New prediction and planning for digital learning based on optimization methods

Mounir Hafsa

UNIVERSITÉ DE LILLE
ÉCOLE DOCTORALE MADIS-631: MATHÉMATIQUES,
SCIENCES DU NUMÉRIQUE ET DE LEURS INTERACTIONS

New prediction and planning for digital learning based on optimization methods

Nouvelle prédiction et planification pour l'apprentissage numérique basées sur des méthodes d'optimisation

Mounir Hafsa

Thèse préparée et soutenue publiquement le **27/01/2023,**
en vue de l'obtention du grade de **Docteur en Informatique et Applications.**

Membres du jury:

Mme. Armelle Brun	Prof., Université de Nancy	*Rapporteur*
M. Jean-Charles Billaut	Prof., Université de Tours	*Rapporteur*
M. Lhassane Idoumghar	Prof., Université de Haute-Alsace	*Examinateur*
Mme. Elsa Negre	MCF (HDR), Univ. Paris-Dauphine	*Examinateur*
M. Jean-Christophe Routier	Prof., Université de Lille	*Président de jury*
Mme. Laetitia Jourdan	Prof., Université de Lille	*Directrice de Thèse*
Mme. Julie Jacques	MCF, Université de Lille	*Co-encadrante de Thèse*
Mme. Pamela Wattebled	Mandarine Academy	*Co-encadrante de Thèse*

Centre de Recherche en Informatique, Signal et Automatique de Lille
Université de Lille - Bâtiment ESPRIT - Avenue Henri Poincaré
59655 Villeneuve d'Ascq Cedex FRANCE

Acknowledgement

I'd like to begin by thanking the reviewers of my work, Armelle Brun and Jean Charles Billaut, for providing suggestions and comments that helped to improve my research. This appreciation extends to the other members of my thesis jury, Lhassane Idoumghar, Elsa Negre and Jean-Christophe Routier for their keen attention and propositions for the future of this work.

Pamela Wattebled, my company's supervisor, for her unwavering support. Whether for life advice in difficult times or career insights when in doubt. Most importantly, her trust and faith in me was the primary motivator in overcoming the various difficulties I encountered during the thesis.

Laetitia Jourdan and Julie Jacques, my academic supervisors. I'm grateful for their help during difficult times, especially at the start of the thesis during the COVID-19 pandemic. Their ideas, expertise, and assistance made this work much richer, particularly the weekly meetings, which were always fruitful. I'm proud to have learned from such inspiring supervisors and honored to have collaborated on several papers. I look forward to working with you again in future research opportunities.

Mandarine Academy's R&D team, led by Eddy Guerin. His technical tips are always helpful, and his counter-strike skills are incredible. Jeremy Morel and Anne Sophie Lesaint for their assistance with the DiLeap Logistic project, as well as their insights on food, places, and history. Siegfried Desmedt for his assistance in testing and validating our work at Mandarine Academy and its partners to solve the timetabling problem. Tim Chevalier and Manuel Leveugle for their technical advices, positivity, fun moments, and love of kebab. Alan Damanti and Laurent Maurer for providing technical assistance with the recommendation system problem (Mission to Mars).

The ORKAD team at Cristal research unit, for their assistance and academic insights, whether for experimental setups, scientific writing recommendations, or rehearsals: Weerapan Sae-Dan, Szczepanski Nicolas, Sara Tari, Meyssa Zouambi, Yousra Badji, Adán José-García, Nesrine Harbaoui, Thomas Feutrier, Agathe Metaireau, Clément Legrand, Lucien Mousin, Nadarajen Veerapen, Clarisse Dhaenens, Julien Baste, Marie-Eléonore Kessaci-Marmion.

Finally, I'd want to dedicate this work to my father's soul, who passed away shortly after this thesis began during the pandemic. During difficult times, the love and support of my mother, Jamila Braham, my father, Mohamed Hafsa who kept supporting me until his last breath, my brother,

Othman, and sisters, Asma and Noura, motivated me to push further.

Although we are separated by long distance, I am grateful to my family for their emotional backing, advice, and life lessons that have enabled all of this to happen.

My closest friends that made a positive impact on my life at various stages: Salah Belkhiria, Nour Majdoub, Hamdi Ben Abdeljelil, Hamza Karifa, Adam Laarif, Nader Chatti, Hosni Mansour, Ahd Khalifa, Ghassen Ghazzeh, Abdel-Basset Chaouch, Mehdi Abdallah.

Aslihan Sumer, for bearing the burden of late nights of bibliography preparation and code debugging. She was the one who kept rooting for this achievement even during the most difficult parts of this thesis and the pandemic.

I'm grateful for such friendships, whether in France, Tunisia, or any other country, especially through tough times, losses, breakups, or when I was down.

Abstract

This thesis addresses two distinct problems: the schedules of professional training courses and recommendation systems. Both problems occur at Mandarine Academy, a French educational technology company that specializes in innovative corporate training techniques such as personalized online learning platforms, training logistics, web conferences, etc. We begin with the scheduling problem related to managing training logistics. The company offers a tool called "Dileap Logistic" that automatically assigns resources (rooms, teachers, and equipment) to time slots (days and hours) at specific locations. Previously, this was done manually, which took a lot of time and yielded inaccurate results (wrong trainers, wrong days, or wrong equipment). Although it is an NP-Complete problem, a mathematical formulation of the problem was developed after examining the literature and the company's requirements. It includes 18 constraints (hardsoft) and 5 objectives, two of which are concurrent. We evaluate 5 multi-objective evolutionary algorithms (MOEAs) starting with the non-dominated sorting genetic algorithm (NSGA II and NSGA III), the decomposition-based multi-objective evolutionary algorithm (MOEAD), the indicator-based evolutionary algorithm (IBEA), and finally the Pareto Strength evolutionary algorithm (SPEA 2). Two customized genetic operators (for Mutation and Crossover) have been proposed and compared to classical operators (PMX and Swap mutation). A tuning phase involving all the algorithms mentioned above is performed to obtain elite configurations. Experiments are divided by problem size (small, medium, and large instances) with 3 to 5 objectives tested. We discuss results such as the performance comparison of each algorithm and convergence graphs.

In addition to providing solutions for training logistics, the company creates daily online educational content (videos, quizzes, documents, etc.) to support the digitization of work environments and follow current trends. With over 550K users spread across 100 learning platforms, the company faces challenges such as information overload and lack of motivation among users. Mandarine Academy gave us access to one of its public learning platforms to conduct our research. After an analysis of the literature and an in-depth analysis of user data, we propose improvements to the overall user experience and graphical interface.

The second part of this thesis addresses the problem of recommendation systems to enhance the user experience on learning platforms. We have mathematically formulated a multi-objective optimization problem with 5 objectives (Similarity, Diversity, Novelty, Root Mean Square Error (RMSE), and Normalized Discounted Cumulative Gain (NDCG)) and

implemented various recommendation techniques to generate initial solutions. All the evolutionary algorithms mentioned above (NSGA II, NSGAIII, IBEA, SPEA2, and MOEAD) are evaluated in our experiments. Real-world implicit interactions in the form of viewing time are chosen to make predictions. Tests were run on two different groups based on user behavior. Both groups are used in the parameter tuning phase and the final experiments. The initial results of various objectives are promising, considering the production mode scenarios and the choice of genetic operators. We further discuss performance graphs and conclude with future work.

Abstract

Cette thèse aborde deux problèmes distincts : les emplois du temps des formations professionnelles et les systèmes de recommandation. Les deux problèmes ont lieu à Mandarine Academy, une société française de technologie éducative qui se spécialise dans les techniques de formation d'entreprise innovantes telles que les plateformes d'apprentissage en ligne personnalisées, la logistique de formation, conférences Web, etc.

Nous commençons par le problème des emplois du temps lié à la gestion de la logistique de formation. L'entreprise propose un outil nommé « Dileap Logistic » qui gère automatiquement l'affectation des ressources (salles, enseignants et équipement) aux créneaux horaires (jours et heures) à des endroits précis. Auparavant, cela se faisait manuellement, ce qui prenait beaucoup de temps et donnait des résultats inexacts (mauvais formateurs, mauvais jours ou mauvais équipement). Bien qu'il s'agisse d'un problème NP-Complet, une formulation mathématique du problème a été élaborée après avoir examiné les ouvrages de littérature et les exigences de l'entreprise. Il comprend 18 contraintes (hard/soft) et 5 objectifs, dont deux sont concurrents. Nous évaluons 5 algorithmes évolutifs multi-objectifs (AEMO) en commençant par l'algorithme génétique de tri non dominant (NSGA II et NSGA III), l'algorithme évolutif multi-objectif basé sur la décomposition (MOEA/D), l'algorithme évolutif basé sur des indicateurs (IBEA) et enfin l'algorithme évolutif de Pareto Strength (SPEA 2). Deux opérateurs génétiques personnalisés (pour Mutation et Crossover) ont été proposés et comparés à des opérateurs classiques (PMX et Swap mutation). Une phase de réglage impliquant tous les algorithmes mentionnés ci-dessus est effectuée pour obtenir des configurations d'élite. Les expérimentations sont divisées par taille de problème (petites, moyennes et grandes instances) avec 3 à 5 objectifs testés. Nous discutons de résultats tels que la comparaison de la performance de chaque algorithme ainsi que des graphes de convergence.

En plus de proposer des solutions à la logistique de la formation, l'entreprise crée quotidiennement du contenu pédagogique en ligne (vidéos, quiz, documents, etc.) pour soutenir la numérisation des environnements de travail et suivre les tendances actuelles. Avec plus de 550K utilisateurs répartis sur 100 plateformes d'apprentissage, l'entreprise fait face à des défis tels que la surcharge d'information et le manque de motivation chez les utilisateurs. Mandarine Academy nous a donné accès à l'une de ses plateformes publiques d'apprentissage pour mener nos recherches. Après une analyse de la littérature et une analyse approfondie des données d'utilisateurs, nous

proposons des améliorations à l'expérience utilisateur globale et à l'interface graphique.

La deuxième partie de cette thèse aborde le problème des systèmes de recommandation pour améliorer l'expérience utilisateur sur les plateformes d'apprentissage. Nous avons formulé mathématiquement un problème d'optimisation multi-objectif ayant 5 objectifs (Similarité, Diversité, Nouveauté, Erreur quadratique moyenne racine (RMSE) et Gain cumulé actualisé normalisé (NDCG)) et mis en œuvre différentes techniques de recommandation pour générer des solutions initiales. Tous les algorithmes évolutifs mentionnés ci-dessus (NSGA II, NSGAIII, IBEA, SPEA2 et MOEA/D) sont évalués dans nos expériences. Les interactions implicites du monde réel sous forme de temps de visionnage sont choisies pour effectuer des prédictions. Les tests ont été exécutés sur deux groupes différents en fonction du comportement de l'utilisateur. Les deux groupes sont utilisés dans la phase de réglage des paramètres et les expérimentations finales. Les premiers résultats de divers objectifs sont prometteurs, compte tenu des scénarios de mode de production et du choix des opérateurs génétiques. Nous discutons davantage des graphes de performance et concluons avec les travaux futurs.

Contents

List of Figures

List of Tables

Chapter 1

Industrial Context and Motivations

Contents

This chapter introduces the company "Mandarine academy" and critically analyzes its products. Our thesis addresses two different problems related to two different commercial products maintained by the company and available for public use.

1.1 Mandarine Academy

Mandarine Academy is an Ed-Tech company created in 2008, it supports the digital transformation of international companies by facilitating the use of new technologies by all employees. Thanks to an exclusive approach that combines a digital platform with personalized support and individual meetings, Mandarine Academy offers a new way of training that is more effective in terms of skills, capacity, time, and budget.

Mandarine Academy is committed to making digital tools accessible to everyone. They make sure employees will be able to effectively use the company's tools and its best practices. Thus, they focus on client support so that each company can achieve its goals. Currently, Mandarine Academy has accompanied more than 3000 clients and more than 1 million users. Multiple collaborations with the private and public sectors have been established, like Microsoft and the French Ministry of Labor.

Since its creation, the company has been offering video-conference training to users on their collaborative tools, office automation, telephone, or business applications. Thanks to a custom-made logistics and training organizational tool, deploying training can be done in a matter of just 48 hours. Aside from video conferences, e-learning video content is tailored for each client and adapted for each profession.

Within the following subsection, we showcase the products involved in our work: DiLeap-Logistic for managing the training logistics, and MOOC-OFFICE 365 for e-learning recommendations.

Figure 1.1 Mandarine Academy Logo.

1.1.1 DiLeap-Logistic

DiLeap is a new solution developed by Mandarine Academy that aims to provide companies with a fully personalized SaaS training platform. It provides a personalized platform with the integration of web conferencing training sessions. Also, pedagogical content management with full integration of Microsoft Office 365 services and MOOCs (Massive Open Online Courses). The platform provides access for content creation to any specific training with an administration and reporting console. Another DiLeap variant is in the works right now with the main purpose to manage training logistics. The new solution is called DiLeap Logistic, a shared platform created by Mandarine Academy and used by multiple companies including Mandarine Academy itself to handle its internal training programs.

Figure 1.2 DiLeap Logistic Logo.

DiLeap Logistic provides a decision support system that allows the planner to manage a timetable. A timetable can be described as the assignment of a set of events (meetings, training, courses, etc..) to a certain time frame, where an event is a combination of resources, such as people, rooms, and equipment.

Timetabling can be classified as a type of scheduling problem due to the challenging large number of events that need to be scheduled and the extensive constraints and preferences that need to be satisfied. It has been widely investigated in the operational research and artificial intelligence research communities due to its increasing difficulty. Timetabling problems belong to the class of NP-Complete problems [87], due to the scarce resources and the tight planning horizon in which subjects must be scheduled. That means it is unlikely that there is a method that can find an optimal solution in a polynomial amount of time. However, finding a feasible solution may still be possible through various heuristics and approximation algorithms.

There are several variations on the timetabling problem in general, and it has a wide range of applications, such as:

- In the transportation sector: train or bus schedules [2], inside or outside the cities, confront different circumstances such as peak and off-peak times, construction zones, and drivers' availability. Flight timetables, for instance, are very crucial for airline companies as well as airports and passengers because of the tight schedules and the various uncertain conditions airplanes operate in [137].

- In the healthcare sector: nurses' efficiency is influenced by the quality of the timetable they get for their shifts [73]. Also, surgeons need an adequate schedule to perform their operations.

- In the educational sector: a large proportion of practical timetabling research in the literature is concerned with high school and university course and examination timetabling. Timetables are important for students in schools and universities as they can influence grades.

Within all these categories of problems, the university timetabling problem has gained increasing interest in the last four decades.

In DiLeap Logistic, each timetable provides information about the event's time, and the resources attached to each event. The tool also provides automatic alerts for trainers about their upcoming sessions. The currently implemented scheduler reduces the time to schedule in comparison with the manual approach. The construction of high-quality schedules within minutes gives the planner the opportunity to improve the overall quality and efficiency not only of the schedules but also of the whole service.

The major advantage is the possibility to provide a pending course list. The planner can validate the propositions of the scheduler; thus, courses will be scheduled. Another option is the planner can verify the conflicts of pending events, this helps to decide whether an event should be planned or not. The input data is in the form of a spreadsheet, which contains the following parameters:

- Course code: a unique course identifier.
- Number of sessions: The total number of times a course (Sessions) should be planned.
- Start date: A starting date for sessions to be planned.
- End date: A date limit for the sessions to be scheduled.
- Location: The location where courses should be scheduled.

The planner transfers the spreadsheet file using the DiLeap automatic scheduling interface and awaits the results which are shown in a timetable with the proposed sessions. The planner needs to validate the entries for a session to be scheduled.

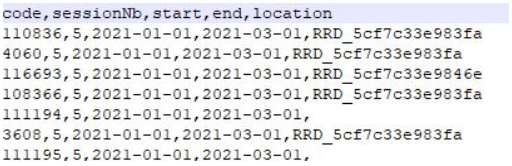

Figure 1.3 Example of the input file for planning sessions in DiLeap Logistic.

Each partner using the platform has access to their own course catalog, objectives, and constraints. Before DiLeap Logistic, timetable construction was done manually, and took an average of 3-6 weeks (30 workers) to produce an initial timetable that covers a year of training programs. To optimize this task, DiLeap Logistic uses a greedy algorithm that takes a list of events to be planned with their periods and locations to automatically create a timetable. Automating the planning process makes it easier to rent a room, alert teachers, and employees about their upcoming sessions, and manage the resources of the company. The planner can confirm the software proposals, so courses will only be scheduled following the decision maker's approval. Alerts regarding conflicts or non-compliance with constraints are also displayed to provide the planner with a sense of what occurred throughout the planning process.

Despite its advantages, the currently used automatic timetabling tool in DiLeap Logistic has several serious flaws: (1) it does not follow the owner's defined restrictions (it can mobilize resources that are unavailable or in the incorrect location), and (2) it does not handle multi-objectives (only a single objective which is maximizing the number of planned training sessions).

1.1.2 MOOC Office 365 Training

Modern online learning platforms like Coursera, Udacity, and edX are becoming more popular. Millions of users from all over the world are registering for courses online, especially after the COVID-19 pandemic.

