Chapter 1

Introduction

1.1 Research Context

As the world becomes increasingly globalised, the content produced in languages other than that of the mother tongue increases, especially that in the English language, which turns out to be the dominant worldwide. The need to learn this language dramatically increases as time progresses, and traditional teaching methods do not provide enough for the entire population that needs to learn it. It is thus necessary to create alternatives that are more accessible and optimal for students who are often occupied by a very active pace of life and have little flexibility in their schedule to practice among their acquaintances.

One alternative to help learn a new language consists of a system in which students who do not know each other are grouped according to their topics of interest. In this manner, students practice the use of language according to having conversations about these interests and a similar level of experience, allowing all participants to understand and take part.

This thesis aims to determine an optimal manner to group students into different virtual groups so that they can improve their use of language and not be frustrated either because of the unattainable difficulty or because the topics do not suit their liking. There are several aspects that can make one grouping better than another; for example, the level of language proficiency must be relatively compatible, that is, very advanced students cannot be put together with beginners, as the advanced student gets bored with the difficulties of the beginner, and the latter does not understand what the advanced say. Additionally, it is sought that the interests of the students are compatible, in order to generate conversations of interest to all. For example, while some students may have sports among their subjects of interest, some prefer football and others prefer swimming. Another criterion is the intervention style of each student, which can be more or less active and more or less shy. Thus, if many shy and passive students are in the same room, it will be difficult to have an agile dynamic that makes the session entertaining. For this reason, it is proposed to address the problem as one of multi-objective optimisation, where the topics of conversation and the level of the student experience are the objectives to optimise. To this end, a mono-objective solution that combines the above criteria in a weighted aggregate function will be implemented. Then a comparison will be made of the results obtained with different genetic algorithms based on the mono-objective solution and

the multi-objective solution. It is believed that through the use of a multi-objective strategy, the student's experience will improve significantly compared to existing methods of teaching and practising foreign languages.

1.2 Problem Definition

The problem is defined as a variation of the Stable Roommate Problem (SRP), which is a variation of the Stable Marriage Problem (SMP) [**?**]. Only in this case, individuals do not know each other, so they have no real preference; nevertheless, the preference is predicted based on the similarity of the level of experience in the foreign language of the student and the interest that is shared with the rest of the group. The above is based on the fact that these two factors directly affect the student's preference to belong to the group.

In addition to this, group size and participation styles are also considered as restrictions for the benefit of the rest of the system, as a person may prefer to belong to a group that has more in common with him/her; however, his/her way of participating can alter the behaviour of the rest of the group. For example, an outgoing student who participates too much in a conversation can prevent the rest of the group from participating; conversely, a group in which no one participates needs an outgoing student to initiate the conversations.

In the same way, the size of the group is defined dynamically to obtain homogeneous participation among its members. It is defined dynamically since the absence or presence of some students can affect the performance of the group, although its size is not considered ideal.

This research aims to obtain the most stable solution possible concerning the predicted preferences of students within a reasonable time. Although there are many examples and variations of SRP in the literature, there is no characterisation that adapts directly to the problem defined considering the groups as variables. SRP is considered as a NP-complete problem, so it is relevant to find a solution in different by the use of heuristics and meta-heuristics. A comparison between single-objective algorithms and multi-objective is suggested to find a good enough solution for this problem.

1.3 Objectives

Compare single-objective and multi-objective strategies to find stable studying groups based on their interests and levels of experience, and consider the group size and their participation as constraints.

The specific objectives that this entails are described below:

1. Define stability and near-stability for the problem.
2. Build an artificial data-set to compare the potential approaches.

3. Define the relevant attributes and parameters to formulate the single-objective and multi-objective functions.

4. Implement mono-objective strategies to generate solutions to the problem, which will be the basis for comparing the multi-objective solutions that are made.

5. Implement and test different multi-objective strategies and compare them with the results obtained with the mono-objective strategies.

6. Review the results and make a report indicating if a better result was reached with a multi-objective strategy and what was the algorithm that had the best results.

1.4 Hypothesis

Multi-Objective algorithms can outperform single-objective algorithms in finding near-stable studying groups based on their interests and level of experience, while considering the group size and participation as constraints.

This research tries to respond the following questions:

- How stable are the solutions found?
- Which advantages multi-objective algorithms may have against single-objective algorithms and vice-versa?
- Is optimisation even necessary? (Can a random solution be improved using optimisation techniques?)
- How does single-objective solutions are, compared with multi-objective ones?
- Can a single-objective algorithm have a solution that belongs to the Pareto front?
- Which single-objective algorithm outperforms the other single-objective algorithms for this problem?
- Which multi-objective algorithm outperforms the other multi-objective algorithms for this problem?
- Which algorithm results to be the best overall?

1.5 Contributions

Below is shown a list of the main contributions of this research. They were established to achieve the objectives and answer the questions previously raised.

- Introduces the problem of Stable Student Groups (SSGP), based mostly in SMP and SRP and other similar combinatorial problems.
- Makes a comparison of optimisation of different preference criteria against the stability of the resulting groups.
- Establishes a way to solve SSGP with unknown preferences, using a predicted preferences based on a preference criteria.
- Defines a set of objective functions to help predict the preference of the students of a given data-set.
- Builds a synthetic data-set based on distribution data, determined by the objective functions previously defined.
- Defines a set of operators to be used for SSGP, based on small but significant changes to the groups.
- Presents different models for the solution of SSGP, and a way to compare them.
- Presents an extensive analysis of the solutions found making use of different ways to compare single-objective algorithms to multi-objective algorithms in a ways that are considered fair.

1.6 Solution Model

The solution for the established SSGP follows a structure for optimisation using evolutionary algorithms. First, the objective functions that represent the objective of the problem are defined. Then a synthetic database is generated using the distribution data for several sources. This data will be evaluated by the objective functions. After that, the operators for the evolutionary algorithms are defined and tested.

Finally, single-objective algorithms and multi-objective algorithms are compared within them and with each other. These comparisons follow closely the one based on the one developed by Ishibuchi et al. [**?**], who compared single-objective algorithms with multi-objective algorithms. A diagram of the solution model can be seen in Figure 1.1.

