniguchi, Yukiko Matsushima, and Toru Nakanishi. A proposal of a menu planning algorithm for two-phase cooking by busy persons. *Proceedings of the International Conference on Complex,*

Intelligent and Software Intensive Systems, CISIS 2011, pages 668–673, 2011.

[11] Nikolaus Hansen. Cma-es python package. 2017.

[12] T. Isokawa and N. Matsui. Performances in ga-based menu production for hospital meals. In *2015 IEEE Congress on Evolutionary Computation (CEC)*, pages 2498–2501, May 2015.

[13] Aynur Kahraman and H. Aydolu Seven. Healthy daily meal planner. In *Proceedings of the 7th Annual Workshop on Genetic and Evolutionary Computation*, GECCO '05, pages 390–393, New York, NY, USA, 2005. ACM.

[14] T. Kashima, S. Matsumoto, and H. Ishii. Evaluation of menu planning capability based on multi-dimensional 0/1 knapsack problem of nutritional management system. *IAENG International Journal of Applied Mathematics*, 39(3), 2009. cited By 4.

[15] Marco Laumanns. Spea 2 : Improving the strength pareto evolutionary algorithm. 2001.

[16] Coromoto Leon, Gara Miranda Valladares, and Carlos Segura. Metco: a parallel plugin-based framework for multi-objective optimization. *International Journal on Artificial Intelligence Tools*, 18:569–588, 08 2009.

[17] Xiaoliang Ma, Qingfu Zhang, Guangdong Tian, Junshan Yang, and Zexuan Zhu. On Tchebycheff Decomposition Approaches for Multiobjective Evolutionary Optimization. *IEEE Transactions on Evolutionary Computation*, 22(2):226–244, 2018.

[18] Gara Miranda, Eduardo Segredo, Juan Manuel Ramos, and Coromoto León. On the Planning of Healthy and Balanced School Lunches through Multi-Objective Evolutionary Algorithms. 2018.

[19] R. P. C. Moreira, E. Wanner, F. V. C. Martins, and J. F. M. Sarubbi. An evolutionary mono-objective approach for solving the menu planning problem. In *2018 IEEE Congress on Evolutionary Computation (CEC)*, pages 1–8, July 2018.

[20] Hea Choon Ngo, Yu-N Cheah, Ong Sing Goh, Yun-Huoy Choo, Halizah Basiron, Yogan Jaya Kumar, Hea Choon Ngo, Yu-N Cheah, Ong Sing Goh, Yun-Huoy Choo, Halizah Basiron, and Yogan Jaya Kumar. A Review on Automated Menu Planning Approaches. *Journal of Computer Science*, 12(12):582–596, dec 2016.

[21] Barbara Koroušić Seljak. Computer-based dietary menu planning. *Journal of Food Composition and Analysis*, 22(5):414–420, 2009.

[22] Barbara Koroušić Seljak. Computer-based dietary menu planning. *Journal of Food Composition and Analysis*, 22(5):414 – 420, 2009. 7th International Food Data Conference: Food Composition and Biodiversity.

[23] Sn Sivanandam and Sn Deepa. Genetic Algorithm Implementation Using Matlab. *Introduction to Genetic Algorithms*, 2008.

[24] N. Srinivas and Kalyanmoy Deb. Muiltiobjective optimization using nondominated sorting in genetic algorithms. *Evolutionary Computation*, 2:221–248, 1994.

[25] Suliadi Sufahani and Zuhaimy Ismail. A new menu planning model for Malaysian secondary schools using optimization approach. *Applied Mathematical Sciences*, 8(151):7511–7518, 2014.

[26] El-Ghazali Talbi. *Metaheuristics.* Wiley, 1 edition, 2009.

[27] M. Tian, X. Gao, and C. Dai. Differential evolution with improved individual-based parameter setting and selection strategy. *Applied Soft Computing Journal*, 56:286–297, 2017.

[28] Darrell Whitley. A genetic algorithm tutorial. *Statistics and Computing*, 1994.

[29] Qingfu Zhang and Hui Li. MOEA/D: A multiobjective evolutionary algorithm based on decomposition. *IEEE Transactions on Evolutionary Computation*, 11(6):712–731, 2007.

[30] L.M. Zheng, S.X. Zhang, K.S. Tang, and S.Y. Zheng. Differential evolution powered by collective information. *Information Sciences*, 399:13–29, 2017.

[31] Aimin Zhou, Bo-Yang Qu, Hui Li, Shi-Zheng Zhao, Ponnuthurai Nagaratnam Suganthan, and Qingfu Zhang. Multiobjective evolutionary algorithms: A survey of the state of the art. *Swarm and Evolutionary Computation*, 1(1):32 – 49, 2011.

[32] Eckart Zitzler and Simon Künzli. Indicator-based selection in multiobjective search. In Xin Yao, Edmund K. Burke, José A. Lozano, Jim Smith, Juan Julián Merelo-Guervós, John A. Bullinaria, Jonathan E. Rowe, Peter Tiňo, Ata Kabán, and Hans-Paul Schwefel, editors, *Parallel Problem Solving from Nature - PPSN VIII*, pages 832–842, Berlin, Heidelberg, 2004. Springer Berlin Heidelberg.

Appendices

Appendix A

Code

A.1 MOEA/D

The code of the MOEA/D algorithm can be found in the following Github repository.