Similar to the aforementioned MOOCs, Mandarine Academy proposes "MOOC-Office365-training" as a solution for workplaces looking for updating their employees' knowledge. The company provides up-to-date content that matches changes in work environments and current trends. Most content is available in more than 11 languages, which is made possible using Microsoft Azure's speech translation services.

1.1.2.1 Learning Materials: Types

Different types of learning materials are proposed and Table.1.1 shows the total number of learning materials in the catalog.

- Learning Paths: A predefined set of **courses** to master certain skills/job.
- Courses: A collection of unordered learning **resources**.
- Resources: **Tutorials and use cases** (short format videos), quizzes, documents, recorded web sessions, SCORM (Sharable Content Object Reference Model), serious games, and documents.

Table 1.1 Statistics about available content in Mooc Office 365 (French and English catalog).

Content Type	Number of Active items
Learning Paths	71
Courses	304
Tutorials	3116
Use Cases	283

1.1.2.2 Learning Materials: Observations

One frequent problem found in modern platforms including MOOC-Office365-Training is that it is becoming difficult to choose courses from many offered content. Online learning should grant users the freedom to control their learning environments and progresses and not be lost due to the enormous information load. Many academic researchers have been working to solve this problem with the help of recommender systems.

Recommender Systems play the role of consultants, as they help users better find interesting items (items being movies to watch, text to read, and

products to buy) without spending extra time finding them. These data-driven systems were the product of major web services such as YouTube, Amazon, and Netflix. Recommender systems are critical in some industries as they can generate a huge amount of income when they are efficient or also be a way to stand out significantly from competitors. As proof of the importance of recommender systems, we can mention that, a few years ago, Netflix organized a challenge (the "Netflix prize") where the goal was to produce a recommender system that performs better than its own algorithm with a prize of 1 million dollars to win. According to McKinsey's conversion research [97], 35% percent of what consumers purchase on Amazon and 75% of what they watch on Netflix come from product recommendations.

Resources (Tutorials and Use-cases) make up most of the catalog's contents. With the latter in an ever-increasing state, it has an impact on users, as they must spend more time selecting the appropriate learning material, among other issues:

- New subscribers/visitors: Newcomers may have difficulty selecting the appropriate material, to begin with, depending on their needs (learn a new skill or build upon existing knowledge).
- Watch next: After finishing an item, users are not given a playlist of what to watch next. This can cause frustration and an increase in dropout rates.
- Lack of personalization: The provided content is intended for all users and is not tailored to a specific individual's needs.

Initial solutions: Mandarine Academy has created an online community (Yammer) [43] for MOOC users where they can share their thoughts and seek help from professionals if they are having problems. This was done to direct users to appropriate resources, share their platform experience, and engage with trainers. Unfortunately, because most users avoid such actions, this did not work very well. The company also used Microsoft Azure Recommender Systems in a second approach. This black-box approach to content delivery did not provide the company with the desired diversity or flexibility. The model only uses page views and is limited to one type of content (courses). The company later abandoned this solution because it provided no significant improvement to course subscriptions and required users to complete multiple courses before receiving personalized recommendations. This was impractical because users rarely follow multiple courses at once.

Exploratory Data Analysis: Content & Interactions The more you know about your users, the better equipped you'll be to make informed de-

cisions about your service. In order to gain a richer understanding of how users interact with content, events are used to independently track users' journeys. A typical method of providing feedback is in the form of rating methods that captures users' preferences in explicit ways (like button, social sharing, course/learning path registration, and bookmarks). The disadvantage is that users tend to avoid the burden of explicitly stating their preferences. To overcome the shortage of explicit ratings, platforms tend to collect users' behavior through multiple ways (page views, percentage of videos watched, etc). This is called implicit feedback. The advantage is that users can trigger a lot of actions when using a service. This generates a lot of data that can be significant in some cases but shows a major inconvenience, which is not having a ground truth. When running short on ratings (explicit or implicit), content descriptors like (subtitles, title, description, number of views, duration, etc.) are used as additional input to recommender systems.

The company provided us with limited access to examine usage data of their public MOOCs. To conduct our analysis, the public MOOC: MoocOffice365[1] (French version) is selected. The MOOC has more than 130K registered users and around 3.5K average monthly users. Most visitors are using the French version (93%) while a small minority (7%) use the English version. The data used was collected from early 2018 to late 2020 and has the following content:

- 41 Learning Paths.
- 142 Courses.
- 1294 Tutorials and 113 Use Cases.

Mandarine Academy as well as other modern online services provide users with multiple ways of expressing their feedback about their online experience.

Collected data was captured from early 2018 to late 2020. Both Table 1.2 and Table 1.3 show available user events (explicit and implicit). Starting with column **% of users** which indicates the percentage of users that used the feature at least once. The difference between explicit and implicit ratings is distinguishable. Only about 1% of users have explicitly indicated their feedback, with social sharing being the most used. When looking at implicit interactions, we see a different story, as more significant users are interacting with content.

The same behavior applies with column **% of content** as explicit interactions have a smaller number of involved content compared to implicit

[1]https://mooc.office365-training.com/

ones. Furthermore, investigating implicit ratings reveals that approximately 7% of pages have never been visited and approximately 9% of videos have never been watched. These findings are alarming, especially when a part of the catalog is hidden from public view.

Finally, column **sparsity score** defines the ratio of unspecified ratings to the total number of entries in the user-item matrix and is calculated as follows $sparsity = 1 - \frac{|R|}{|U|x|I|}$.

Table 1.2 Explicit interactions captured from Mooc-Office365 (French) starting 2018 to late 2020.

Interaction	% of users	% of content	Observations	Sparsity
Likes/Dislikes	0.02%	0.9%	28	0.830%
Social Shares	0.66%	58.11%	2179	0.997%
Content Subscription	0.439%	40%	1202	0.996%
Bookmarks	-	-	-	-

Table 1.3 Implicit interactions captured from Mooc-Office365 (French) starting 2018 to late 2020.

Interaction	% of users	% of content	Observations	Sparsity
Page View	21.86%	93.08%	610,956	0.985%
Video View Time	8.26%	91.57%	68,894	0.993%

1.1.2.3 Suggestions and Objectives

Observations were taken from Table 1.2 and 1.3 not only show a high sparsity score which is normal for real-world data but also a usage gap between both implicit and explicit interactions. Since explicit ratings are visible to users, we suppose that other reasons besides avoiding expressing their opinion might be possible. To confirm our hypothesis we investigate the graphical interface available for both registered users and visitors. We list below our findings per page.

1. Home page: The current homepage offers a list of the newest courses and tutorials. Users/visitors are limited if they are looking to learn about certain tools and required skills for specific jobs or certifications. What we propose for **visitors** is a list of items (courses and resources) with options to select popular or newer items. Furthermore, categories (skills, jobs, certificates) should be shown at top of the page to guide visitors efficiently. For **registered users**, multiple

personalized lists of recommended items provided by combining our approach with other common algorithms will help users find relevant content easier.

2. Content page: Learning paths, courses, and videos (tutorials and use cases) are presented to both users and visitors without similar items, visible interactions, or feedback options. The like and share buttons are provided without text, only a small icon. In case a video doesn't correspond to a user's needs, they must go back to the previous page and spend additional time looking for another one. We propose for both users and visitors a more appealing interface with visible interactions (like, dislike, social share), the addition of a "save to watch later (Bookmark)" and "feedback" options. Different recommended items (based on similarity or popularity) to minimize the burden of content search and provide guidance.

Overall, Mandarine Academy is working to improve the user experience and satisfaction on their e-learning platforms, as well as to provide relevant content and maintain quality services to their clients. Despite the significant changes to the platform, the company is eager to gradually improve the learning experience, beginning with the use of recommender systems and progressing to the graphical changes suggested in our research.

1.2 Motivations

This thesis handles two different real-world problems found at Mandarine Academy.

1.2.0.1 Timetabling Problem

The first problem is well known in the literature under the name of 'timetabling problems' which has been proven to be an NP-Complete problem [87], This means that obtaining an optimal solution in polynomial time is challenging. However, it is worth noting that exact solvers have made significant progress in recent years, allowing for more efficient solutions to be found in some cases. What makes this task even harder is the fact that this is different from what is commonly found in the literature as most works fall under the educational category and have fewer real-world constraints and business objectives. A detailed comparison is provided in Chapter 2 between our problem and what is commonly found in the literature. The second problem is not a scheduling one but is much more complicated.

1.2.0.2 Recommendation Problem

In the second part, we handle e-learning recommendations for online learners using the platform MOOC-Office365. After performing data analysis, we found that many users are leaving the platform after viewing an average of 4 videos. This was alarming for the company especially since the MOOC is public and offers both free and paid options. What makes things harder is the fact that there aren't enough data to understand users' opinions. Not only that, but front-end analysis shed light on many visual problems that may have contributed over the years to the dropout rates. To solve this problem, we need to provide not only a mathematical model for objectives and user profiles, but also propose graphical changes that will make our work visible and explainable to end-users.

1.3 Thesis Outline

This thesis is organized according to the following plan. Chapter 1 introduces Mandarine academy company and its different products. We provide a critical analysis of DiLeap-Logistic and MOOC-Office365, two public online services provided by the company. We highlight the different problems concerning the two products and then we explain the motivation behind such work.

Chapter 2 provides background on mathematical optimization techniques such as single and multi-objective metaheuristics. We also look at the various related subjects such as parameter tuning and evaluation techniques specific to metaheuristics. In the second part, we study the timetabling problem found in the literature by providing a definition for each problem type and criticizing works that resemble what we will be doing in this work. In the third part, a state of art concerning recommendation systems and their different types is discussed using literature works.

Chapter 3 addresses the first part of our thesis, which focuses on solving a real-world timetabling problem using the DiLeap-Logistic tool. We begin this chapter by introducing the various entities involved in the scheduling process and their mathematical models. A multi-objective genetic algorithm (MOGA) approach is applied, and we define different genetic structures (encoding, crossover, and mutation) that will be used to solve the problem. A proposed experimental protocol is discussed in detail before presenting the obtained results.

Chapter 4 in the second part of our thesis we handle recommendation systems inside Mandarine Academy's public MOOC. Same as in the pre-

vious chapter, an introduction to entities and objectives is provided with mathematical modeling. Later, a genetic representation with operators is discussed along-with experimental protocol. We conclude this chapter by discussing production mode integration and results.

Chapter 5 summarizes all the above chapters and identifies future improvements for both solutions (DiLeap-Logistic and Mooc-Office-365).

Chapter 2

State of the Art

Contents

This chapter begins with an overview of combinatorial problems and multi-objective optimization. Following that is a general introduction to the timetabling issues that are the focus of the first part of this work. Following that, because it is directly related to the second part of this work, an introduction to recommendation systems is discussed. Finally, a discussion of related studies at the end of both the timetabling and recommendation systems sections is provided.

2.1 Combinatorial Optimization

Combinatorial optimization is the process of finding the best solution from a limited set of possible solutions [26]. Each solution is evaluated during the search process, and the solution with the best evaluation is returned at the end. Many real-world issues can be expressed as combinatorial optimization problems. For example, finding the best route for package delivery or the best schedule to attend all college classes, and so on. Due to the vast number of possible combinations that can result in a valid solution, combinatorial problems are notoriously difficult to solve.

Combinatorial optimization problems can be thought of as looking for the best element of a set of discrete items; thus, in theory, any search algorithm or metaheuristic can be used to solve them.

2.1.1 Background on Combinatorial Optimization

Metaheuristics can be seen as a general algorithmic system that can be applied with very few modifications to different optimization problems to make them adapt to a particular problem. Today there are several types, and their creation comes from different sources of inspiration [123].

Some are made by analogy to other scientific areas such as physics (simulated annealing), biology (ant colony and evolutionary algorithms), neurology (tabu search) and sociology (memetic algorithms, particle swarm optimization, and multi-agent systems) [116].

Unlike exact methods, meta-heuristics deliver suitable solutions within a reasonable period of time to tackle large-scale problem instances. There is no guarantee that global optimal solutions can be found.

In designing a metaheuristic, two contradictory techniques must be taken into account: the exploration of the search space (diversification) and the exploitation of the best solutions found (intensification). In intensification, the promising regions are explored more thoroughly in the hope to find better solutions.

In diversification, non-explored regions must be visited to be sure that all regions of the search space are evenly explored and that the search is not confined to only a reduced number of regions.

2.1.1.1 Exact and Approximate Algorithms

Computing optimal solutions are intractable for many optimization issues of industrial and scientific importance. Meta-heuristics represent a family of approximate optimization strategies that gained a lot of popularity within the past decades. They offer "good" solutions to complicated and complex problems in a reasonable time. This explains the significant in-

crease of interest in the meta-heuristic domain within the scientific literature.

Unlike exact optimization algorithms, metaheuristics do not guarantee that the solutions obtained are optimal [62]. They do not describe how close the solutions obtained are to the optimal solution. Exact approaches can prove a solution's optimality, whereas an approximate approach can't. However, exact algorithms will typically constitute a brute-force style approach. Nonetheless, for scheduling, given a huge dataset where we have to schedule hundreds or thousands of events in a small number of rooms or time span, exact algorithms would be poorly implemented and cannot provide a solution within an acceptable time frame. Below Figure.2.1 shows different optimization methods.

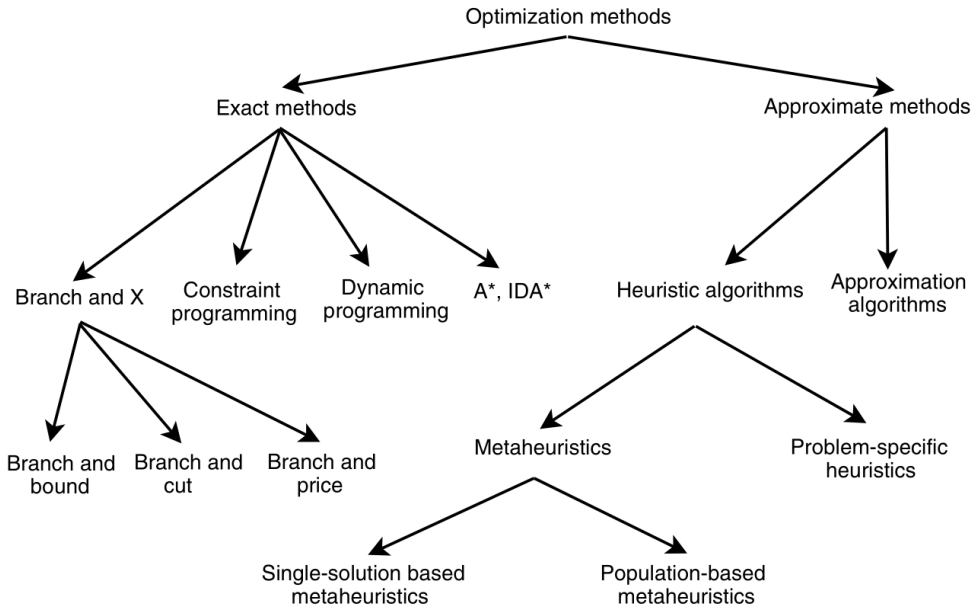

Figure 2.1 A general overview of optimization methods [123].

Exact Methods: In the class of exact methods, one can find generally the following algorithms: Dynamic programming, Branch and X family of algorithms (branch and bound, branch and cut, branch and price), constraint programming, and A* search algorithms (A*, IDA* —Iterative Deepening Algorithms) [125]. These methods can be used as algorithms for tree search. The search is performed in the entire search space and the problem is solved by splitting it into simpler problems (sub-problems).

Dynamic programming is based on dividing a problem recursively into simpler sub-problems [9]. This procedure is based on the theory of Bellman that says, "the sub-policy of an optimal policy is itself optimal". While dynamic programming has been applied to various problems such as knapsack

problems, planning, and routing, its success, and efficiency depend on the complexity of the algorithm (in terms of time and space). In some cases, dynamic programming can be memory-intensive and less efficient. However, for specific problems like the shortest path problem, dynamic programming can be an effective and efficient solution. As always, the efficiency of dynamic programming depends on the complexity of the algorithm in terms of time and space requirements.

Branch & bound and A* methods are based on an implicit listing of all approaches to the optimization problem considered. The search space is explored by dynamically constructing a tree whose root node represents the problem to be solved and the entire search space associated with it. The leaf nodes are the possible solutions, and the internal nodes are the subproblems. The pruning of the search tree is based on a bounding function that prunes sub-trees that do not contain any optimal solution [99].

Linear Programming is a powerful and general mathematical framework for solving optimization problems that involve linear constraints and objective functions. It encompasses various problem-solving methods, such as shortest path, network flow, MST, matching, assignment, etc. Most commercial software like CPLEX and GUROBI implements Linear Programming methods. In linear programming, constraints and objective functions are linear and the simplex method is commonly used in practice thanks to its reliability [9].

We deal with mixed-integer programming (MIP) problems when the decision variables are both discrete and continuous. Hence, MIP models generalize LP and IP models. Advanced optimization methods such as relaxation and decomposition approaches and cutting plane algorithms have significantly improved lately in solving MIP problems.