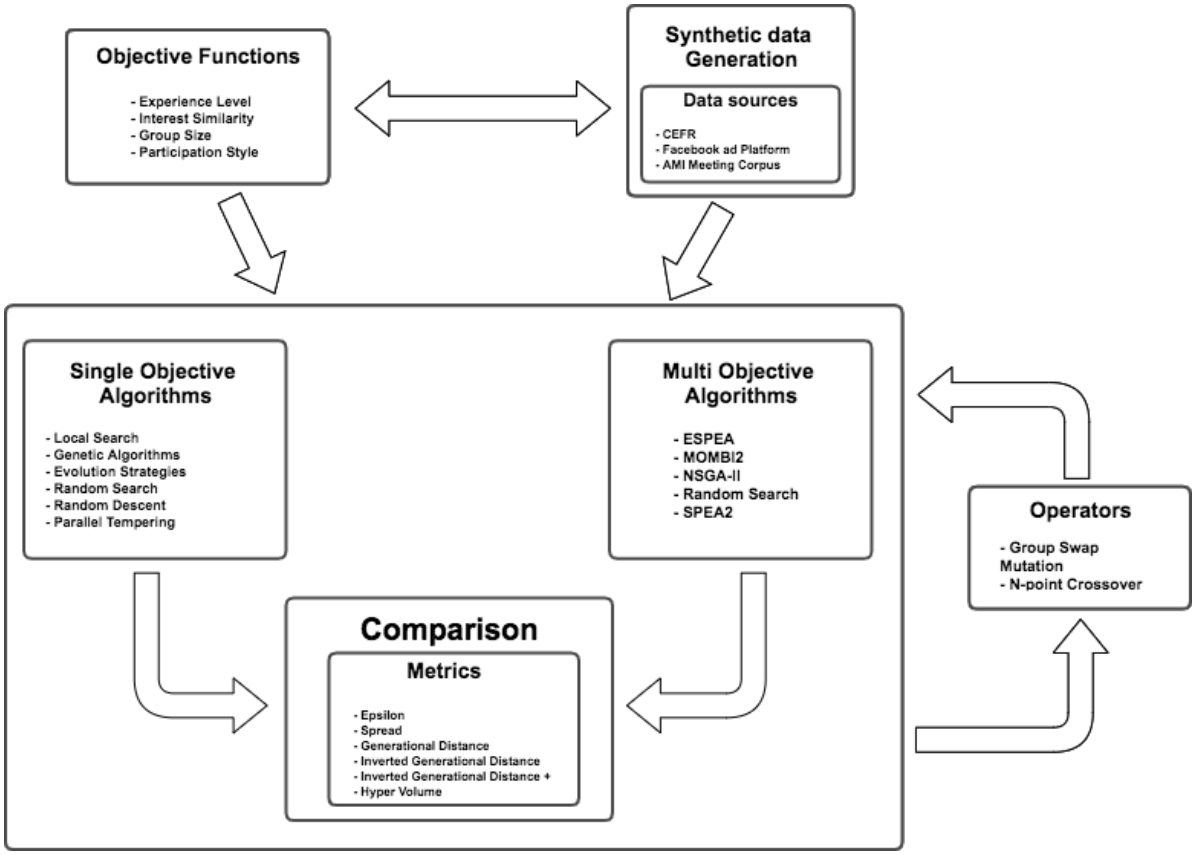

Figure 1.1: Solution model proposed to compare the approaches to solve SSGP.

1.7 Document Structure

Including this introduction, the thesis consists of 6 chapters.

Chapter 2 provides a review of the literature. Includes basic definitions for the single-objective algorithms and multi-objective algorithms to be used. Additionally, it includes some background for performing a comparison between them, including the different metrics used. This chapter also discusses the preference criteria, which is later used to define the objective functions. Finally, it includes a brief introduction of the frameworks used to evaluate the different algorithms.

Chapter **??** presents the design of the problem and the approach to solving it. It defines the objective functions to use and the operators for the different algorithms.

Chapter **??** shows a set of preliminary experiments. Beginning to test the defined objective functions. Then, a brief methodology for the data-set creation is explained and later tested with a set of exploratory experiments.

Chapter **??** contains the main experiments and a discussion of their results. The experiments

are separated in the setup and three phases. The setup introduces the parameters to be used for the rest of the experimentation. The first phase contains the first set of experiments using the parameters defined in the setup. Second and Third phases are the result of the feedback obtained during the first phase.

Chapter **??** explains the conclusions of the research and mentions the future work that can be derived.

Chapter 2

Background

This chapter presents a review of the literature for the considered single-objective algorithms and multi-objective algorithms. It also defines important concepts to understand the results produced, such as the concept of optimisation and dominance. After, it introduces the metrics that help compare the different approaches. Finally, it explains the criteria considered for preference prediction.

2.1 Optimality

There are two interpretations of optimality. One refers to single-objective optimisation and the other to multi-objective optimisation.

2.1.1 Single-Objective Optimisation

An optimisation problem refers to find the solution that gets the minimum or maximum value of a given function $f(x)$, subject to a set of constraints. Mathematically, it can be stated as follows:

$$\begin{aligned} &\min f\left(\boldsymbol{x_i}\right) \\ &\text{subject to } g_i\left(\boldsymbol{x_i}\right) \leq 0 \\ &\qquad\qquad h\left(\boldsymbol{x_i}\right) = 0 \\ &x_i \in \Omega \end{aligned} \tag{2.1}$$

where $\vec{x} = (x_1, ..., x_n)$ is an n-dimensional decision variable vector, $g_i(x_i)$ and $h_j(x_i)$ refer to the set of constraints that must be fulfilled while optimising $f(x_i)$, and Ω is the universe representing all the acceptable solutions possible that can be used to satisfy $f(x_i)$ and its constraints. The $\vec{x}$ decision vector can be whether continuous or discrete variables and its evaluation can also be continuous or discrete.

Definition 1. *Single-objective global optimisation is defined as given a function $f : \omega \subseteq \mathbb{R}^n \to \mathbb{R}, \Omega \neq \emptyset$ for $x_i \in \Omega$ the value $f^* \triangleq f(\vec{x}^*) > -\infty$ is called a **global minimum** if and*

only if

$$\forall x \in \Omega : f(\vec{x}^*) \leq f(x). \tag{2.2}$$

$\vec{x}^*$ is by definition the global minimum solution. The goal of determining the global minimum solution is called the **global optimisation problem** for a single-objective problem.

2.1.2 Multi-Objective Optimisation

Optimisation problems that involve two or more objectives are known as multi-criteria or multi-objective optimisation problems (MOPs). In MOPs, objectives are usually in conflict, this means that it is not possible to compare two solutions $\vec{x}$ and $\vec{y}$ directly, as there is no single solution that would simultaneously be the best for all objectives. The quality of the solutions is therefore compared by their dominance.

Definition 2. *The **Dominance** of a solution a_i of a set $A = (a_1, ..., a_n)$ is said to dominate a solution b_i of another set $B = (b_1, ..., b_n)$, denoted by $a_i \preceq b_i$ or $f(a) \preceq f(b)$ with, if and only if a_i is at least as optimal as b_i in all objectives of b_i and with at least one of its objectives better than b_i [**?**].*

In multi-objective optimisation, the goal is to find the set of optimal solutions, known as the **Pareto Optimal Set**. Pareto optimality is defined concerning the concept of non-domination between the points of the objective space Ω if and only if there is no vector in the set A that results better in set B.