Approximate Methods: Two main sub-classes of algorithms can be differentiated within the class of approximate methods: Approximation and Heuristic Algorithms. Heuristic algorithms can be further divided into sub-categories, such as Greedy Algorithms

Approximation algorithms provide valuable insights into a problem's complexity and can help develop powerful heuristics. However, their applicability is limited due to their problem-dependent nature. Furthermore, the quality of the solutions they produce can be significantly different from the ideal global optimal solution in practice [72].

Typically, greedy algorithms start from scratch (empty solution) and pick the best option at the moment, following a pre-defined rule, even if doing so means missing better options for later on in the solution. Once a decision is made, it is not reconsidered. This process is repeated until a complete solution is created [131].

Greedy algorithms are very common techniques because they are sim-

ple to design. Additionally, greedy algorithms have reduced complexity compared to iterative algorithms in general. Despite their simple design, they are useful when integrated into a heuristic approach. These are widely used as part of elaborate approaches, especially in the preparation of local search solutions (which can provide an initial solution).

Heuristics, approximation algorithms, and greedy algorithms are all related concepts, but they have some differences. Heuristics are problem-solving techniques that find relatively "good" solutions in a reasonable time, but they usually don't have a guarantee of approximation on the solutions obtained. They can be divided into two families: specific heuristics, which are designed to solve a particular problem, and meta-heuristics, which are general-purpose algorithms that can be used to solve almost any optimization problem [123].

Approximation algorithms, on the other hand, provide provable solution quality and provable run-time limits. They are a type of heuristic that guarantees a certain level of approximation to the optimal solution.

Greedy algorithms are a specific type of heuristic that makes the best choice at each step, following a pre-defined rule, without reconsidering previous decisions. While they are simple and efficient, they may not always find the optimal solution.

In summary, greedy algorithms are a type of heuristic that makes decisions based on the best option at the moment, following a pre-defined rule, without reconsidering previous decisions. Heuristics are problem-solving techniques that find relatively good solutions in a reasonable time, and approximation algorithms are a type of heuristic that guarantees a certain level of approximation to the optimal solution.

2.1.1.2 Single Solution-Based Metaheuristics

Before diving into Single Solution-Based Metaheuristics (S-Metaheuristics) methods, we need to further grasp some concepts. We start with the neighborhood and then the initial solution. The definition of the neighborhood is a required common step for the design of any meta-heuristic. The neighborhood structure plays a crucial role in the execution of a single solution-based meta-heuristic. If the neighborhood structure is not adequate for the problem, any meta-heuristic will fail to solve the problem. A solution s' in the neighborhood of s is called a neighbor of s, $s' \in N(S)$. A neighbor is generated by the application of a "move" operator m that performs a small perturbation to the solution s. The Fitness Function (also known as the Evaluation Function) determines how close a given solution is to the ideal solution to the desired problem. Generally, high-fitness solutions are selected as the best solution in each iteration of the optimization problem.

In addition to the neighborhood and initial solution, it is essential to

understand the concept of "coding." Coding refers to the representation of a solution in a specific format that can be easily manipulated by the metaheuristic algorithm. This representation is crucial for the algorithm to efficiently search the solution space and find the optimal solution. Different problems may require different coding schemes, and selecting an appropriate coding method is vital for the success of the metaheuristic algorithm.

The initial solution is developed through two main strategies: a random approach and a greedy one. In terms of the consistency of solutions and the computational time, there is often a trade-off between the use of random and greedy initial solutions [120]. The best approach to this trade-off will rely primarily on the efficiency and effectiveness of the available random or greedy algorithm and the meta-heuristic properties. For example, the larger the neighborhood, the less sensitive the initial solution is to metaheuristic performance.

Generating a random initial solution is a simple operation but it may take a much greater number of iterations to converge with the metaheuristic. A Greedy heuristic can be used to speed up the search. Indeed, greedy algorithms have reduced polynomial-time complexity in most cases. Using greedy heuristics also contributes to local optima being of better quality. In general, the meta-heuristic should require fewer iterations to converge towards a local optimum. This does not mean, however, that the use of better solutions as initial solutions would always lead to better local optima.

After understanding both concepts, we list the different algorithms found in S-Metaheuristics. Starting with Local Search (LS).

Local Search (LS) is probably the simplest method in meta-heuristics. It finds good solutions by iteratively substituting a neighbor for the current solution that improves the objective function [1]. The current solution is replaced in each iteration by a solution from its neighborhood. The rule used to select the new current solution is called a move or search strategy. The search stops when all candidate neighbors are worse than the current solution, meaning a local optimum is reached.

There are two common search strategies: "best improvement" and "first improvement." In the "best improvement" strategy, also known as the steepest descent or steepest ascent strategy, the best move from the neighborhood is chosen. Meta-heuristics using this strategy are often referred to as hill climbers. It selects the move that improves the current solution by the smallest amount [59].

On the other hand, the "first improvement" strategy selects the first move encountered in the neighborhood that improves the current solution. This strategy can be more efficient in terms of computation time, as it does not require evaluating all possible moves in the neighborhood. However, it

may not always find the best possible improvement in each iteration.

Simulated Annealing (SA) was first described in [83]. SA is inspired by the metallurgic annealing process. In this natural process, a material is heated and slowly cooled under controlled conditions to increase the size of the crystals in the material and reduce their defects. The heat increases the energy of the atoms allowing them to move freely, and the slow cooling schedule allows the discovery and exploitation of a new low-energy configuration. Each configuration of a solution in the search space represents the different internal energy of the system. Heating the system leads to a relaxation of the acceptance criteria of the samples taken from the search space. As the system cools down, sample acceptance criteria are narrowed to focus on improving movements. Once the system has cooled, the setup will represent a sample at or near a global optimum.

Tabu Search (TS) algorithm became very popular in solving optimization problems in an approximate manner. Today it is one of the most common meta-heuristics. The use of memory, which stores search-related information, represents the characteristic of tabu search [48]. The whole neighborhood is usually explored deterministically, whereas a random neighbor is selected in Simulated Annealing (SA). As in local search, when a better neighbor is found, it replaces the current solution. When a local optimum is reached, the search continues by choosing a candidate which is worse than the current solution. The best solution in the neighborhood is selected as the new current solution even if it is not improving the current solution. Tabu search may generate cycles where previously visited solutions could be selected again. To avoid cycles, TS discards the neighbors that have been previously visited. It memorizes the recent search trajectory. Tabu search manages a memory of the newly applied solutions or moves called the tabu list. The approach strategy is to keep the specific changes of recent moves within the search space in short-term memory and to prevent future moves from undoing those changes.

Iterated Local Search (ILS) is widely used because of its simplicity and efficiency. It starts the search with an initial solution. Iterated Local Search explores a sequence of solutions that are created as perturbations of the current best solution, resulting in an embedded heuristic [95].

1. First, a local search is applied to an initial solution.
2. Then, in each iteration, a perturbation of the obtained local optima is carried out. Finally, a local search is applied to the perturbed solution. The generated solution is accepted as the new current solution under some conditions.

3. This process iterates until a given stopping criterion.

There exist several variants of ILS in the literature. Some accept worse solutions with certain conditions. Others consider more than one type of perturbation and apply them at specific steps of the algorithm.

Variable Neighborhood Search (VNS) The strategy for the Variable Neighborhood Search involves iterative exploration (Randomly or systematically) of larger and larger neighborhoods for a given local optimum until an improvement is located [58]. The strategy is motivated by three principles:

1. A local minimum for one neighborhood structure may not be a local minimum for a different neighborhood structure.

2. A global minimum is a local minimum for all possible neighborhood structures.

3. Local minima are relatively close to global minima for many problem classes.

Numerous VNS variants can be found in the literature, namely reduced VNS and skewed VNS.

Guided Local Search (GLS) is a deterministic meta-heuristic, applied primarily to problems of combinatorial optimization [129]. The strategy for the Guided Local Search algorithm is to use penalties to encourage a Local Search technique to escape the local optima and discover the global optima. A Local Search algorithm is run until it gets stuck in a local optimum. The features from the local optima are evaluated and penalized, the results of which are used in an augmented cost function employed by the Local Search procedure. The Local Search is repeated several times using the last local optima discovered and the augmented cost function that guides exploration away from the discovered local optima.

2.1.1.3 Population-Based Metaheuristics

Like the Single Solution-Based Metaheuristic, some concepts need to be cleared before exploring the P-Metaheuristic methods. This includes both the initial population and the stopping criteria.

Population-based meta-heuristics are naturally more exploration-oriented search algorithms due to the large diversity of initial populations, whereas single solution meta-heuristics are more exploitation-oriented search algorithms. The determination of the initial population is often disregarded in the design of a Population-metaheuristic. Nonetheless, this step plays a

crucial role in the effectiveness of the algorithm and its efficiency. The main criterion to address in the generation of the initial population is diversification. If the initial population is not well diversified there may be premature convergence for any meta-heuristic population based [37]. For instance, this may happen if the initial population is generated using a greedy heuristic or a single solution-based meta-heuristic (e.g., local search, tabu search) for each solution of the population. Many stopping criteria based on the evolution of a population may be used. Some of them are similar to those designed for single solution-based meta-heuristics [40].

- Static procedure: In a static procedure, the end of the search may be known a priori. For instance, one can use a fixed number of iterations (generations), a limit on CPU resources, or a maximum number of objective function evaluations.

- Adaptive procedure: In an adaptive procedure, the end of the search cannot be known a priori. One can use a fixed number of iterations (generations) without improvement when an optimum or satisfactory solution is reached.

Some stopping criteria are specific to population-based meta-heuristics. They are generally based on some statistics on the current population or the evolution of the population. Mostly, they're related to population diversity. This stopping criterion deals with the stagnation of the population. It's useless to keep the execution of a population-based meta-heuristic when the population stagnates. Many stopping criteria based on the evolution of a population may be used. Some of them are similar to those designed for single solution-based meta-heuristics [40]. Population-based meta-heuristics share many common concepts. They could be viewed as an iterative improvement in a population of solutions. A key difference from other approaches is that there is some mechanism within the population to exchange information between different candidate solutions [32]. First, the population is initialized. Then, a new population of solutions is generated. Finally, some selection procedures are applied to integrate this new population into the current one. When a given condition is met (stop criterion), the search process is stopped. Algorithms such as Evolutionary Algorithms (EAs), Scatter Search (SS), Estimation of Distribution Algorithms (EDAs), Particle Swarm Optimization (PSO), Artificial Bee Colony (ABC), and many others belong to this class of meta-heuristics.

Evolutionary Algorithms (EAs) Evolutionary algorithms are based upon the simplified evolutionary biological model and natural selection [29]. They are based on the simplified biological model of evolution and natural selection. EAs are population-based algorithms that process a whole set

of candidate solutions simultaneously. While solving a particular problem, the parameters of the problem create an environment in which potential solutions can evolve, this promotes the evolution of good solutions. EAs act on a population of possible solutions and include three steps: (1) selection, (2) regeneration, and (3) replacement. In the selection phase: high-fitness solutions are selected to be next-generation parents. In the regeneration phase: two crossover and mutation operators are performed on parents who have been selected in the selection and replacement phases: and the solutions of the initial population are replaced by the newly created solutions.

2.1.1.4 Genetic Algorithms (GAs)

Genetic algorithms have been developed by J. Holland in the 1970s [63] to understand the adaptive processes of natural systems. Then in the 1980s, they were applied to machine learning and optimization. GAs are a hugely popular EAs class. GAs is traditionally associated with using a binary representation but nowadays one can find GAs using other types of representations. GA considers a solution to be a chromosome structure that contains good and bad phenotypes. A GA usually applies a crossover operator to two solutions that play a major role to produce new chromosomes called children chromosomes. The children are then mutated. The mutated children's fitness (quality of the solution) is then calculated, and they are inserted into the population. While the mutation is bit flipping, the crossover operator is based on the n-point or uniform crossover. The population considered after crossing depends on the algorithm's variant. The first variant is the generational genetic algorithm (GGA) where the population is constituted only by the new chromosomes. The second variant is the steady-state genetic algorithm (SSGA) where the children are not inserted directly but are in competition with existing chromosomes. Figure.2.2 presents a general flowchart of Genetic Algorithms.

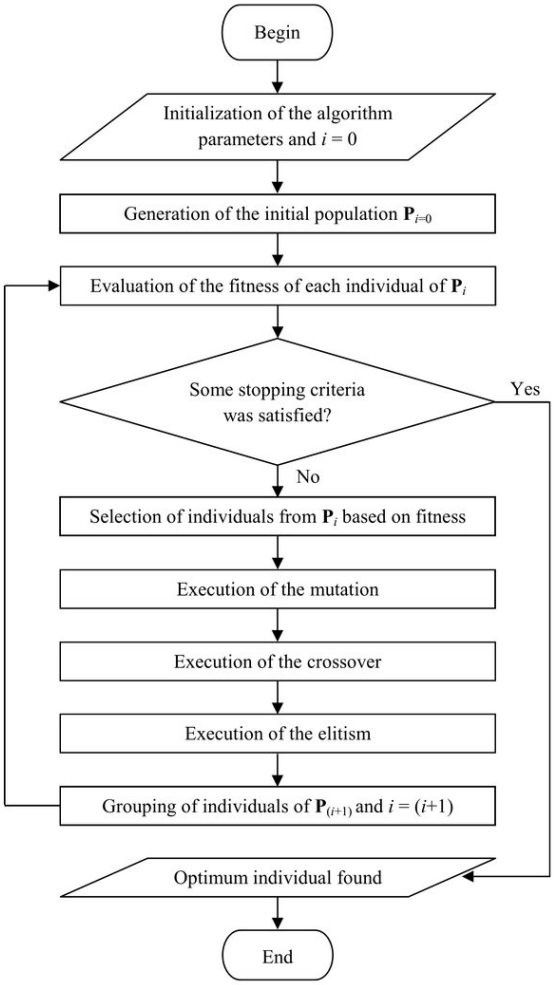

Figure 2.2 Basic scheme of a genetic algorithm [78].

Genetic Operators (Mutation & Crossover): Genetic operators are considered a mechanism to generate evolution from one generation to another generation. They are crucial for the genetic algorithm process. Note that these operators are applications specific and different behavior is expected for each problem.

Once the initial generation (population) is created evolution takes place by using these operators:

- Selection Operator (SX): The idea is to give preference to individuals with good fitness scores and allow them to pass their genes to successive generations. Generally, the roulette wheel and tournament selection are used as selection operators. Examples of both operations can be found in Figure.2.3 and Figure.2.4 respectively.

- Crossover Operator (CX): Two individuals are selected using the selection operator and crossover points are chosen randomly to exchange their genes in order to produce a new chromosome called "children chromosome - offspring". Examples of both 1 and 2-point crossover operators can be found in Figure.2.6.

- Mutation Operator (MX): A mutation is a change to a chromosome that introduces variety into a population, allowing it to escape local optima (premature convergence). An example of a 'Swap' mutation is illustrated in Figure.2.5.

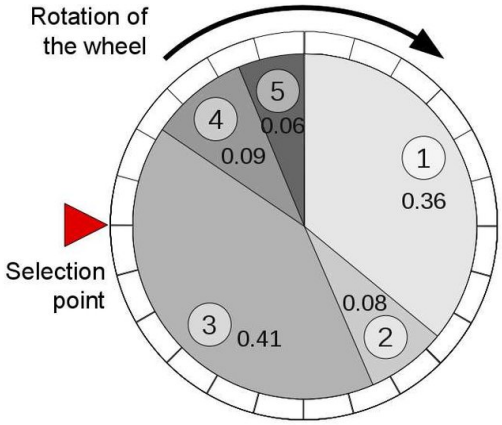

Figure 2.3 Example of roulette wheel selection [122].

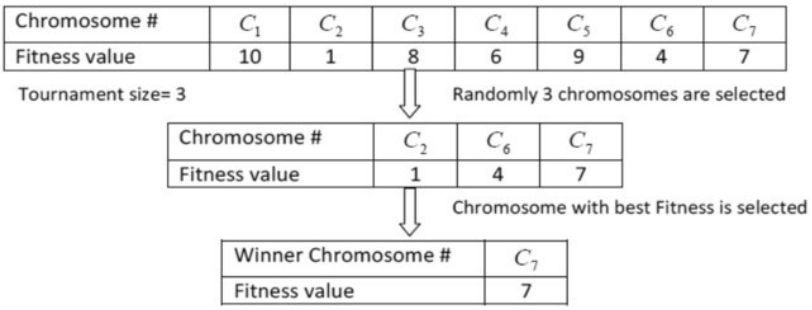

Figure 2.4 Example of tournament selection [7].

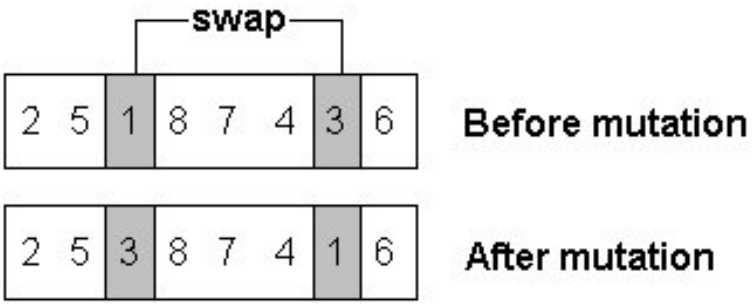

Figure 2.5 Example 'swap' mutation [39].

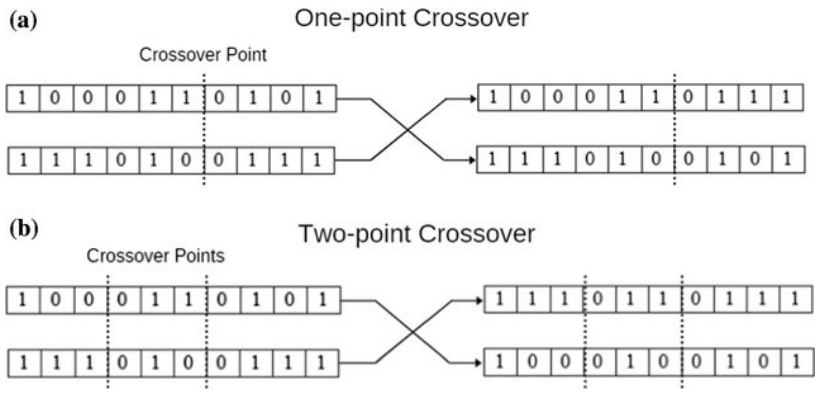

Figure 2.6 Example of 1 and 2-point crossover [75].

Swarm Intelligence (SI) Swarm intelligence is the study of collective intelligence-inspired computational systems [15]. Collective Intelligence emerges through large numbers of homogeneous agents working together in the environment. Examples of this include bird flocks and ant colonies. Such intelligence is decentralized, self-organized, and spread across an environment. Such systems are commonly used in nature to solve problems such as effective food foraging, prey evading, or relocation to the colony. Typically, the information is stored throughout the participating homogeneous agents or is stored or communicated in the environment itself, such as by using pheromones in ants, dancing in bees, and closeness for fish and birds. The paradigm consists of two dominant sub-fields (1) Ant Colony Optimization investigating probabilistic algorithms inspired by ant staggering and foraging behavior, and (2) Particle Swarm Optimization investigating probabilistic algorithms inspired by flocking and herding. Like evolutionary computation, swarm intelligence 'algorithms' or 'strategies' are considered adaptive strategies and are typically applied to search and optimization domains.

Particle Swarm Optimization (PSO) was developed for continuous-variable optimization. The main idea is that particles in the swarm fly through an environment following the swarm's fitter members and generally biasing their movement towards historically good surrounding areas [38]. A particle represents an individual of a group in a swarm and the solutions are the positions (or the search places). Initially, random positions are assigned to all particles in the space with small initial random velocities. Each particle memorizes its current position in the search space and the best-visited position. The best single position represents the particle's individual experience. The group's experience represents the global best position the population has found. Every particle has a movement speed that represents the degree of change to the objective function that can occur in the next iteration. At each iteration of the PSO algorithm, the movement speed and the current position of an individual are updated. The movement speed update is managed using three orientations:

1. The current speed of the particle multiplied by the inertia factor.
2. The tendency of returning to the previous individual experiences multiplied by a cognitive factor.
3. The tendency of a group's experiences multiplied by a social factor.

Ant Colony Optimization (ACO) The Ant Colony Optimization (ACO) algorithm is inspired by ants' foraging behavior, specifically the communication of pheromones between ants regarding a good path between the colony and a food source in an environment. It was first proposed by [25] and is used for solving combinatorial optimization problems. ACO is used to solve problems such as the problem of the traveling salesman, vehicle routing and scheduling, and many others. Ants initially wander around their surroundings at random. Once the food is located an ant will start to lay pheromone in the environment. Numerous trips are undertaken between the food and the colony, and if the same route is followed that leads to food, then an additional pheromone is set. In the environment, pheromone decays, so older paths are less likely to be followed. The ants who choose the shortest route randomly will be the fastest to return to the nest and therefore this route receives pheromones earlier than other routes and then it is more likely that ants will choose this route over others. The pheromone trail strengthens as more ants follow the shorter path until no ants follow the longer route. The strategy's aim is to exploit historical and heuristic information to build candidate solutions and fold the information learned from building solutions into history. History is updated in proportion to the quality of the best-known solution and decreased in proportion to the usage of discrete solution components.

During the search for optimal solutions, single solution-based algorithms (e.g., local search, simulated annealing) manipulate and transform a single solution, while a whole population of solutions is evolved in population-based algorithms (e.g., particle swarm, evolutionary algorithms). These two families have complementary characteristics [123]: single solution-based meta-heuristics are exploitation oriented; they have the power to intensify the search in local regions. Population-based meta-heuristics are exploration oriented; they allow a better diversification in the whole search space.

2.1.2 Multi-Objective Combinatorial Optimization

Many industrial domains are concerned with large and complex optimization problems involving many criteria. Indeed, optimization problems encountered in practice are rarely mono-objective. In general, there are many conflicting objectives to handle. For instance, in designing a given product, one must have to minimize its cost, maximize its quality (e.g., in terms of physic, mechanic, or service), and minimize its environmental impact. In fact, many diverse areas (e.g., engineering design, bio-informatics, logistics, transportation, telecommunication, environment, aeronautics, and finance) are concerned with multi-objective optimization problems (MOPs). In this section, we give a brief description of Multi-Objective Combinatorial Optimization Problems (MOCOPs).

2.1.2.1 Definition

Multi-Objective Combinatorial Optimization can be defined as optimizing $F(x) = (f_1(x), f_2(x), ..., f_n(x))$ where $x \in F_{sol}$, n is number of objectives ($n \geq 2$), x being a vector of decision variables, F_{sol} is a set of feasible solutions and $f_i(x)$ depicts objectives that we want to minimize/maximize [24].

Unlike single-objective optimization, results are not a single solution but a Pareto set [105] of optimal solutions where no improvement can be made for an objective without sacrificing another objective. In mathematical terms, Pareto dominance can be defined by:

A feasible solution $x_1 \in X$ dominates $x_2 \in X$ if $\forall i \in 1, ..., k, f_i(x_1) \leqslant f_i(x_2)$, *and* $\exists i \in 1, ..., k, f_i(x_1) < f_i(x_2)$ Therefore, a solution $x^* \in X$ is called Pareto Optimal if there does not exist another solution that dominates it. The set of Pareto optimal solutions denoted x^* is called the Pareto Front. For a multi-objective optimization problem, the front is bounded by a nadir (worst solutions) and Ideal/Utopian solutions. This is illustrated in Figure.2.7.

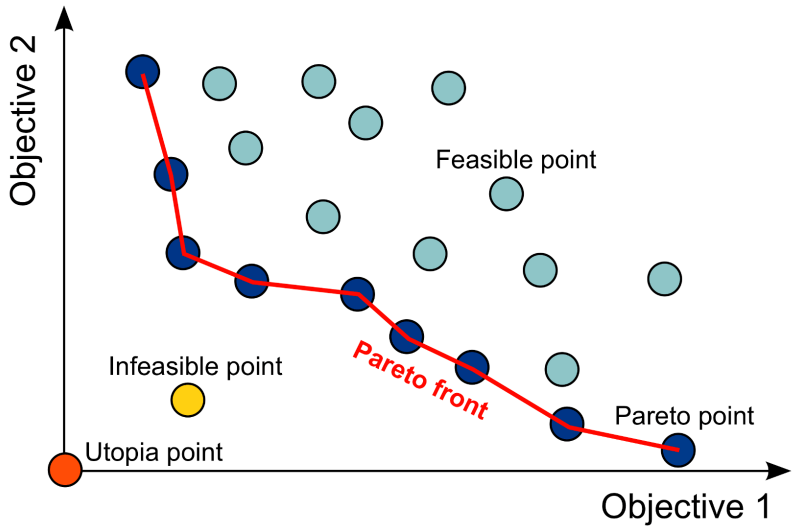

Figure 2.7 Example of a Pareto front in a minimization problem [84].

2.1.2.2 Evaluating Multi-Objective Methods

Analyzing the performance of metaheuristic methods is a crucial task that must be carried out fairly. This includes conducting an experimental design process to determine which instances, parameters, and objectives to work with first. The major difficulty in assessing multi-objective optimization methods is that the output of the optimization process is not a single solution but a set of solutions representing an approximation of the Pareto front. A comparison between these sets of solutions is mandatory to identify the best-performing approach. Before diving into metrics, there are two concepts we need to consider. The first is convergence, which measures the closeness of the solutions to the optimal Pareto front. The second is diversity, which measures the spread of solutions across the set.

The first metric is the Generational Distance (GD) [142]. It works as follows, the average overall solutions $a \in A$ of the distance between solution a and the closest solution in a reference set R:

$$GD(A, R) = \frac{1}{|A|} \sum_{a \in A}^{r \in R} min\ dist(a, r) \tag{2.1}$$

Where $dist(a, r)$ is the Euclidean distance in objective space between solution a and r. There exist another variant of the GD called Inverted Generational Distance (IGD), which can be expressed as $IGD(A, R) = GD(R, A)$. Both are considered easy criteria to meet since having even one close solution to the set yield an excellent score. Figure.2.8 shows an example of Generation Distance calculation.

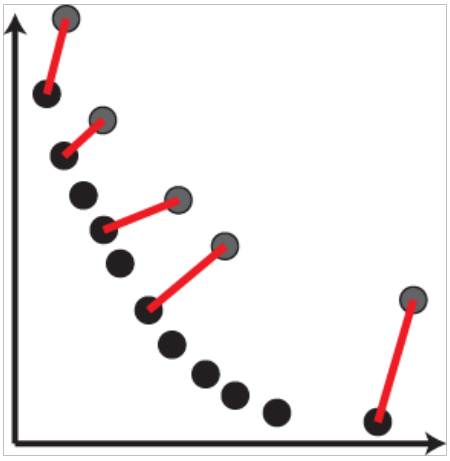

Figure 2.8 Example of Generational Distance, Red lines show the distance between approximation and the best approximation (Minimization) [80].

The second metric is the Epsilon indicator (ϵ) [142] which uses the worst-case distance to the optimal Pareto front. This means if we have multiple solutions close to the optimal Pareto front and one solution far from it, ϵ will only consider the worst solution. The metric is harder to meet, unlike GD. Figure.2.9 illustrates an example of the metric. The epsilon metric of a set A with respect to a reference set R is defined as:

$$epsilon(A, B) = \max_{r \in R} \min_{a \in A} \min_{1 \leq i \leq n} epsilon(a_i, r_i) \tag{2.2}$$

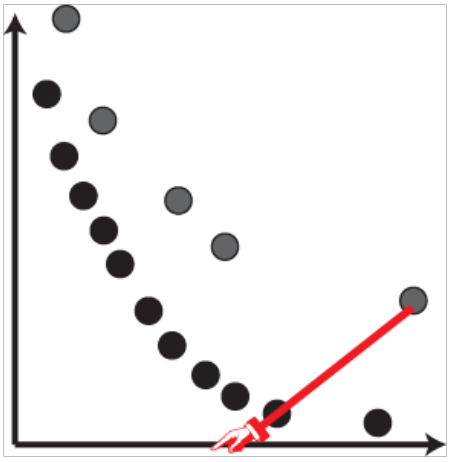

Figure 2.9 Example of the Epsilon Indicator with the worst case distance (Minimization) [80].

Finally, the hypervolume metric (HV) [142] captures both convergence and diversity. It looks at the multidimensional "volume" created by each

set, relative to a reference point. However, computing the hypervolume is expensive.

$$HV(f^{ref}, X) = \Lambda \left(\bigcup_{X_n \in X} \left[f_1(X_n), f_1^{ref}) \right] \times \cdots \times \left[f_m(X_n), f_m^{ref}) \right] \right) \quad (2.3)$$

Where $HV(f^{ref}, X)$ resolves the size of the space covered by an approximation set X, $f^{ref} \in \mathbb{R}$ refers to a chosen reference point, and Λ refers to the Lebesgue measure. According to the example in Figure.2.10 the hypervolume metric compares a multidimensional volume determined by the approximation (red) to the volume determined by the best-known approximation (black), relative to a reference point.

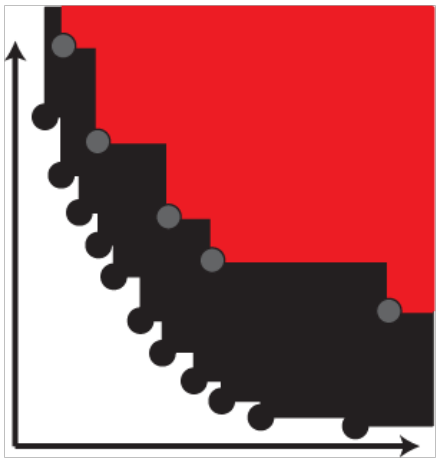

Figure 2.10 Example of the Hypervolume Indicator (Minimization) [80].

2.1.2.3 Frameworks for Metaheuristics

This section discusses the benefits of using a framework for metaheuristics. Frameworks are classified into two types: white-box frameworks and black-box frameworks. Components in black box frameworks can be reused together without having to worry about how they accomplish their tasks. White box frameworks, on the other hand, necessitate a more in-depth understanding of how components work. This allows for greater customization and freedom for users but comes with a steeper learning curve.

A framework is designed to provide a convenient testing environment for complex problems. However, multiple criteria must be met to satisfy the needs of developers who use the framework[123]:

- Code reuse: This minimizes introducing major changes to the existing code base. users should be able to develop a simple problem-specific code without needing deep knowledge of the framework.

- Adaptability: Adding features or metaheuristics should be possible. This is because most frameworks won't have every metaheuristic available to use but only the most known approaches (such as evolutionary algorithms).

- Efficiency: This includes both performances in terms of time/space complexity and also run time. Capabilities such as cross-platform support and parallel/distributed computing are of huge advantage.

jMetalPy [10] is a Python framework for single/multi-objective optimization that provides parallel computing capabilities and user-friendly visualization tools. This framework can be used to develop custom evolutionary algorithms/operators and define new problems.

We highlight jMetalPy [10], an object-oriented Python-based framework for single/multi-objective optimization with metaheuristic techniques. It is based on the jMetal framework (in Java) and inherits most of its features. What makes jMetalPy special is the fact that it uses the python ecosystem, which includes many popular libraries for data processing, analysis, visualization, and parallel computing. A wide variety of algorithms are implemented such as local search, Genetic Algorithms, and simulated annealing along with multi-objective Genetic Algorithms such as NSGA-II (non-dominated sorting genetic algorithm) [121] and SPEA2 (Strength Pareto Evolutionary Algorithm) [143]. The framework makes it simple to create new custom problems, genetic operators, algorithms, and test instances. Furthermore, known benchmark instances such as the ZDT [141] can be used easily. Performance indicators like the hypervolume metric are already implemented. Compared to DEAP [46] or PYMOO [12], jMetalPy is found to be more mature. We will be utilizing the jMetalPy libraries to implement our optimization algorithms, rather than coding the algorithms from scratch. This approach allows us to leverage the powerful features and capabilities provided by the jMetalPy framework, such as parallel computing and visualization tools, while focusing on the development of custom evolutionary algorithms and problem definitions specific to our project.

2.1.2.4 Multi-Criteria Decision Making

Multi-Criteria Decision Making (MCDM) is a subset of the Operations Research (OR) field that deals with decision problems involving multiple decision criteria. Mathematical optimization problems with multiple objective functions are a common example. Because most real-world problems have multiple objectives, choosing the best solution can be difficult, especially since these objectives frequently conflict with one another. We list some of the methods used to select the best solution from a set of final solutions.

Compromise Programming Compromise Programming tries to find the closest solution to the ideal point [92]. This is done by reducing the size of best-found solutions. The decision maker must provide weights to identify which objectives are more privileged than others. A decomposition function is later applied to find the best solutions. An example with 2 objectives is provided below in Fig.2.11.

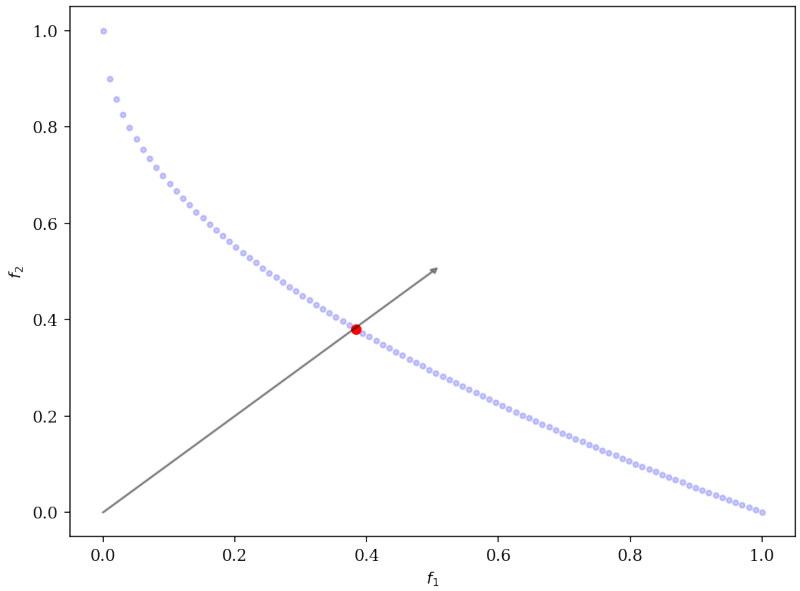

Figure 2.11 Example of compromise programming applied on Bi-objective problem (Minimization) [13].