Definition 3. *A **Pareto Optimal Set** (POS) is composed of optimal solutions, which its internal objectives cannot be improved simultaneously. Formally defined as:*

$$\mathrm{POS} := \{\mathrm{x}^* \in \mathcal{X} : \nexists \mathrm{x} \in \mathcal{X} \subset \mathbf{R}^n, \mathrm{x} \prec \mathrm{x}^*\} \tag{2.3}$$

Definition 4. *The **Pareto Optimal Front** (PF) is the corresponding objective space of vectors from the **Pareto optimal set** termed as non-dominated. This collection of Pareto optimal solutions being the result of the same MOP, where:*

$$PF^* := \{u = F(x) | x \in P^*\} \tag{2.4}$$

2.2 Evolutionary Algorithms

Evolutionary algorithms (EAs) are stochastic optimisation techniques inspired by Darwin's theory of evolution. An EA uses mechanisms inspired by biological evolution such as reproduction, mutation, recombination, and selection. The candidate solutions for optimisation are part of a population, and a fitness function determines the quality of each of these solutions. This fitness takes part after different iterations or generations in which it is sought to get as

close as possible to the global optimum [**?**].

Based on this fitness, the best candidates are more likely to be chosen as the parents for the next generation by applying recombination and mutation operators. Recombination is an operator that produces new candidates (children) solutions by combining the genetic information of two or more selected candidates (parents). A mutation operation applies small changes to a solution. Executing recombination and mutation leads to a set of new candidates (the offspring) that compete based on their fitness for a place in the next generation. This process is then iterated until a stop criterion is met.

EAs have several characteristics that can help position themselves within the families of generation and testing methods:

- They are population-based, this means that they process a variable number of candidate solutions simultaneously.
- They usually use a recombination to inherit the information of more candidates in new solutions.
- These algorithms are stochastic.

The main components of an EA are listed below:

- Representation (definition of solutions).
- Evaluation function.
- Population.
- Parent selection.
- Variation, recombination and mutation operations.
- Survivor selection mechanism.

The above mentioned components are defined to a particular problem. When solving a problem, EAs usually perform the following steps:

- **Initialisation**: A population of individuals (solutions) is generated at random.
- **Evaluation**: Individuals are evaluated using the fitness functions.
- **Termination**: A criterion to determine whether the algorithm should stop or not.
- **Selection**: Select the solutions that survive to the next generation.
- **Variation**: This step can combines the information of two or more individuals to produce a new offspring solutions through a crossover operator. Mutation operation, conversely, changes the genetic information for a single individual.

.

2.3 Single-Objective Optimisation Algorithms

This section describes the different single-objective optimisation algorithms (SOAs) that will be used for the comparison.

2.3.1 Local Search

Local Search (LS), also referred as Hill Climbing in maximisation problems and Steepest Descent in minimisation problems, is a SOA based on the idea of generating random starting solutions. Then, at each generation, LS generates the starting solution for the next iteration by perturbing the local optimum found thus far.

It is worthy to mention that the perturbing mechanism is an important feature of LS. A weak perturbation may not be enough to escape the local optimum. Conversely, a too strong perturbation may make the algorithm too similar to a multi-start LS with randomly generated starting solutions. A pseudo-code for this algorithm can be seen in Algorithm 1.

```
1 Choose, at random, an initial solution s in the search space;
2 Apply a search starting from s to get a solution s*;
3 repeat
4 |   Perturb s* to get a solution p;
5 |   Apply search starting from p to get a solution p*;
6 until the stopping criterion is satisfied;
```

Algorithm 1: Local Search Algorithm

2.3.2 Genetic Algorithms

Genetic Algorithms (GAs) were developed in the early 70s in Michigan by J. Holland and his students, who were researching adaptive systems at the time [**?**]. The definition can be very generic and most of their parts are usually implemented differently according to each problem. Their essential parts include a representation of the solution, a selection strategy, evolutionary operators, and a mutation operator.

In GAs, the binary string is the most common representation. For a binary representation, the mutation is performed by doing a bit-flip in a gene. Conversely, the crossover operator recombines the genes of two or more individuals, by exchanging half of the information from one parent and a half from the other one.

Finally, the replacement or survivor selection uses the fitness value to identify the individuals for the next generations and is responsible to assure the survival of the fittest individuals. An example of an implementation of a GA can be seen in Algorithm 2.

```
Generate Initial population p;
Compute fitness of each individual;

repeat
    // Produce a new generation;
    for size of p / 2 do
        Select two individuals from the previous generation for matching;
        Recombine the two individuals to give two new off springs;
        Insert the offspring in the new generation;
    end
until stopping condition has been met;
```

Algorithm 2: Genetic Algorithm

2.3.3 Evolution Strategies

Similar to the GAs, Evolution Strategies (ES) follow the principles of natural evolution as their method to solve optimisation problems. They were introduced during the 70s by Rechenberg [**?**] and further developed by Schwefel [**?**]. The first algorithm used in experimentation was two-membered ES and had a simple mutation-selection scheme.

The basic structure consists of a single parent which produces an offspring following a normally distributed mutation resulting λ children. Next, a selection operator determines the fittest individual which then becomes the parent of the next generation.

Later, Rechenberg proposed a multi-member ES [**?**], where more than one parent can participate in the generation of one offspring individual. This is denoted as $(\mu + 1)$-ES where μ represents the number of parents. Two other definitions were then introduced by Schwefel [**?**], which were $(\mu + \lambda)$-ES and (μ, λ)-ES.

- $(\mu + \lambda)$-ES, known as elitist evolution strategy, indicates that μ parents will create $\mu \geq 1$ descendants by recombination and mutation, keeping the fittest individual of the previous generation and discarding the rest.
- (μ, λ)-ES, known as non-elitist evolution strategy, is similar with the difference that none of the parents is kept in any of the subsequent generations, disregarding how good or bad was its fitness compared to the generation they belong.

1 *Initialise* the population with random individuals;
2 *Evaluate* each individual **repeat**
3 | *Select* parents;
4 | *Recombine* pairs of parents;
5 | *Mutate* the resulting offspring;
6 | *Evaluate* new individuals;
7 | *Select* individuals for the next generation;
8 **until** *the termination condition is satisfied*;

Algorithm 3: Evolution Strategy Algorithm

2.3.4 Random search

Random Search (RS), also known as Pure RS, was first defined by Brooks [**?**] and later named in the classic volumes by Dixon and Szegö [**?**]. RS is considered as one of the most basic search strategies, and it consists of the following:

- It creates a new random solution
- The best solution is updated if the new solution outperforms the fit value of the previous best solution.

While this algorithm sacrifices the guarantee of determining the optimal solution within a given error, it can be shown that RS converges to the global optimum with a probability one [**?**]. It is commonly used to compare the performance of other search algorithms to random sampling [**?**].

2.3.5 Random Descent

Random Descent (RD), also known as Stochastic Hill climbing, is an algorithm similar to regular Hill Climbing. However, RD includes a neighbourhood, which limits the search to only the solutions that are close to the current best solution at each step. The most popular RD implementation was done by Forrest and Mitchell, naming it Random Mutation Hill Climbing (RMHC) [**?**].

The local improvements are determined by a neighbourhood structure, which can be seen as an undirected graph G on vertex set S, and a fitness function on the search space of the algorithm. The algorithm checks a member of its neighbourhood randomly and transitions to this solution when there is an improvement in the fitness function result or is the same. The process is then repeated at each step.Algorithm 4 describes the pseudocode for RD.

```
Currrent solution s from new population;
Evaluate the objective function f(s);
stop = FALSE;
iter = 0;
while stop = FALSE do
    Choose a random solution in the neighbourhood s_n;
    if f(s_n) > f(s) then
        replace s by s_n;
    else
        iter = iter + 1;
    end
    if iter = iter_max then
        stop = TRUE;
    end
end
```

Algorithm 4: Pseudocode for stochastic hill climbing

2.3.6 Parallel Tempering (Replica Exchange Monte Carlo)

Parallel Tempering (PT), also known as Replica Exchange Monte Carlo, consists in replicas of a system of interest were simulated at a series of different temperatures. Then, these replicas undergo a partial exchange of configuration information. The more general form of PT with a complete exchange of configuration information was introduced in 1991 by Geyer [**?**].

This algorithm runs several replicas of a particular system of interest, using different temperatures in each one of them. High temperature can generate large volumes of phase spaces, while low-temperature systems tend to have more precision in the sampling of the phase space and tend to be trapped in a local optimum.

Using this algorithm requires a minimum and maximum temperature, with each replica of the system with an assigned temperature. After each step, the replicas are ordered according to their temperature, and then swapped to push the better solutions to the lowest temperatures, so they can eventually converge while higher temperature replicas are constantly modified in a search for improvement. Each of the replicas applies repeatedly a series of moves to a private solution, and then the global solution is tracked in the main search. The pseudocode for this algorithm can be seen in Algorithm 5.

for $i \leftarrow 0$ ***to*** $M-1$ **do**
  $T_i \leftarrow$ calculate temperature i;
  $replica_i \leftarrow$ new instance of Metropolis search with temperature T_i
end
repeat
  for $i \leftarrow 0$ ***to*** $M-1$ **do**
    Run $replica_i$ for N_{swap} iterations
  end
  $G \leftarrow min(Cost(global_best_{replica(0)}, ..., Cost(global_best_{replica(M-1)})))$;
  $j \leftarrow U\{0, M-1\}$;
  `// Randomly select adjacent temp. levels`
  $\Delta E \leftarrow (Cost(active\_solution_{replica(j)}) - Cost(active\_solution_{replica(j+1)}))/G$;
  $\Delta\beta \leftarrow 1/T_j - 1/T_{j+1}$;
  if $U(0,1) < min(1, exp(\Delta E \Delta\beta))$ **then**
    Set the temp. of $replica_j$ and $replica_{j+1}$ to T_{j+1} and T_j, respectively;
    swap($replica_j$,$replica_{j+1}$) `// Keeps replicas ordered by temp.`
  else
until *The stop criterion has been met*;

Algorithm 5: Parallel Tempering

2.4 Multi-Objective Algorithms

This section describes the different Multi-Objective Optimisation Algorithms (MOAs) that will be used for the comparison.

2.4.1 Multi-objective Random Search

Multi-objective RS is a probabilistic algorithm that serves mostly as an algorithm to benchmark other optimisation algorithms in the literature [**?**]. Just like its single objective counterpart, this produces a random set of solutions in each step. The output of the algorithm is the Pareto-optimal set of all solutions generated.

2.4.2 Electrostatic Potential Energy Evolutionary Algorithm

Electrostatic Potential Energy Evolutionary Algorithm (ESPEA) [**?**] is an algorithm that generates several Pareto front approximations, focusing only on the solutions that appear interesting to the decision-maker. Proposed in [**?**] after noticing that many of the algorithms found in the literature make use of popular crowding distance metrics that do not necessarily converge into optimality [**?**]. The algorithm attempts to design an electromagnetism inspired heuristic, where each solution has an assigned charge based on how close it is to a randomly selected subset from an archive of non-dominates solution. Then, the charges are translated to force vectors, which move the solutions in the search space. The best solutions are stored in an archive that uses clustering and crowding distance to prune the solutions.

```
Generate initial population P;
Copy all non-dominated solutions in P to archive A;
repeat
    Generate a single new solution p;
    Remove all solutions from A dominated by p;
    if p ⊁ a ∧ f(p) ≠ f(a) ∀a ∈ A then
        if |A| < N then
            A := A ∪ {p};
        end
        else
            Calculate e(a) for all a ∈ 𝒜;
            Calculate e := (e_{a^1}(p), ..., e_{a^N}(p));
            update(A, p, e);
        end
    end
until stop criterion has met;
return A;
```

Algorithm 6: ESPEA

2.4.3 Many Objective Metaheuristic Based on the $R2$ indicator

Many Objective Metaheuristic Based on the $R2$ indicator (MOMBI2), introduced first by Hernandez Gomez and Coello Coello [**?**]. It has an advantage over other many-objective optimisation algorithms found in the literature, as a lot of them relies on non-dominated sorting and crowding distance, which can become inefficient as the number of objectives grows.
The $R2$ indicator was introduced by Hansen and Jaszkiewicz [**?**]. The $R2$ indicator compares two sets of points being a reference Pareto front and the approximation of the front. MOMBI2 uses $R2$ because it induces the complete ranking in the set of all approximations. It is based on the assumption that it is allowed to add values of different utility functions in the set. Therefore, it is also dependent on the scaling of its utility functions. Algorithm 7 describes MOMBI2.

Require: MOP, stopping criterion,set of weight vectors W;
Ensure: Pareto set approximation;
Initialise population $P_i, i \leftarrow 1$;
Evaluate population $P - i$;
Calculate the $L_2 - norm$ of objectives for P_i;
Set $\vec{z}^{min} \leftarrow \vec{z}^{*}$ and $\vec{z}^{max} \leftarrow \vec{z}^{nad}$;
while *termination condition is not fulfilled* **do**
    Perform parent selection;
    Generate offspring P_i' using variation operators;
    Evaluate population P_i';
    Calculate the $L_2 - norm$ of objectives for P_i';
    Normalize objective functions for $P_i \cup P_i'$;
    Execute $R2$ ranking algorithm $(P_i \cup P_i', W)$;
    Reduce population $P_{i+1} \leftarrow \{P_i \cup P_i'\}$;
    Update reference points $(\vec{z}^{min}, \vec{z}^{max}, P_{i+1}, m)$;
    $i \leftarrow i + 1$
end
return P_i

Algorithm 7: MOMBI2

2.4.4 Non-dominated Sorting Genetic Algorithm II

Non-dominated Sorting Genetic Algorithm II (NSGA-II), proposed by Deb et al. [**?**], makes use of the Crowding Distance which is a certain preference for solutions that are less crowded in the solution space.