Pseudo Weights Pseudo Weights calculates the normalized distance to the worst solution regarding each objective i [33]. Please note that for non-convex Pareto fronts the pseudo weight does not correspond to the result of an optimization using the weighted sum. However, for convex Pareto-fronts the pseudo weights are an indicator of the location in the objective space. The following equation provides the pseudo weight w_i for the i-ith objective.

$$w_i = \frac{(\mathrm{f}_i^{max} - f_i(x))/(\mathrm{f}_i^{max} - \mathrm{f}_i^{min})}{\sum_{m=1}^{M}(\mathrm{f}_m^{max} - f_m(x))/(\mathrm{f}_m^{max} - \mathrm{f}_m^{min})} \tag{2.4}$$

The steps are rather simple. First, we get nadir (ideal) points from the Pareto front. We proceed to calculate the normalized distance to the worst solution for each objective w_i. Finally, we find the closest solution to the normalized distance.

High Trade-off Points The high trade-off method, proposed by [31] involves selecting high trade-off points with neighboring solutions above

an upper threshold. This threshold was defined as the average trade-off plus twice the standard deviation using the entire Pareto-optimal set of solutions. After several knee points (high trade-off points) are identified, the one having the most balanced pseudo-weight vector among them is selected. Figure.2.12 and Figure.2.13 show an example of 2 and 3-dimensional solutions.

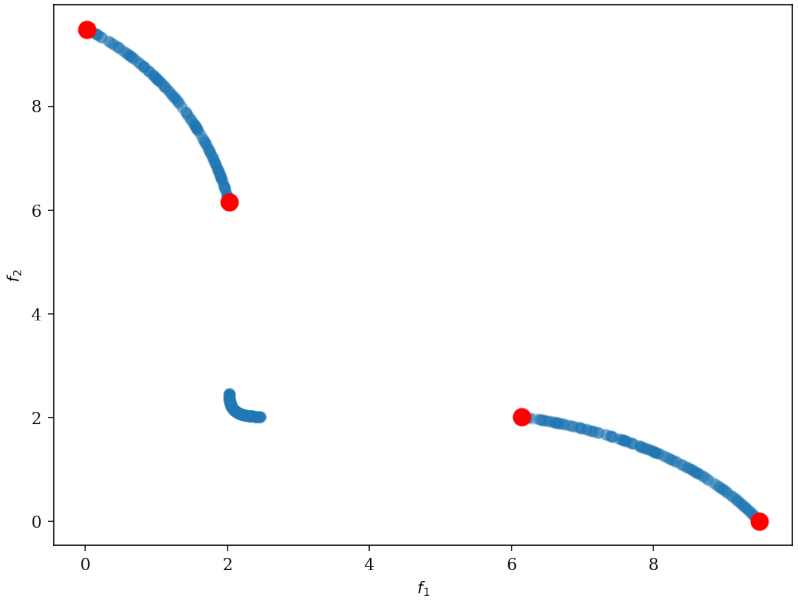

Figure 2.12 Example of High Trade-off Points applied on Bi-objective problem (Minimization) [13].

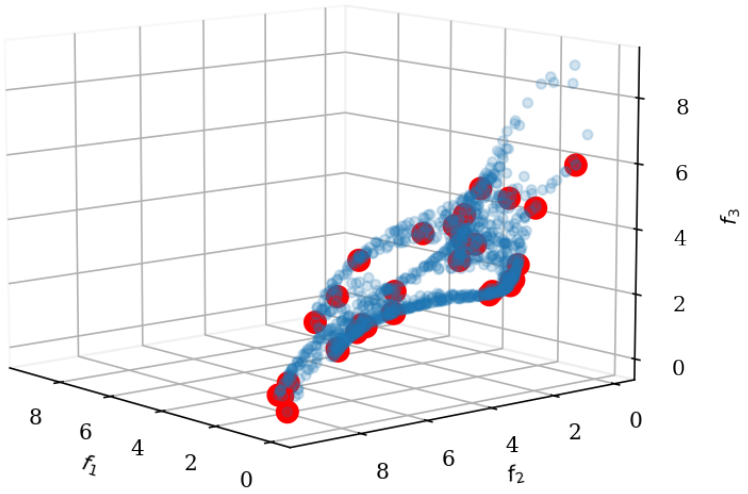

Figure 2.13 Example of High Trade-off Points applied on 3 Objectives problem (Minimization) [13].

2.1.3 Multi-Objective Genetic Algorithms

The complexity of MOPs becomes more and more significant in terms of the size of the problem to be solved (e.g., number of objectives, size of the search space). Moreover, the search time for solving these problems must be reasonable for most of the MOPs encountered in practice. The optimal solution for MOPs is not a single solution for mono-objective optimization problems, but a set of solutions defined as Pareto optimal solutions. There are four classes of multi-objective metaheuristics currently used in the literature.

- Scalar-based approaches: transforms a MOP into a mono-objective problem by aggregating various objectives f_i into a single objective function F. These approaches require the decision maker to have a good knowledge of his problem. Despite being simple and having a low computational cost, these methods lack diversity and are also sensitive to constraints.

- Criterion-based approaches: In these methods, search is performed by treating objectives separately. Both Vector Evaluated Genetic Algorithm (VEGA) and Ant Colony Optimization (ACO) are widely used criterion-based approaches.

- Dominance-based approaches: These methods use the concept of dominance and Pareto optimality to guide the search process. The objective vectors of solutions are scalarized using the dominance relation. Examples of dominance-based approaches include NSGA-II and III (non-dominated sorting genetic algorithm) and SPEA2 (strength Pareto evolutionary algorithm). In dominance-based approaches, a solution is considered dominant if it is better than another solution in at least one objective and not worse in any other objective. This helps in finding a set of solutions that are not dominated by any other solution, which is known as the Pareto front.

- Indicator-based approaches: In these methods, performance quality indicators are used to drive the search toward the Pareto front. An example of such an approach is IBEA (indicator-based evolutionary algorithm). Indicator-based approaches use specific measures to evaluate the quality of a solution, and the search process is guided by these measures to find the optimal solutions.

We will be using both Dominance and Indicator-based methods found in the literature due to their proven efficiency in handling complex problems [71, 55]. We cover each selected multi-objective metaheuristic in more detail below:

2.1.3.1 Non-dominated Sorting Genetic Algorithm II (NSGA-II)

The Non-dominated Sorting Genetic Algorithm (NSGA II) is implemented based on the original design of NSGA [121]. The algorithm follows the general outline of a genetic algorithm with a modified mating and survival selection [34].

In NSGA-II, solutions are selected front-wise. By doing so, there will be a situation where a front needs to be split because not all solutions are allowed to survive. In this splitting front, solutions are selected based on crowding distance. Figure.2.14 shows an example of multiple fronts for a bi-objective minimization problem. The figure demonstrates the concept of Pareto optimality, where solutions on the same front are considered equally optimal, and solutions on a lower front are more optimal than those on a higher front. The $W^{(k)}$ notation simply denotes the different fronts in the figure, with k representing the front number.

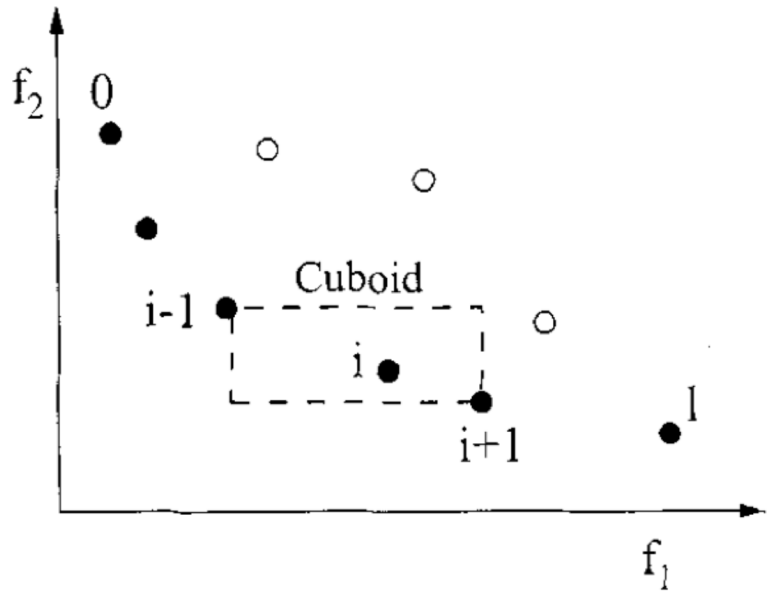

Figure 2.14 Example of a Bi-Objective Minimization Problem with Multiple Pareto Fronts

The crowding distance is the Manhattan Distance in the objective space. However, the extreme points are desired to be kept every generation and, therefore, get assigned a crowding distance of infinity. Furthermore, to increase some selection pressure, NSGA-II uses a binary tournament mating selection. Each individual is first compared by rank and then crowding distance. This process is illustrated in Figure.2.15.

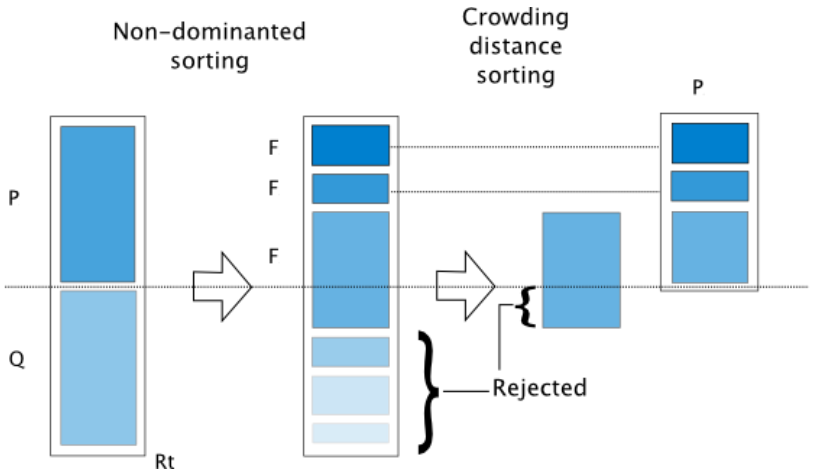

Figure 2.15 Crowding distance sorting example.

Overall, NSGA-II is one evolutionary algorithm that has the following three features:

- It uses an elitist principle, i.e., the elites of a population are given the opportunity to be carried to the next generation.
- It uses an explicit diversity-preserving mechanism (Crowding distance).
- It emphasizes the non-dominated solutions.

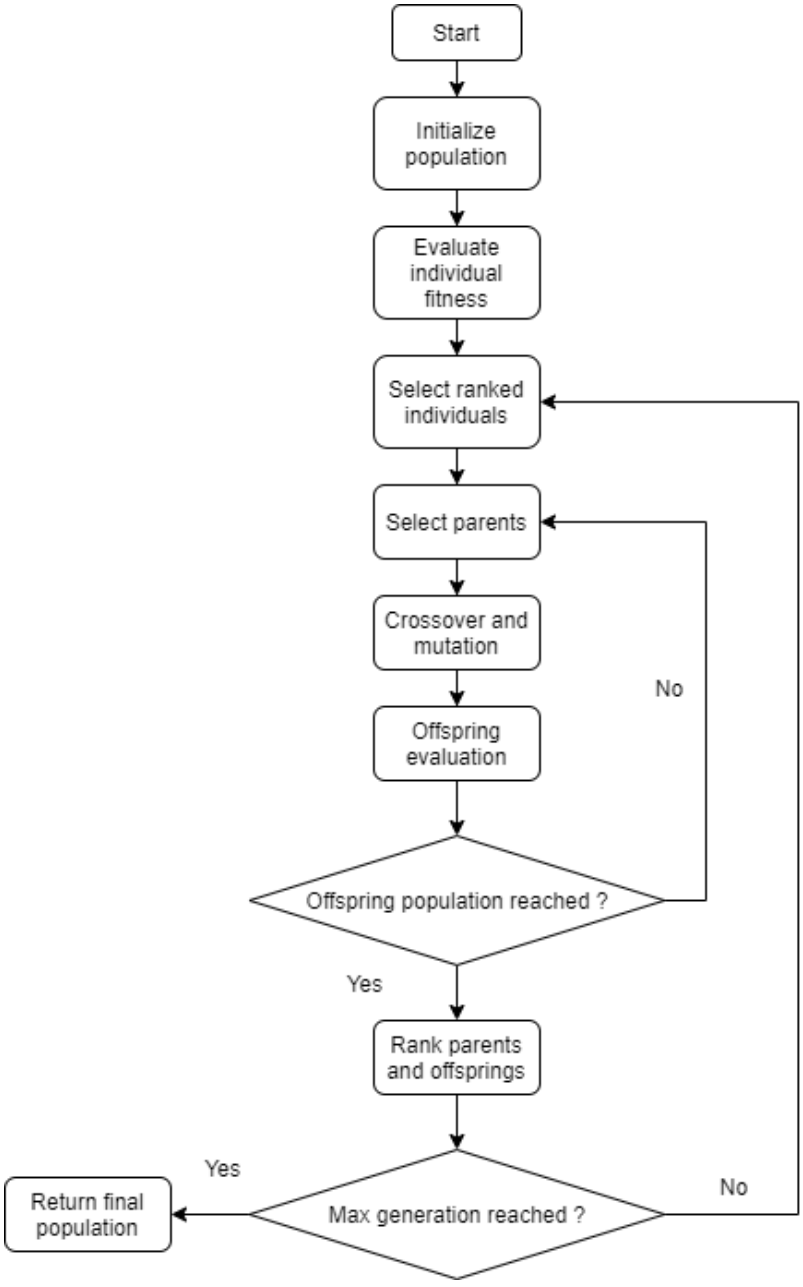

Figure 2.16 Flowchart of NSGA-II [104].

2.1.3.2 Non-dominated Sorting Genetic Algorithm III (NSGA-III)

Non-dominated Sorting Genetic Algorithm III is a genetic algorithm that solves multiple optimization problems simultaneously by applying a non-dominated sorting technique [35]. It uses a reference points-based selection operator to explore solution space and preserve diversity.

Most evolutionary many-objective optimization algorithms, for instance, NSGAIII or MOEA/D, start with a description of several predefined sets of reference points (or directions) on a unit simplex. NSGA III uses a different selection mechanism that relies on associating solutions to reference points before applying a niche preservation operation. A complete flowchart of the NSGA-III is illustrated in Figure.2.17.

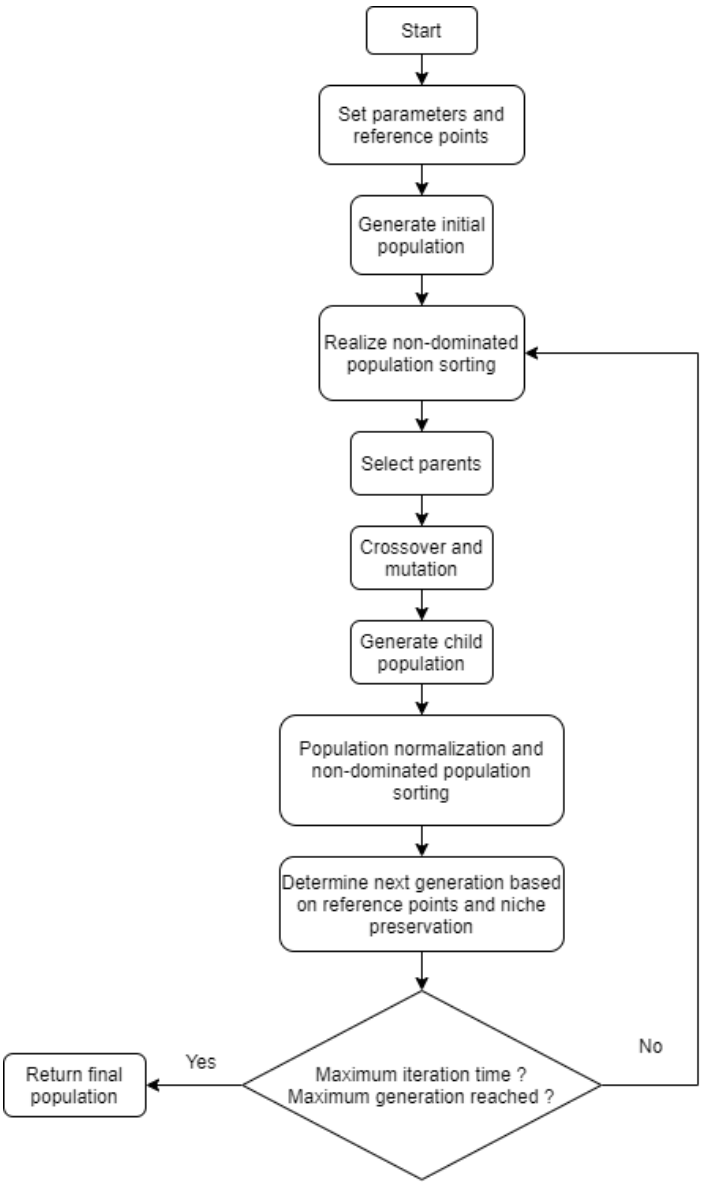

Figure 2.17 Flowchart of NSGA-III [126].

The first steps are similar to the non-dominated sorting done in NSGA-II. Figure.2.18 shows an example of a bi-objective problem with 3 fronts.

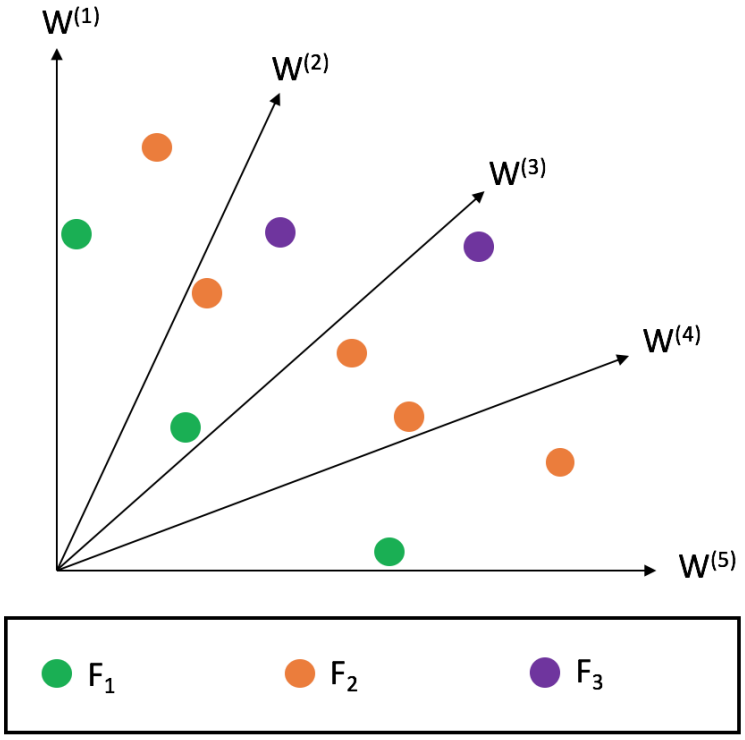

Figure 2.18 Example of Bi-Objective minimization problem.

From the splitting front, some solutions need to be selected. NSGA-III fills up the underrepresented reference direction first. If the reference direction does not have any solution assigned, then the solution with the smallest perpendicular distance in the normalized objective space is surviving. In case a second solution for this reference line is added, it is assigned randomly. Thus, when this algorithm converges, each reference line seeks to find a good representative non-dominated solution. This is illustrated in Figure.2.19. There are 5 reference directions in this example, but the number of criteria can vary depending on the problem being solved. S refers to the set of solutions in the splitting front, and F_L represents the set of solutions in the L-th front. The figure aims to demonstrate the process of selecting solutions in NSGA-III, where each reference direction seeks to find a good representative non-dominated solution. The solutions are chosen based on their perpendicular distance in the normalized objective space, and additional solutions for the same reference line are assigned randomly.

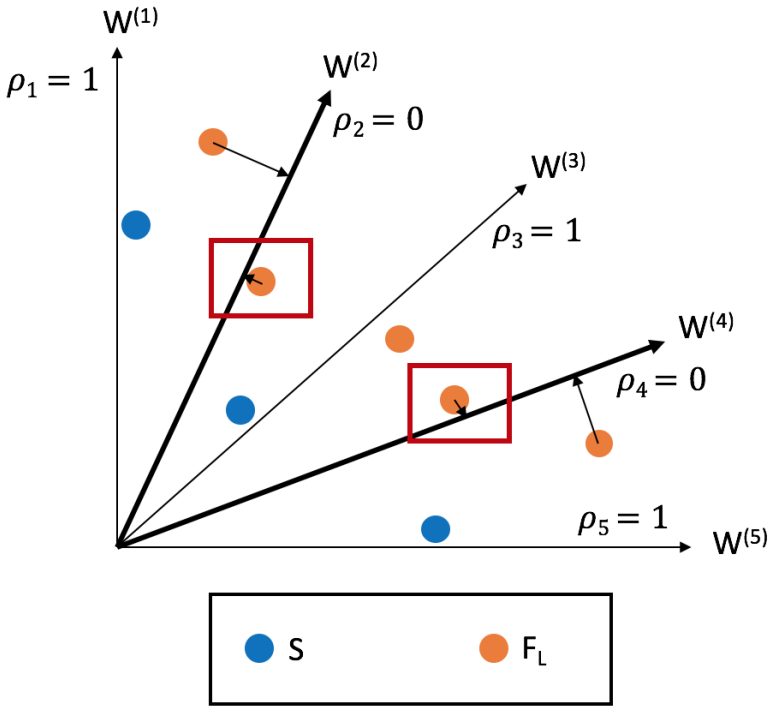

Figure 2.19 Solution selection in NSGA III.

2.1.3.3 Multi Objective Evolutionary Algorithm by Decomposition (MOEA/D)

MOEA/D is a multi-objective optimization algorithm [138], based on the idea of decomposing the multi-objective function into several scalar functions through aggregation functions and optimizing each objective in parallel with some effective evolutionary algorithms. The diversity of the population is controlled by the weight vector, which indicates each objective's importance.

MOEA/D requires an aggregation function to transform a multi-objective optimization problem into N sub-problems. The Tchebycheff approach is the most common in the literature due to its reliability. It's also used in this paper.

$\min g^{te}(x|\lambda, z^*) = \max_{1\leq i\leq m} \{\lambda_i |f_i(x) - z^*|\}$ subject to $x \in \Omega$.

Where z^* is the reference point and $z_i^* = \min\{f_i(x)|x \in \Omega\}, i = 1, ..., m$. While $\lambda = (\lambda_1, ..., \lambda_m)$ is the weight vector. Different sub-problems in MOEA/D have different weight vectors [51].

2.1.3.4 Indicator-Based Evolutionary Algorithm (IBEA)

IBEA was first suggested by Zitzler et al.[142], it presents an indicator concept to comprehensively evaluate the solution quality. A quality indicator is a general function that maps k Pareto set approximations to a real number, which is an extension of the dominant relationship. The indicators are utilized to measure the quality of solutions at each generation and to select the solution set with better quality value than that of the previous generation. Because the indicator is a comprehensive evaluation, IBEA does not require an additional diversity maintenance mechanism. This is the advantage of IBEA over MOEA/D and NSGA2 [90].

There are two typical IBEA indicators: hypervolume indicator and additive epsilon indicator. The first one is the hypervolume indicator, which is a comprehensive quality evaluation method of the solution set. It evaluates coverage, homogeneity, and universality simultaneously, and then obtains comprehensive evaluation results. A solution with a larger hypervolume score dominates the one with a smaller indicator score. The second one is the additive epsilon indicator [139], which is a binary quality indicator and is defined as the minimum distance needed for a solution to dominate another solution. Based on the distance in objective space, un-dominated solutions can weakly dominate other solutions. The additive epsilon indicator is easy to be used to compare the quality of two solutions relative to each other because of its strong separating capacity.

2.1.3.5 Strength Pareto Evolutionary Algorithm (SPEA-2)

SPEA 2 [143] is an extension of SPEA (Strength Pareto Evolutionary Algorithm) [144]. SPEA 2 first initializes a population of candidate solutions Pk, then stores the best solutions in an explicit archive Ak, separate from the population. To emphasize non-dominated individuals, SPEA 2 uses a combination of the dominance count and the dominance rank methods [82]. Each individual is assigned a raw fitness value depending on both the number of individuals it dominates and the number of individuals by which it is dominated. The density information is expressed as a function of the k-th smallest Euclidean distance in the objective space to the k-th nearest neighbor. The non-dominated individuals from the union of the archive and the current population are then updated.

In particular, if the number of non-dominated individuals is less than the pre-established archive size, some dominated individuals from the current pool form part of the archive. Otherwise, some individuals are removed from the archive using a truncation operator.

This procedure recursively removes individuals based on the nearest neighbor Euclidean distance. If there is more than one candidate solution with the same minimum distance, then the decision is made by considering

the second nearest neighbor, and so forth.

The mating pool used to generate the next population $Pt + 1$ is filled by the individuals of the updated archive selected on the basis of a given selection mechanism. The off-springs are then generated by a set of variation operators as in the previously discussed algorithms like NSGA-II and NSGA-III.

2.1.4 Parameter Tuning

Generally, the performance of evolutionary algorithms is sensitive to the choice of hyperparameters. Hyperparameters can be defined as values we can tune manually from the algorithm itself, like the probability of mutation, crossover, learning rate, number of solutions, etc. The optimal parameter values are determined primarily by the problem and test instances that the user wishes to solve.

Unfortunately, there is no universal parameter tuning approach that applies to all problems; however, two distinct strategies exist:

- Off-line: The values of parameters are fixed before the execution of the metaheuristic. This strategy does not provide real-time parameter updates, but it can incorporate all available data into its parameter-tuning phase at the time of execution.

- Online: The values are updated adaptively during the execution of the metaheuristic. The primary operational challenge of this strategy is that data must be processed incrementally. Because of the amount of computation involved in each iteration, real-time results are not always possible and can provide lower scores.

After considering the benefits and drawbacks of each parameter optimization strategy, we choose the offline method due to its robustness and the fact that we must test on the entire dataset to account for changes in the data state. The irace package [94] provides an automatic configuration tool for tuning optimization algorithms given a set of instances of an optimization problem. It implements the iterated racing procedure, which is an extension of Iterated F-race [11]. Figure.2.20 gives a general scheme of how irace works.

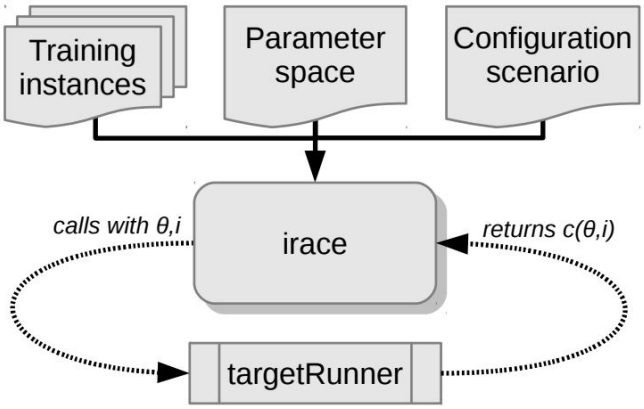

Figure 2.20 Flowchart of irace workflow [94].

irace operates as follows: it receives as input a parameter description file ("parameters.txt") that defines the parameters of the target algorithm that will be tuned. Users can define parameter names, types, possible values, conditions, etc.

Training and test instances for which the parameters must be tuned are placed inside the "Instances" folder. Additional options specific to irace can be found inside the "scenario.txt" file. This can include the time limit, the maximum number of runs, execution file path, parallel computing, etc. irace moves through the parameter search space in search of good-performing algorithm configurations, for doing so irace executes the target algorithm on different instances and different parameter configurations.

To execute the target algorithm with a specific parameter configuration θ and instance i a "targetRunner" must be provided. The "targetRunner" acts as an interface between the execution of the target algorithm and irace, it receives the instance and configuration as argument and must return the evaluation of the execution of the target algorithm. Typically, "targetRunner" includes the path to the algorithm execution path. Note that irace assumes minimization by default, if a user wishes to maximize the evaluation score, they need to multiply the score by -1 within their algorithm.

To fairly compare the performance of each algorithm, the i-race package is used to find the best-performing parameters (elite configurations) based on the average best-chosen performance metric across different test instances.

2.2 Timetabling Problems

Timetabling is one of the most tedious and time-consuming tasks for any institution, and it has consequently received considerable attention over several decades.

Institutions and universities dispose of different environments and working places, which implies various constraints and real cases to be solved. Researchers throughout the years tried to bring forward different solution methods and procedures to face the increasing difficulty of these problems. Although the problem has broad similarities across different institutions, each use case is unique in general, because institutions have different rules and use different terms to evaluate the quality of solutions.

Terminology	Definition
Event	An activity (course, training, exam, etc.) to be scheduled.
Time slot (time period)	An interval of time (with a starting and ending date) in which events can be scheduled.
Resource	Resources required by events. Can be a classroom or special equipment (i.e., screen projector)
Constraint	A restriction to schedule the events. Room capacity can be considered as a constraint for example.
Conflict	Two events sharing at least one common resource and scheduled at the same time slot.

Table 2.1 Timetabling common terminology.

For any timetabling problem, several constraints must be satisfied for a timetable to be feasible. There are usually two main types of constraints which are: Hard constraints consist of a set of essential requirements which must be met in full. A timetable that meets essential constraints is feasible and thus can be called a valid timetable. If hard constraints are broken, the timetable is no longer valid. Hard constraints may include, for example, forbidding trainers or classrooms from being scheduled to different courses at the same time. Generally, the solution for most timetables is to satisfy hard constraints. Soft constraints consist of a set of desirable requirements which does not have to be fully met. This means that these constraints can be violated, and the timetable will still be valid. Soft constraints can include specific features such as personal preferences, events or resources, and geographical location. The satisfaction of soft constraints can make one valid timetable better than another [135]. According to Ross et al. [112],

hard and soft constraints can be categorized into four main classes which are listed below:

1. Unary constraints, which concern a single event. For example: "Event A must be scheduled on a specific day".
2. Binary constraints involve two events. For example: "Event A must be allocated after Event B".
3. Capacity constraints, typically involve room capacities. For example: "Events must be scheduled in rooms that possess sufficient capacity".
4. Event spread constraints; they handle the distribution of events within the timetable. Events can be close ("clumping together") or separated ("spreading out").

2.2.1 Educational vs Professional Timetabling

A variety of articles published in recent years address the problem of course scheduling and many other related such as exams, conferences, universities, and school schedules. We will discuss the difference between them and explain why the problem we're dealing with in this research is different.

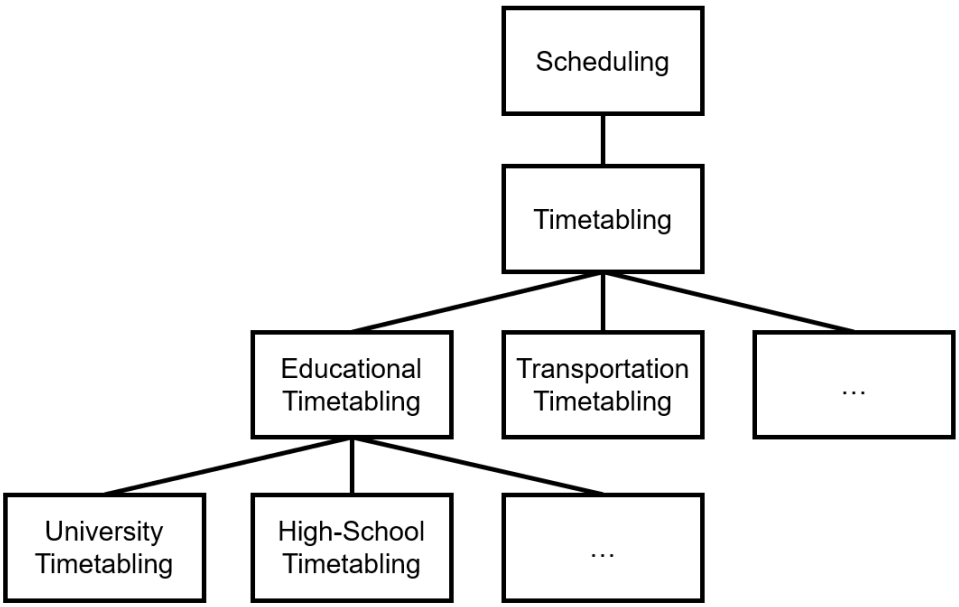

Figure 2.21 Family of known scheduling problems. An extension from [92].

2.2.1.1 University Timetabling

University timetabling is part of the broader family of scheduling problems. University scheduling addresses the planning of lectures and seminars at

universities [113]. In practical terms, the goal is to assign teachers and courses to rooms and periods.

The timetable has a huge effect on students' and staff's weekly schedules because it restricts their other activities. Many of the resources of a university are dedicated to teaching, both in terms of the staff time required to prepare and produce materials, and in terms of space where universities must assign lecture and classroom space. It is therefore necessary to build high-quality timetables that meet the needs of students and staff and allow efficient use of the resources. The term teachers refer to professors and lecturers, while the duration of a period is defined by the length of one lesson. The planning period usually comprises one week, and the planning cycle repeats each semester or term. A course means a group of students to be instructed on a specific topic. Many topics are covered in several lectures or classes and may even be given by several teachers in parallel. The first constraint is allocating all courses to the schedule. Another straightforward requirement is to prevent double assignments of courses or rooms (cannot be assigned more than once in the same time period). The availability and capacity of rooms are also considered. In addition to the feasibility of the solutions found, a second objective addresses preferences established by the teachers for periods as well as rooms, in most cases this preference is to be respected as much as possible while preserving feasibility. In some cases, this is considered a soft constraint.