This algorithm works as follows. After an initial population is created, each individual is ranked and sorted according to their non-dominance level, each solution then is assigned a fitness according to their non-domination level (1 is assigned to the best, then 2 and so on). Then a binary tournament selection, crossover and mutation take place and an offspring population of equal size is created. The process is then repeated joining the previous generation and their offspring as the same population, and it continues with the population always taking the parents of the previous generation; therefore, elitism is ensured. It is important to notice that in the following generations, the ranking will also consider the crowding distance. The pseudocode for this algorithm can be seen in Algorithm 8.

Generate a random initial population P_t;
$R_t \leftarrow P_t \cup Q_t$;
combine parent and offspring population;
$F \leftarrow$ fast-non-dominated-sort(R_t);
run the fast non dominates sort procedure and collect $F = (F_1, F_2, ...)$ with the non-dominated fronts of R_t;
$P_{t+1} = 0$ and $i = 1$;
repeat
    crowding-distance-assigment(F_i);
    calculate crowding-distance in F_i;
    $P_{t+1} = P_{t+1} \cup F_i$;
    $i = i + 1$;
until $|P_{t+1}| + |F_i| \leq N$;
until the parent population is filled;
Sort$(F_i, \prec_n)$;
$P_{t+1} = P_{t+1} \cup F_i[1 : (N - |P_{t+1}|)]$;
$Q_{t+1} =$ make-new-pop(P_{t+1});
$t = t + 1$;

Algorithm 8: NSGAII

2.4.5 Strength Pareto Evolutionary Algorithm 2

Strength Pareto Evolutionary Algorithm 2 (SPEA2) [**?**], first introduced by Zitzler and Thiele in 1999, was among the first techniques that were extensively compared to several existing evolution-based methods [**?**].

SPEA2 makes use of an external non dominated set. For each individual in this set, a strength value is computed, which consist of the raw fitness value assigned from the objective functions and a density estimation similar to the ranking value of MOGA [**?**]. The algorithm applies the respective selection, crossover and mutation operators to fill an archive of individuals, then applies this fitness function and the non-dominated individuals from both the original population and the archive are copied into a new population. If the number of non-dominated individuals is greater than the population size, a truncation operator based on the distance to the k-nearest neighbour is used. Hence, the individuals having the minimum distance to every other are discarded. Pareto dominance is used to ensure that the solutions are properly distributed along the Pareto front. SPEA2 differs from its predecessor SPEA in three ways:

1. **Considering the fitness assignment**, the individuals that are dominated by the same archive members have identical fitness values. That means, in case that the archive has only one individual all population members will have an independent rank value, therefore lowering the selection pressure and making SPEA behave like a random search algorithm.

2. **For the density estimation**, if many individuals are indifferent to the current generation, which means there is no domination between them, there is little to no information

that can be obtained from them, which happens rather frequently, density information is used to guide the search more effectively. Clustering makes use of this information but only regarding the archive and not the population.

3. **Archive Truncation**, considers that, while in SPEA the clustering technique was able to reduce the non-dominated set, without destroying its features, it could lose outer solutions, however, these solutions must be kept in the archive to keep the spread of non-dominated solutions.

Then the fitness values are assigned to the archive and the population members.

- Each individual i is assigned a strength value $S(i) \in [0, 1)$, which at the same time considers the fitness function $F(i)$. $S(i)$ is the number of population members j that are dominated or equal to i, divided by the population plus one.
- The fitness $F(j)$ of an individual j in the population is calculated by summing the values of $S(i)$ of the rest of the archive members that dominate or are equal to j plus one.

Then for the mating selection, individuals are selected by binary tournaments. This assumes a minimisation, each member of the archive has a higher chance of being selected than the rest of the population. Finally, after crossover and mutation operators are applied, the population is replaced by the resulting offspring population. The pseudocode for this algorithm can be seen in Algorithm 9.

Input: *population size* N, *archive size* $\bar{N}$, *maximum number of generations* T;
Output: *nondominated set* **A**;
Initialize: Generate an initial population P_0 and create the empty archive $\bar{P}_0 = \emptyset$. Set $t = 0$;
repeat
    Fitness assignment: Calculate fitness values of individuals in P_t and $\bar{P}_t$;
    Environmental selection Copy all nondominated individuals in P_t and $\bar{P}_t$ to $\bar{P}_{t+1}$. ;
    if $size(\bar{P}_{t+1}) > \bar{N}$ **then**
        Reduce $\bar{P}_{t+1}$ by means of the truncator operator
    else
        Fill $\bar{P}_{t+1}$ with dominated individuals in P_t and $\bar{P}_t$
    end
until $t \leq T$ *or any stopping criterion is satisfied.*;
Termination: **A** $\leftarrow$ the set of decision vectors represented by the nondominated individuals in $\bar{P}_{t+1}$;
Mating selection: Perform binary tournament selection with replacement on $\bar{P}_{t+1}$ in order to fill the mating pool.

Algorithm 9: SPEA2

2.5 Comparing Single-Objective and Multi-Objective Optimisation

H. Ishibuchi et al. [**?**] presented a general study of this topic. The experiments are related to the comparison of single-objective GAs (SOGA) and multi-objective GAs (MOGA). In this work is also discussed the several difficulties occurring when comparing both types of algorithms, such as the inability to use hypervolume (HV) as a comparison metric, since it strongly depends on the location of the origin.

Different works include a comparison between single-objective and multi-objective strategies for different fields of the industry [**?**, **?**, **?**]. For example, Jiao and Zeng [**?**] among others make use of different strategies to convert a mono-objective problem into a dynamic multi-objective one in which apply a multi-objective evolutionary algorithm with improved results compared to existing solutions. Another recurring theme in which their performance is compared is in the design of antennas to improve cellular reception, as is the case of the work of Rahmat-Samil and Nanbo Jin. [**?**].

Finally, Battiti and Passerini [**?**] talk about the use of GAs that adapt to the decision-maker, this is relevant because in the final platform the user does not properly have a real decision on which group will end up being assigned but this is intended to be done automatically. Battiti mentions some of the possible strategies for adapting the behaviour of a decision-maker and also discusses some of the limitations that such a system would present according to their preferences [**?**].

2.6 Metrics

The next section includes metrics to use for comparison with multi-objective algorithms. The only metric that has shown to represent the domination of a set to another is the HV, however this metric is considered costly as the number of objectives and solutions grows and is usually considered to compare the performance between different algorithms, while the rest of the metrics serve as good approximations of the HV, that require less computational cost and can be calculated in each generation or step to choose a set of solutions over another. Besides there exist other sets of metrics such as spread that measure the distribution of the solutions within a given front, this helps determine the quality of the front because it considers a wider space covered within the search space.