2.2.1.2 School Timetabling

School scheduling addresses the planning of school lessons which involves the assignment of teachers, subjects, and courses to rooms and periods [114]. Furthermore, in school timetabling, the assignment of rooms is only a minor problem, as every class will have a dedicated classroom. In schools, most courses could be handled by several teachers, so the situation here differs from that in university scheduling where teachers and courses can be identified with each other. A course denotes a class of students. A period represents a lesson. The planning horizon comprises a week. The results of the planning process are valid for the complete school term. Most subjects are taught in several courses and can be administered by several alternative teachers. Hence, in school scheduling courses, subjects, and teachers cannot be identified with each other, so all these planning objects must be dealt with explicitly. In addition, no concurrent events constraint is applied to the school timetable as well. A new constraint known as schedule compactness can be found in some works where the schedule may contain no idle periods (or empty holes).

2.2.1.3 Professional Course Scheduling

Courses delivered in a professional environment, aim to confer expertise or skills and thus qualify as training rather than education. They are business-centered and goal-oriented, often preparing trainees or company employees for evaluation or certification. Training material is therefore following a specific learning path. This means that usually, a greater number of constraints and objectives will have to be dealt with in making the timetable.

The problem discussed in this work might fall in the same category of academic timetabling but in fact, it totally differs in many aspects. To add more clarity, the problem handled in this article falls under the "professional course scheduling" category [52] which is separate from academic timetabling.

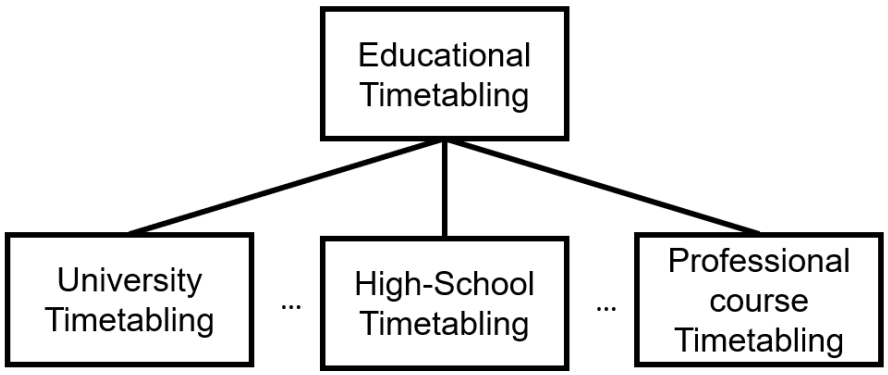

Figure 2.22 Professional Course Scheduling belongs to the family of Education Timetabling [52].

Both problem categories have common constraints related to resource characteristics (room size, teacher's preferred days, etc.). In professional course scheduling, teachers can be referred to instructors, tutors, or trainers charged with conducting the training. Several other differences are worth pointing out:

Objectives: Most academic timetabling works are single-objective (e.g., maximizing planned events, minimizing empty classrooms) while the problem considered in this work has multiple competing objectives.

Planning window: Academic timetabling focuses on covering small periods of time (typically one or two weeks) with courses that are offered on a regular basis over a longer length of time (semester or year). We will be dealing with longer spans of time in this article (months or years).

Feasibility: In exam-timetabling works, most exams if not all, must be scheduled, but in our circumstances and due to a lack of available resources, it's possible to find no acceptable timetable for some events.

Locations: Another key difference compared to academic timetabling is the accessibility to resources in other sites (national/international).

Professional scheduling does not involve the assigning of employees to classes, as the organization or the training provider handles this.

- Different teachers may follow the same course. However, some trainers will typically be eligible to cover only a few courses which require advanced skills.

- Whereas the scheduling of academic courses seeks to find regular schedules that can be replicated at regular intervals, often one week. Professional course schedules, on the other hand, are non-repetitive and typically span a longer planning period that can range from months to even years.

In certain instances, longer courses are broken into small chunks, following a defined order. The timetable needs to include all the chunks while acknowledging the constraints that are required by the course. For example, a course may need a special classroom or equipment. Also, a course may require to be scheduled in specific hours or days. Each course has its own duration, which can range from hours to days.

Many limitations are similar to university or high school timetabling problems (resource availability, no concurrent resource allocation). By resource, we mean teachers, classrooms, and equipment. Trainings often adhere to a time frame defined by a start and end time that must be set. One issue with educational scheduling is having one objective defined (e.g., maximizing scheduled courses, exams, and classrooms). Nonetheless, instead of trying to find a schedule for all the subjects, which is difficult in certain situations, we must accept the fact that some courses simply cannot be scheduled. Therefore, the main goal isn't to try to find a feasible solution but to also consider other objectives that can lead to a better quality of the timetable such as an even distribution of teacher workload for example. Furthermore, because timetables frequently need to be modified or fully remade (e.g., school timetables are often redesigned at the beginning of each academic year, bus timetables need to be adapted to deal with new road configurations and bus stops, and new courses need to be arranged, etc.), timetables will have to adapt to such changes. Nevertheless, unforeseen disturbances can occur, and changes may be needed. For example, if at any time a resource is inaccessible, an alternate solution must be sought, and the timetable must be updated.

2.2.2 Metaheuristic Approaches to Timetabling Problems

In recent years, the issue of Timetable Scheduling has been attracting considerable attention from the scientific community. This led to the creation of a series of conferences called the PATAT (Practice and Theory of Automated Timetabling) in 1995, which has since been held every two years. In 2002, PATAT helped establish the International Competition of Timetabling (ITC 2002) and the famous ITC 2007. The literature has already suggested many different methods for solving timetabling problems such as graphs, where the events are ordered, and then assigned sequentially into valid time slots [18]. Metaheuristic methods were also used, such as Genetic Algorithms (GA) [36], simulated annealing [8], tabu search [17], and other heuristic approaches, that tend to be inspired by nature or by natural processes and phenomena.

2.2.2.1 Main categories of optimization algorithms for timetabling

According to Lewis et al. [89], meta-heuristic algorithms for timetabling can be generally categorized into three main classes:

One-stage optimization algorithms: this strategy looks for a solution that can adequately satisfy both hard and soft constraints. The violation of both hard and soft constraints is allowed.

The major advantage of the one-stage approach is that it is comparatively easy to implement and change constraint weights. Typically, hard constraints have much higher penalties when violated than soft constraints. At the same time, the search must seek to find feasibility and optimality.

However, this approach has not been very successful at finding solutions for complex problems, because of several disadvantages:

- The choice of weights is often arbitrary; there is no generally applicable scheme for setting the weights and they are often very specific to the problem at hand.
- A small change in the timetable often results in a large change in the penalty score (i.e. when a hard constraint has been violated), making the fitness landscape more difficult to navigate.
- The incorporation of soft constraints could take the solution further away from attractive (feasible) regions of the search space.

Two-stage optimization algorithms: during the first stage, only hard constraints are considered, after a feasible timetable has been constructed,

the second stage aims to minimize any soft constraint violations, without breaking the feasibility of the timetable (neighborhood operations must not violate hard constraints).

The success of this approach depends on two criteria:

- It must be possible to find an initial feasible timetable in a short time. If feasibility is very hard or even impossible to achieve, the algorithm might never reach the second stage; in this case, a one-stage approach could produce at least a solution with a suitable compromise between hard and soft constraints violation.
- During the second stage, search space must not be strongly constrained otherwise neighborhood moves might find it difficult or even impossible to reach another possible solution.

Algorithms which allow relaxations: violations of hard constraints are prevented by relaxing the problem (e.g., adding more available days, more available time periods). This approach relaxes the problem by either initially leaving some events unassigned, which cannot be feasibly scheduled anywhere, or by opening additional time slots when an event cannot be assigned to any of the existing time slots. The produced timetables consequently never contain any hard constraint violations, but don't necessarily include all events or have the required number of times-lots. The goal of the relaxation algorithms is to minimize the soft constraint violations as well as simultaneously trying to remove all of the relaxations. Whereas the quality of a candidate timetable during the first stage of the two-stage approach is measured by the number of hard constraint violations, the quality of the relaxed timetable is measured by the distance to feasibility, i.e. the number of unassigned events or the number of excess time-slots opened.

2.2.2.2 State of the art of professional course scheduling

In this part, we outline the contributions of works that address professional and educational timetabling issues in real-world settings. While the subject we're addressing falls within the category of "professional course scheduling," as defined by Haase et al. [52], most techniques focus on educational timetabling. Table.2.2 provides a summary of relevant works to our research.

Authors	Opt. Cat	Opt. App	Obj	Const	Init	Operators	Evaluation	Dataset
Derigs et al.[36]	Two-stage	GA	3	8 Hard 1 Soft	Heuristic	PMX, CX, OX and OX2 (Cross) One/Two-point (Mut)	Number of scheduled courses / teaching days	Ford Historical Data
Matias et al.[98]	One-stage	GA GLS	1	7 Hard 7 Soft	Random	Multi-point (Cross) Custom-Swap (Mut)	Weighted Cost Function Jain's Fairness Index	Caraga University Historical Data
Sorush Niknamian [103]	Two-stage	NSGA-II	8	8 Hard 6 Soft	Random	Custom (Cross) Custom (Mut)	Weighted Cost Function	Azad University Historical Data
Haase et al.[52]	Two-stage	LS	1	10 Hard	Heuristic	-	Total profit margin	Lufthansa Historical Data
Lü et al.[96]	Two-stage	TS ILS	1	4 Hard 4 Soft	Heuristic	Simple and Kempe Swap (Neigh)	Weighted Cost Function	ITC 2007
Mushi A.R [101]	Two-stage	TS	1	5 Hard 5 Soft	Heuristic	Time Move and Course Swap (Neigh)	Weighted Cost Function	Dar As Salaam University Historical Data
Bellio et al.[8]	One-stage	SA	1	3 Hard 4 Soft	Random	Move Lecture, Swap Lectures (Neigh)	Weighted Cost Function	ITC 2007
Irfan et al. [70]	Two-stage	ACO	1	5 Hard 4 Soft	Heuristic	Swap (Neigh)	Weighted Cost Function	ITC 2007
Czibula et al. [27]	Two-stage	IP LNS	3	13 Hard	Heuristic	Swap neighborhood	Weighted Cost Function	Ausgrid Historical Data
Goh et al. [49]	Two-stage	TS, SA	1	6 Hard 3 Soft	Heuristic	Kempe Chain, Swap and Move Neighborhood	Weighted Cost Function	ITC 2007, ITC 2002, Socha

Table 2.2 Overview of related works.

At first glance, Table.2.2 shows that the majority of works manage timetabling problems utilizing a two-stage technique. This is the most common type of optimization strategy for timetabling; as we've seen, this method aims to satisfy only hard constraints before applying optimization methods to improve solutions.

Metaheuristics such as Genetic Algorithms (GAs), Tabu Search (TS), and Simulated Annealing (SA) with other local search techniques (LS, ILS, LNS) were applied in all of the works. This variety of options demonstrates that the challenges are distinct and that there is no single best method that performs better than others. While the majority of the works deal with a single objective, certain works in the professional timetabling problem category, such as Czibula et al. [27] and Derigs et al. [36], are multi-objective, confirming that this category of timetabling problems is rarely expressed in a single objective.

Moving on to popular initialization strategies, this phase is critical for providing initial feasible timetables. We find that works are divided into two categories: random techniques that produce many solutions that are later fixed, and heuristics that try to only supply valid solutions while limiting the time required to repair infeasible timetables. In terms of timetabling operators, we discover a variety of genetic operators such as the Cycle Crossover (CX), the Order Crossover (OX), and the simpler 1- or 2-point crossover/mutation. Various local search neighborhood operators, most notably the Swap, Kempe, and Move operators, are also utilized.

Moving on, we'll look at how objectives are quantified, and we'll see that most works aggregate their objectives into a single cost function. Despite being a prevalent approach in the literature, this strategy does not

provide the option to choose which objectives to focus on when picking final solutions. Finally, studies differ in terms of the data they used to test their techniques. While educational works emphasize both ITC2007 and historical data from schools, professional timetabling papers emphasize their company's historical data [36, 52, 27]. In the parts that follow, we go through the contribution of each work in further detail.

(1) The first work resembles the problem found at Mandarine Academy, done by Derigs et al. [36], which utilized a multi-objective genetic algorithm to reduce expenses and increase the number of planned events at the Ford Service Organization.

The authors developed a web-based decision support system that allows planners to generate, analyze, and compare various timetables suggested by the system. Their problem handles the scheduling of courses and trainers over a period of 3 months with 8 hard constraint categories and 1 soft constraint.

The authors define a feasible solution as a timetable that considers the following 8 hard constraints:

1. A trainer is allowed to teach courses in which he is specialized only.
2. A trainer cannot teach more than one course at a time.
3. A trainer cannot be scheduled for a period in which he is not available.
4. At every time a room can only be assigned to at most one course.
5. For its entire duration every course is assigned exactly one suitable trainer and one suitable room as well as the necessary units of equipment.
6. Every course is scheduled without a weekend break.
7. Every course has to end before its deadline.
8. Rooms and trainers are not available during course preparation, which is performed immediately before the start of a course. Here, a weekend between preparation and the course is allowed.

They also defined one soft constraint:

1. For every trainer the number of teaching days should be at most 75% of the number of potential teaching days.

Since this work deals with a real-world complex professional course scheduling problem, three different objective functions are used:

1. Maximize the total number of courses that are offered. In contrast to most educational timetabling problems, not all courses can be scheduled.

2. Maximize the weighted number of courses that are offered. For each course, a priority is defined. Some courses are more privileged.

3. Maximize the number of teaching days.

Their work follows the two-stage optimization approach. The first stage is a construction heuristic that starts with an empty schedule and a list of courses to schedule ordered by a priority scheme defined by the authors. The heuristic starts by finding a start/end time for a suitable room, then checks for trainer availability and finally equipment. If everything is available a schedule for that course is made. Any infeasible course remains unscheduled. The second stage involves Genetic Algorithms that aim to provide a fair workload among trainers.

To test their approach, 2 Data-sets with historical data are used:

1. Cologne 2/2003: 96 courses, 5 trainers, and 6 rooms with 217 training days.

2. Cologne 3/2003: 110 courses, 6 trainers, and 6 rooms with 225 training days.

Their method which takes 2 minutes of CPU time was found to be faster and more accurate than a manual solution that takes about six weeks of manpower. For the Cologne 2/2003 dataset, their approach was able to schedule all courses. For the Cologne 3/2003 dataset, GAs were able to schedule 81 courses with 150 training days. The provided solutions were approved by an expert as valid.

(2) Matias et al. [98] tackled the university course timetabling problem by combining a Genetic Algorithm (GA) with Guided Local Search (GLS). The initial population of timetables is generated randomly using genetic operators, if a solution is found unfeasible, a guided repair is employed. Authors consider a timetable valid if it respects all defined hard constraints. Hard and soft constraints are listed below:

1. A teacher could not attend two classes at the same time.

2. A teacher attends only one course in one room at each time slot.

3. At each daily timeslot in one classroom, only one group of students and one teacher could attend.

4. The capacity of the classrooms should be considered.

5. All courses required for each student must be scheduled,
6. An exact number of teaching periods of the course required in the curriculum must be completed.
7. An uninterrupted period required for a class should be exactly assigned exactly on a given day.

The soft constraints are described as follows:

1. Teacher's priority timeslots are considered.
2. Teachers might have unavailable or pre-assigned periods.
3. Teachers are limited to 4 preparations.
4. Teachers are limited to 21 or fewer units' load.
5. Maximum teaching hours in the classroom is 4.
6. A group of students should be vacant for at least one period a day.
7. Rooms might have unavailable or pre-assigned periods.

In case some solutions are invalid, a repair operator will focus on satisfying hard constraints to make the solution feasible. Their objective function was to reduce the number of hard and soft constraint violations (Minimization). To evaluate their approach, the authors used a real-life dataset from the department of information technology at Caraga State University.

The dataset comprises more than 100 courses that are needed to be assigned to a timetable with 45 time slots corresponding to 6 days of 9 hours each.

The results show the proposed method is better compared to an older implementation using only a simple genetic algorithm. The approach was able to generate a timetable with a 61% fairness index (using Jain's Fairness Index) and no hard constraint violations, compared to a 15% fairness index when using the older approach.

(3) Another work used Genetic Algorithms, Niknamian [103] applied a multi-objective non-dominant sorting genetic algorithm (NSGA-II) to schedule courses in the Department of Industries at the Islamic Azad University. This two-stage approach uses a random approach to generate solutions that are then grouped in a weekly program and assigned to specific time slots. The produced solutions must respect defined hard constraints to be valid. The author considers the following hard constraints in their work:

1. Curriculum of lecturers based on the hours of their university attendance.
2. Each lecturer cannot tutor more than one lesson during each time period.
3. Each lecturer cannot teach for more than a specific time per day.
4. Creating provisions for all the groups of a tutorial.
5. A classroom does not have the possibility of holding more than one tutorial at every time period.
6. Classroom Capacity.
7. Lecturers can only teach for specific periods throughout the week.
8. Obligatory attendance of a faculty member for impromptu counseling

The author considers the following soft constraints in their work:

1. The preference of engaging faculty members rather than having visiting lecturers.
2. Minimizing the presentation of tutorials at the closing time period of each day.
3. Each lecturer cannot teach for more than a specific time per day.
4. The conduction of synchronous and optional courses.
5. The excellence of lecturers.
6. Maximizing the deployment of lecturers.
7. Maximizing the utilization of a classroom by a lecturer.