2.6.1 Hypervolume

First defined by Zitzler and Thiele [**?**], as:

Definition 5. *The size of space covered S by a front $X' = (x_1, x_2, ..., x_k)$, as a set of points in a Front, then $S(X')$ gives the volume enclosed by the union of the polytopes $p_1, p_2, ...p_k$*

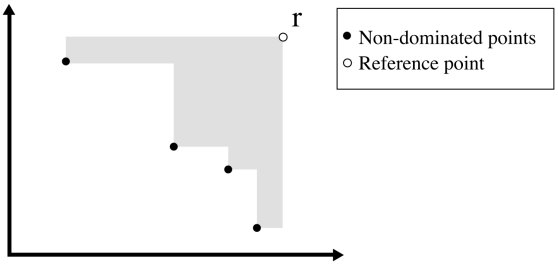

Figure 2.1: Demonstration of the (HV) in two dimensions, the grey area represents the value of $HV(S, r)$

where each p_i is formed by the intersection of the hyperplanes arising out of x_i along with the axes, for each of them in the objective space, there exists a hyperplane perpendicular to this axis and passing thought the point $(f_i(x_i), f_2(x_i), f_n(x_i))$. In the two dimensional case, each p_i represents a rectangle defined by the points $(0, 0)$ and $(f_1(x_i), f_2(x_i))$.

HV is considered the only unary indicator that is capable of truly detecting that a front A is not worse than another front B for all its pairs $A \triangleright B$ [**?**]. This concept is referred as *pareto complaint* [**?**]. It is also noticed that for all the points in a front A and a front B, $A \triangleright B$ follows $I_H(A) > IH(B)$, because this means that A must include at least one point that is not weakly dominated by B, therefore a portion of the objective space is covered by A but not by B.

This can be more clearly seen in Figure 2.1, wherein the former the grey area represents the $HV(S, r)$ in the search space.

One more general definition can be found in [**?**] where it is described as the volume of the space in the objective space dominated by an approximation of the Pareto Front S and delimited from the above by a reference point $r \in R^m$, so, for all the points z in the obtained front. The hyper-volume is given by:

$$HV(S, r) = \lambda_m \left(\bigcup_{z \in S} [z; r] \right) \tag{2.5}$$

Where λ_m is the $m-$dimensional Lebesgue measure.

One of the major drawbacks of using HV is that its complexity results in $\mathcal{O}(|S|^{m/2} log|S|)$ and is also the main reasons other measurements are preferred over HV especially in many-objective problems. It has been used without issues with the number of objectives lower than 10 and a non-dominated number of solutions less than 10000. However other methods and algorithms have been proposed to address this issue [**?**].

2.6.2 Spread

First defined by Deb et al. in [**?**], the metric spread often represented as $\triangle$. It measures the extent of spread achieved among the obtained solutions. Its goal is to get a set of solutions

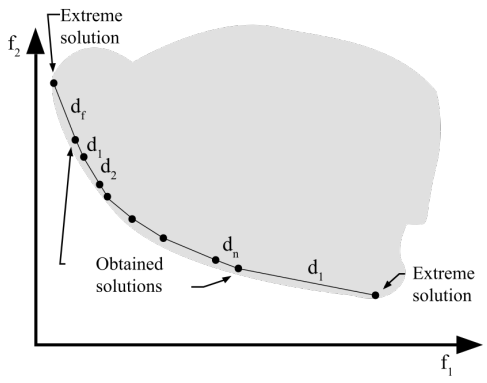

Figure 2.2: Graph representing the Spread metric.

that spans the entire Pareto-optimal region.
The euclidean distance d_i is calculated between each consecutive solution in and obtained front. Then the average $\overline{d}$ is calculated for these distances, so for a front composed of non-dominated solutions, first the extreme solutions of the objective space are obtained, these are considered the boundaries of the obtained solutions, this is done by fitting a curve parallel to the Pareto front. Afterwards, the spread is calculated using Equation 2.6.

$$\Delta = \frac{d_f + d_l + \sum_{i=1}^{|NDS|-1} \left| d_i - \bar{d} \right|}{d_f + d_l + (|NDS| - 1)\bar{d}} \tag{2.6}$$

Here d_f and d_l are the Euclidean distances between the extreme solutions, this can be better seen in Figure 2.2 where all the non-dominated solutions are found within the curve drawn.

There may exist a scenario where are solutions lay within the same point, this is not considered the worst case of spread. When there is a great variance between the distance of the solutions the increment can be greater than one. On the other hand, a good distribution would make all the distances d_i equal to the average $\overline{d}$, making d_f and d_t equal to zero, making $\triangle$ also zero. In the case of higher dimensions, a triangularization technique is used to calculate the distances for the solutions.

This metric was one of the most preferred in the literature. However, its usage has decreased in the recent years. One of the major drawbacks is that the metric alone only measures the diversity of the approximation to the Pareto front [**?**]. Nevertheless, combined with other convergence metric is still a relevant approach to assess the quality at a low computational cost. It should also be noted that it remains as the most popular metric to measure the diversity, but there is no more interest in the literature to measure the diversity alone to determine a quality set of solutions.

2.6.3 Epsilon

First defined in by Zitzler and Thiele in 2003 [**?**]. For a long time, it was considered the main indicator to describe the domination of the solutions [**?**]. It consists in getting a factor ϵ from which if an objective front are multiplied the Pareto front or any another front still dominates the former.

In the case of minimisation, a vector $\vec{z_1}$ is said to $\epsilon-$dominate another vector $\vec{z_2}$ if and only if for all the points of the vector $\vec{z_2}$ multiplied by a factor of ϵ remains higher than $\vec{z_1}$, this can be best described in Equation 2.7.

$$I_{\in}(A, B) = \inf_{\epsilon \in R} x\{\forall \vec{z_2} \in B \exists \vec{z_1} \in A : \vec{z_1} \succeq_{\in} \vec{z_3}\} \tag{2.7}$$

Loosely speaking, a vector $\boldsymbol{z}$ is said to epsilon dominate another vector if the multiplication of each objective value in the other vector by a factor of e and the resulting objective vector is still weakly dominated by the first. The resulting Epsilon is the E precise factor.

According to this the ϵ-indicator gives the factor by which an approximation set is worse than another with respect to all its objectives, therefore $I_{\in}(A, B)$ equals to that minimum factor ϵ such that any objective in B multiplied by the factor ϵ is ϵ-dominated by at least one objective vector in A. For a single objective evaluation $I_{\in}(A, B)$ is simply the ratio between the two objectives A and B.

For example, in Figure 2.3, the dark blue area represents the subspace ϵ-dominated by the solutions in A_1 for an $\epsilon = 9/10$, the light blue area refers to the subspace ϵ-dominated by the solutions in A_1 for an an $\epsilon = 4$.

2.6.4 Generational Distance

Generational Distance (GD), was first used by Van Veldhuizen and Lamont in the early experiments done in 1998 [**?**]. GD is a value that consists in representing how far a known Pareto front is from the real Pareto front defined in Equation 2.8.

$$G \triangleq \frac{(\sum_{i=1}^{n} d_i^p)^{1/p}}{n} \tag{2.8}$$

where n is the number of solutions in the known Pareto front and d_i is the Euclidean distance in the objective space between each of the solutions and the nearest member of the real Pareto front, and p representing the number of objectives. When the known and the real Pareto front are the same the distance should be zero. Any other result shows how much the known Pareto Front deviates from the real one.

GD has been lowering its presence in the literature in recent years, as HV has been increasing is usage [**?**]. One of its main drawbacks in comparison to other metrics is that if a known front

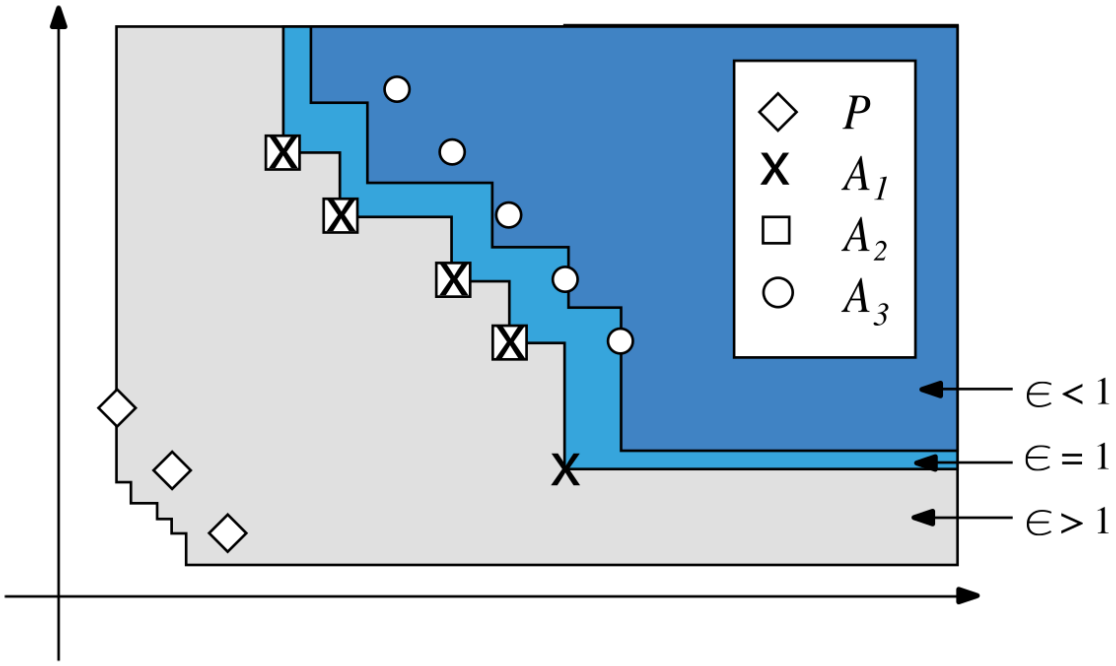

Figure 2.3: Graph representing the subspace of which Epsilon is based.

presents a large fluctuation in the distance values the calculated value may not represent the true distance.

2.6.5 Inverted Generational Distance

Inverted GD (IGD) [**?**] can be calculated as the inverse of the GD starting with the Real Pareto Front and calculating the distance to the fronts produced by their tested algorithms [**?**]. It can also be calculated by the following equation:

$$D\left(A, P^{*}\right) = \frac{\sum_{v \in P^{*}} d(v, A)}{|P^{*}|} \tag{2.9}$$

Where $d(z_i, a_j)$ is the distance be a distance between z_i and a_j in the objective space. It is mentioned by [**?**], that If $|Z|$ is large enough to represent the Pareto Front well, then $D(A, P*)$ can be used to measure both the diversity and the convergence.

There are two main advantages of using IGD, the first is its computational efficiency even considering multi-objective problems with more than four objectives, called many-objective problems, especially comparing it to HV. The other is its generality, it usually shows the overall quality of an obtained front, considering its convergence and to it and its diversity. This is the main reason why its often used in the found literature [**?**].

However, besides the advantages gained from IGD it still has several disadvantages which are discussed in the following subsection.

2.6.6 Inverted Generational Distance +

While IGD is increasing its use, there are several disadvantages mentioned in [**?**], one of these known flaws, especially comparing it to the HV is its lack of Pareto compliant property, this can be further be seen in Figure 2.4

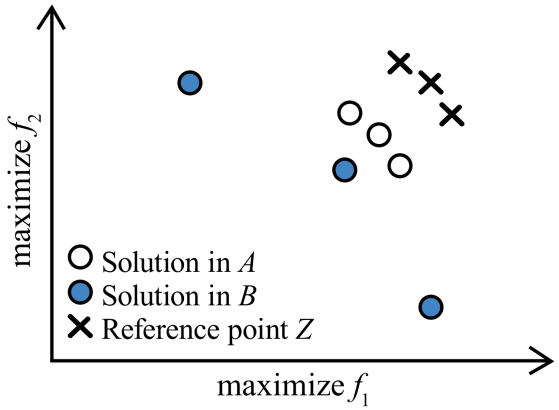

Figure 2.4: Example of distance between Reference Z, Solution A and Solution B.

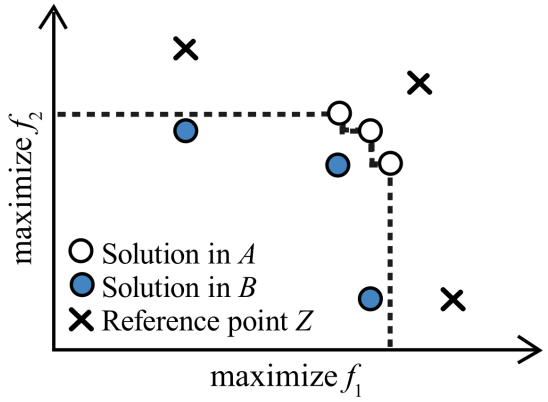

Figure 2.5: Example of distance between Reference Z, Solution A and Solution B, here the solutions from set B appear to be closer to Z.

This presents a comparison between the front $A = (a_1, a_2, a_3)$ and the front $B = (b_1, b_2, b_3)$, and a reference Pareto front $Z = (z_1, z_2, z_3)$ for a maximisation problem. Here every solution in the front B is clearly dominated by the solutions in the front A, however the measurement of IGD between Z and B would be lower, because of the way the solutions in B are distributed which has a similar spread to Z.

To address these drawbacks, IGD+ was proposed by Ishibuchi and Masuda [**?**]. It is closely similar to IGD with the difference of having a weak Pareto compliant property, and it is based on the idea of calculating the distance from each reference point to a dominated region instead of the nearest solution, this can be seen in Figure 2.6.

Then, ($IGD+$) is defined as the distance between a Pareto front Z and a solution front A.

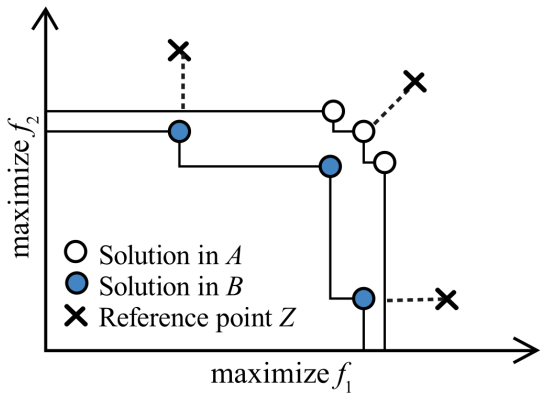

Figure 2.6: A graph representing the main idea behind ($IGD+$).

Here, if a solution A is dominated by the reference front Z, then $d_{IGD+}(A, Z)$ is the same as the euclidean distance $d(A, Z)$, because all the solutions in the front A are smaller than those from the reference front Z, and this also holds for each point in their respective fronts. When A and Z are not dominated with each other, only the objectives in A that are lower to their counterparts in Z are used, which means that if a point in the front A is better than another from point Z with respect to some of its objective, then these are not used in the calculation of IGD+.

2.7 Frameworks

2.7.1 jMetal

jMetal is a framework based on the Java programming language. It is focused on development, experimentation and study of meta-heuristics to solve MOPs. JMetal includes various of the most common algorithms. A series of example problems and regularly used indicators to measure the performance of algorithms. It also includes tools to carry out experimental studies that can be configured to generate statistical reports of the obtained results. It is also possible to take advantage of a multi-core architecture to make the experimentation process faster [**?**].

Some of the features sought are part of this type of software are:

- Include the most current algorithms available

- Contain the most accepted performance tests for multi-objective problem solving
- Give quality indicators to measure the performance of each of the tests and assist in a more accessible way in the investigations of its users.

2.7.2 James

James, which stands for Java Metaheuristics Search, is an object-oriented Java framework for discrete optimisation using local search algorithms that takes advantage of the generality of such meta-heuristics by separating search implementation and application from each problem specification. A wide range of generic local searches is provided, including: (Stochastic) Hill climbing, Tabu search, Variable neighbourhood Search and Parallel Tempering. These can be applied to any user-defined problem by designing a custom neighbourhood for the corresponding solution type. Using an automated analysis workflow, the performance of different search algorithms can be compared to select an appropriate optimisation strategy [**?**].

Implementations of specific components are included for subset selection, such as a predefined solution type, generic problem definition and several subset neighbourhoods used to modify the set of selected items. In comparison with existing Java meta-heuristics frameworks that mainly focus on population-based algorithms, JAMES has a much lower memory footprint and promotes the efficient application of local searches by taking full advantage of move-based evaluation and parallelization.

2.8 Preference Criteria

This section describes the different metrics used as a basis for the objective functions based on the predicted preference of the students according to the group they want to belong.

2.8.1 Benne and Sheats functional roles

The Benne and Sheats functional roles are a tool that helps define what kind of reaction a person has at the time of starting a conversation [**?**]. The roles we are specifically interested in are known as the **Maintenance Roles**.

- **Attacker**: Is that person who invalidates the comments of others and ”attacks” the rest of the group members.
- **Protagonist**: is the person who initiates the conversations, assumes an authority role. In this case, it is a very valuable role in our research.
- **Follower (or Supporter)**: This person has a cooperative attitude, pays attention to the talk, accepts the proposals and offers technical support.
- **Gatekeeper**: is a person who plays the role of moderator, checking that each person has their opportunity to participate similarly.

- **Neutral**: It is similar to the follower role, only instead of actively participating, it accepts the comments passively.

It should be noted that a person can represent different types of roles in the same conversation, although it is generally classified depending on which was their most prominent role.

AMI Meeting Corpus

AMI meeting corpus is a data-set containing the recording of several conversations in different contexts. With the purpose to analyse the interaction between the participants. It has been used in several works to predict participation styles such as Vinciarelli [**?**] who uses this database to identify the functional roles of Benne and Sheats [**?**].

2.8.2 Facebook Ad Platform

The social network known as Facebook is useful to maintain contact between friends and acquaintances, however, it is also a useful marketing tool to give users who belong to certain market ads focused on each of them according to their preferences, behaviours and locations. This follows a marketing methodology known as Demography, Behaviours and Interests. As far as this research is concerned, we are only interested in the Interests.

These interests are determined by the classification of the pages or groups that the user follows, however it has a more general classification as an ontology that Facebook has been building over the years using different clustering techniques so that It can have a more general categorisation of the different types of interests of each of the users.

Besides, Facebook allows businesses through a platform known as **Facebook ad Platform** [**?**] publish different types of ads aimed at a type of market according to the interests or behaviours corresponding to a market. This platform is public and can be used to determine an approximate cost when preparing marketing campaigns. However, this research is used to determine the population size for each of the interests described as it will be mentioned in Sections **??** and **??**.

This platform can also be used at the same time to generate the distribution of the data since it is directly linked to the ontology and can be consulted to measure the distances between one interest and another. Part of the ontology can be seen in Figure 2.7.

The ontology that Facebook has created is determined with a tree of 8 general and 314 specific interests. Each classified according to the relation to the general interest. This allows to search for specific interests in common, for example, if a person likes television programs we can also assume that she may also like reality shows or contest programs etc. In this way, another person who has an interest in television series has a direct relationship with the first. On the other hand, interests that have no relation, have no connection in the ontology.

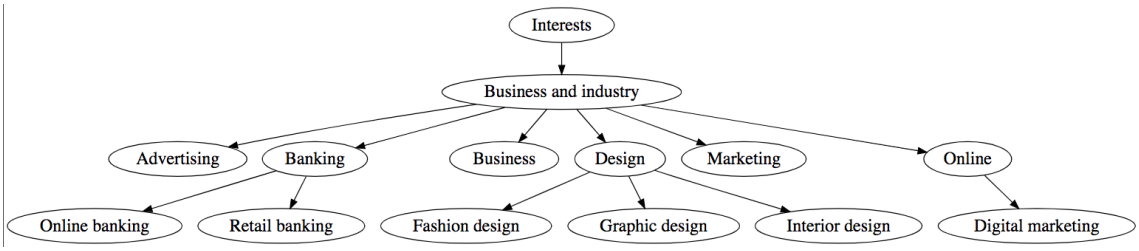

Figure 2.7: An example of the Facebook interests ontology.

2.8.3 CEFR

Known as **Common European Framework of Reference for Languages** for its acronym in English, it is a tool used internationally to assess the level of knowledge of a foreign language and is commonly known throughout the world in such a way that although a different test has been performed, it is possible to find a type of equivalence using this classification, which allows being able to take into account at the level of a person regardless of what type of test taken.

The levels are classified as A1, A2, B1, B2, C1 and C2. The first two levels, A1 and A2, refer to the most basic levels where a person knows the basics of the language, such as introduce themselves or asking for directions. The second tier, B1 and B2 are considered the intermediate level, people in this classification can understand long casual conversations as well as writing and reading non-specialised content. The last two levels C1 and C2 refer to people with the ability to deeply understand and speak the language, moreover, to write full documents using it.