Following this, NSGA-II tries to optimize solutions based on defined objectives and soft constraints. Furthermore, the author introduces two custom genetic operators that are adapted to the chosen solution's representation. To evaluate the quality of the proposed approach, the authors use the following metrics:

1. ND_t: Ratio of days for the attendance of lecturers.
2. Q_t: Quality level of lecturers in programming.
3. S_t: Ratio of simultaneous courses.
4. R_t: Ratio of lessons rendered in the final time period.

(4) Haase et al [52] developed a construction heuristic with Local Search for solving a professional course scheduling problem at Lufthansa Technical Training. Their objective function was to maximize the profit margin induced by a schedule.

The author's approach consists of a heuristic algorithm that employs a serial scheduling scheme and uses deterministic priority rules to solve the selection or assignment conflicts. The authors defined a total of 8 different priority rules that govern the way courses are selected for scheduling. Their method is essentially an iterative procedure that in each Iteration constructs several solutions. If the best so-found solution is better than the last best one, the search process is intensified in the surrounding regions; otherwise, the procedure terminates.

For the test phase, the planning horizon was restricted to six months. The authors compared their different priority rules to see which approach was able to perform better. A second test was aimed at finding the performance of the local search method.

The authors compared CPU-Time, iterations required, and the percentage of courses scheduled. In their final comparison, the best solution was compared against the manual approach.

Their approach was able to maximize the total profit margin, but the manual solution was better in terms of the average number of days and scheduled courses. Finally, the time required to construct an operational schedule was heavily reduced by the new approach.

(5) Lü et al. [96] applied the Tabu search algorithm to solve the university course timetabling problem. This algorithm is ranked as one of the five finalists for the 3rd track of the Second International Timetabling Competition ITC–2007.

The authors presented an Adaptive Tabu Search (ATS) for solving the curriculum-based course timetabling problem. The proposed algorithm follows a general framework composed of three phases: initialization, intensification, and diversification.

1. First stage (Initialization) concerns the initial solution, a sequential greedy heuristic starting from an empty timetable, where assignments are constructed by inserting one appropriate lecture into the timetable at each time. The goal for this is to have a feasible timetable (a timetable that respects all hard constraints).

2. Second stage (Intensification), uses a feasible timetable to optimize the soft constraint scoring function without breaking the hard constraints. This is done by using two different neighborhood structures (Simple Swap and Kempe Swap). The algorithm stops when the best solution cannot be improved within a given number of moves.

3. Third stage (Diversification), this stage combines tabu search with iterated local search (ILS) which provides diversification mechanisms to guide the search to escape from the current local optimum and move toward new promising regions in the solution space using Penalty-Guided Perturbation Strategy.

(6) Mushi [101] also used Tabu Search for the academic course timetabling problem. His work uses real-world data from the University of "Dar es-salaam". The objective function is to minimize violated constraints. The author also defined weights for each violation of hard (5) and soft (5) constraints.

For the initial timetables, a heuristic algorithm assigns each event (ordered from highest to lowest duration) to the earliest possible feasible time slot and first available room. The initial solution can be completely feasible or partially feasible.

The author used two different neighborhoods, first one is swapping random events that belong to the same group. The second neighborhood is assigning random time slots to random events. For the aspiration criteria, any solution which brings an improvement by a certain degree is accepted. The author criticized using a fixed number of iterations as a stopping condition because the algorithm can run for a long time without improvement just to complete the set number of iterations.

Instead, the algorithm stops after 1000 iterations without solution change. To evaluate the proposed approach, a comparison between the use of the two moves is conducted and it shows that swapping events performs much better than randomly assigning time slots. Also, the quality of the manual solution is much lower (71%) compared to the automatically generated solution (99%).

(7) Bellio et al. [8] applied a one-stage approach using the Simulated Annealing (SA) algorithm for the Curriculum-Based Course Timetabling problem (CB-CTT).

In their approach they used two different neighborhoods:

1. Move Lecture (ML), which changes the period and the room of one lecture.

2. Swap Lectures (SL), which swaps the period and the room of two lectures of distinct courses.

For the objective function, they calculated the cost of a solution using the weighted sum of hard and soft constraints violation. For simulated annealing, a cut-off-based temperature cooling scheme was applied, and a maximum number of allowed iterations was used as a stopping condition.

The evaluation in this work shows that their approach outperformed the results of previous works using the ITC2007 dataset. Their approach was able to surpass multiple works on 10 instances out of an overall 21 instances. The authors explain that this was possible by just tuning the parameters of the Simulated Annealing algorithm.

(8) Patrick et al. [106] developed a greedy Ant Colony Optimization (ACO) strategy for the Curriculum-Based Course Timetabling problem. Their approach uses ACO for both construction and improvement phases, which is different from other ant systems that find a solution in a single phase.

Two stages were presented in their work:

1. First stage is dedicated to finding an initial solution where two algorithms are used.
    - First algorithm is Ant Walk, which assigns time periods and rooms to courses in a greedy approach.
    - Second algorithm is Ant Colony, which minimizes constraints violation from the generated timetable in the first algorithm.
2. the second stage is the improvement phase where Ant colony and hill climbing strategies are used to improve found solutions. One neighborhood move is used, Swapping two events.

To evaluate their approach, the authors compared their work against eight other works, their approach was able to outperform some works but not all. The dataset used was the ITC 2007.

(9) Czibula et al. [27] presented an Integer Programming (IP) approach to solve a multi-objective timetabling problem. Their work was conducted at Australia's largest electricity distributor "Ausgrid".

Despite the difficulty in solving large-scale Integer Programming (IP)-based models, the authors started by decomposing a larger problem into multiple sub-problems, one for the class timetabling and one for the trainer rostering. Still, the model was too large to be solved in an acceptable time given real-world data.

To tackle this, the authors proposed a three-stage heuristic approach. The first stage produces an initial feasible class timetable, this is accomplished by successively solving the class timetabling IP model for one-course instance at a time. The most demanding courses (longer duration, modules, and specific room and resource requirements) are selected first and the least demanding courses are scheduled last. Authors state that starting with complex courses first, resulted in a high-quality initial timetable.

The second stage attempts to improve the class timetable produced in the first stage by using the Large Neighborhood Search (LNS) heuristic. Neighborhoods are selected by analyzing the timetable and identifying which scheduled course instances negatively affect the objective values. The authors used the Swap neighborhood move with LNS. The third stage allocates individual trainers in the timetable produced in stage 2. The IP model for the rostering sub-problem is small thus it can produce an optimal solution faster.

Being a large-scale real-world problem, they had multiple objectives to consider:

- Minimize the number of expected students (not accommodated).
- Maximize the room rental revenue.
- Minimize the number of room swaps in the timetable.

The authors used IBM ILOG CPLEX and three timetable test sets with different densities: Low (L), Medium (M), and High (H). The authors have had success as their heuristic was able to find the optimal timetable for 13 of the 16 cases for the low-density timetable, 8 out of 16 in the medium-density timetable, and 9 out of 16 cases in the high-density timetable.

(10) Goh et al. [49] tackled the post-enrolment course timetabling problem by using a two-stage approach that combines both Tabu Search (TS) and Simulated Annealing (SA) algorithm.

1. First stage was dedicated to finding a feasible solution using Tabu Search (TS) algorithm with sampling and perturbation. Two different neighborhoods were used during this stage, Swap and Kempe chain. The stopping condition was either:
    - Finding a feasible solution that respects all hard constraints.
    - Execution time exceeded a certain duration.
    - Number of iterations is exceeded.
2. Second stage uses Simulated Annealing with reheating to improve soft constraint violations. This stage relies on the cost to determine whether the search is stuck in a local optimum. Three different neighborhoods were used during this stage, transferring an event to a slot, swap events if they didn't break hard constraints, and finally Kempe Chain between two events in different time slots.

Their objective was to satisfy all hard constraints and minimize soft constraint violations. To evaluate their approach, the authors used three different data sets (Socha, ITC2002, and ITC2007). The evaluation was made in two stages.

1. For the first stage, a comparison between using a simple Tabu Search (TS) and using Tabu Search with Sampling and Perturbation (TSSP). TSSP performed better than TS in all data sets.

2. Second evaluation, a comparison between the author's approach using Simulated Annealing with Reheating (SAR) and 18 different works that worked with the same data sets. SAR was able to outperform all works.

2.3 Recommender Systems

2.3.1 Objectives and Common Problems

Recommender systems aim to provide both the user and business owner with a better experience and performance. From the user perspective, recommenders shorten the time needed to find suitable items, thus improving user satisfaction/loyalty toward the platform. From a business owner's perspective, this gives insights into what users like without putting effort into marketing. Other benefits of using recommenders are listed below.

1. Increased product/content consumption: Recommender engines reduce marketing efforts by targeting users with products that might interest them, thus boosting sales and engagement. So when users discover new relevant content before even searching for it, they are more likely to return to the platform.

2. Customer satisfaction: Reducing the path to content/products means less cognitive effort for the user in order to find what they need. Increasing customer loyalty to the brand means increasing the probability of them returning to your platform. This improves customer retention and improves click-through rate (CTR).

3. Personalized experience: By providing personalized content, users feel more familiar/attached to the brand since it reacts to their interests so the number of visits goes up while churning drops. Customization doesn't only affect the platform the user is using but can also include personalized emails and notifications.

4. Relevance: Recommending relevant or interesting items to users is the most crucial goal of a recommender system. Users are more likely to interact with the content they deem relevant or interesting to them. Although relevance is important, other metrics that aren't well-known have a significant impact on the quality of recommender systems.

5. Novelty: Recommending content the user has not seen in the past can be helpful especially if the user shows interest. However, unlike novelty, serendipity is different since recommendations are genuinely surprising to the user, rather than simply something they did not know about before.

6. Diversity: Recommending a list of items with different types can achieve better results than recommending a list of similar items.

Since recommender systems are basically data filtering tools, they rely heavily on lots and lots of data. Data can be obtained from either users, content, or other sources (context). Modern websites provide users with multiple ways of expressing their feedback about the platform, content, or just their experience using the service.

Implicit and Explicit feedback A typical method of providing feedback is in the form of a five-star rating system, like or dislike buttons, or even numerical rating values (1-10). These rating methods that capture users' preferences are explicit ways used by companies. The disadvantage is that users tend to avoid the burden of explicitly stating their preferences. To overcome the shortage of explicit ratings, platforms tend to collect users' behavior in multiple ways (purchase data, watching/reading time, user mouse, and scrolling activity in addition to browsing history). This is called implicit feedback. The advantage of implicit feedback is that users can trigger a lot of actions when using a service. This generates a lot of data that can be significant in some cases but shows a major inconvenience, which is not having a ground truth (we cannot know for sure if a user really likes a product after seeing it multiple times). Another approach, called hybrid feedback leverages explicit ratings whenever possible, otherwise uses implicit feedback.

When ratings are not an option, businesses can use content descriptors/features to provide relevant recommendations. Content descriptors can vary from textual (summary of a video, song lyrics, movie subtitles, titles, etc.), to numerical descriptions (year, number of views, duration, etc.) as well as other information (time of visit, weather data, location data, etc.).

To build an effective recommender system, many challenges need to be overcome [79]. Some of them are detailed below.

Data sparsity and cold start problem To power up recommenders, we usually need either rating from users towards items or item descriptors. Ratings can be hard to obtain as most of the time it's sparse (lacks feedback). This can make the recommendation process difficult since there are not enough data to find patterns.

The cold start problem might concern either the items or users. A user cold start problem occurs when a new user or a group of users enters the platform. Since this is a new user, we do not know anything yet (no ratings are provided) which means we cannot find similar users from which to draw insights. The item cold start problem occurs when novel items are added to the catalog and there are not any users who rate them.

Scalability Generally, a Collaborative-filtering (CF) algorithm has a worst-case complexity of O(MN) where M is the number of users and N is the number of items. The CF system also must react to ratings in real time. As the number of users and items increases, the time and memory requirements also increase, and the CF system suffers from scalability issues.

Trust, Privacy, and Security A user should be able to trust that his CF provider does not sell his ratings, preferences and personal information to a third party and a CF provider has to make sure that it detects malicious users deliberately lowering or increasing an item's ratings to his advantage. Obviously, there is a need to maintain a balance between privacy and personalization.

2.3.2 Recommendation Approaches

Recommender systems mainly fall into two groups, collaborative filtering, and content-based. The main difference between these two approaches is the type of data used. Collaborative filtering algorithms employ user usage data (ratings, navigation history, etc.). Content-based algorithms use item characteristics or features which can be (metadata, textual descriptors, price, etc.) in order to recommend similar items. A combination of both methods is possible, and it's called Hybrid Recommenders. As you can see in Figure.2.23, diverse types of recommender systems are illustrated. In the following section, we will discuss each type in depth and see what their benefits and drawbacks are.

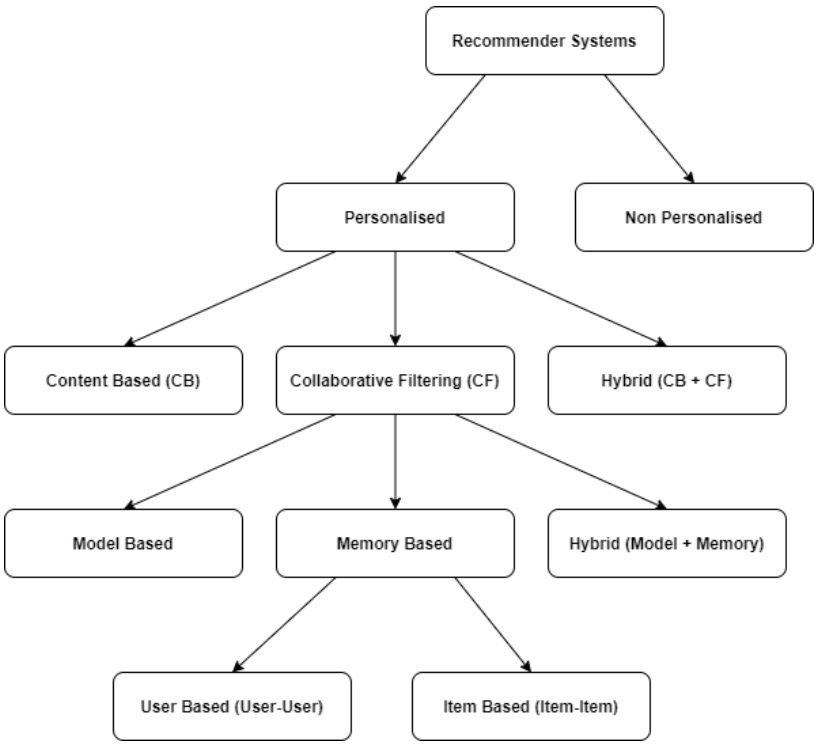

Figure 2.23 Types of Recommender Systems.

2.3.2.1 Content-based Filtering (CBF)

As mentioned earlier, content-based recommender systems focus on extracting knowledge from items to provide a way to describe them. Knowledge can be manifested in keywords, attributes, or a set of values [44]. This approach is useful for new items or items that don't have enough ratings.

Content-based filtering leverages the user profile (items seen, rated, liked in the past) to recommend related items (similar attributes) [107]. This approach follows the assumption that if a user likes an item, they will also like items with similar characteristics. Thus, does not account for a change in preference, opinion, or taste over time.

Using this approach comes with a challenge, which is finding the right attributes to use that gives the best description for an item. Extracting relevant information about the content can be challenging as there are multiple attributes that can or cannot contribute to identifying a specific item from a catalog. For example, news articles can have multiple attributes (date of the article, genre, country, text size, title, tags, etc.). Getting good descriptions of an item is not easy because the quality of descriptions can vary their types (textual vs non-textual) as well.

Content-based approaches use 3 components to function properly.

- Content profile: Creates content profile or descriptors.

- User profile: Creates a user profile, which can be a list of recently viewed items or liked items.
- Similar items: Find items that share the same attributes as the items found in a user's profile.

Content-based methods have some advantages, and they are listed below:

- Can work with new items that aren't yet rated because other items with similar characteristics might have been already rated.
- New users with no history can receive recommendations based on the similarity of items (using their descriptors).
- Content-based approaches can recommend items for users with specific tastes.

However, Content-based methods do have several disadvantages:

- Content descriptors: It can be difficult to find relevant information about a product.
- Over-specialization: Content-based recommenders won't suggest items that a user didn't show interest in. So results can be obvious since they rely on keywords on content. This reduces the diversity and serendipity of recommended items.

2.3.2.2 Collaborative-based Filtering (CF)

Collaborative filtering leverages users' activities to make recommendations. This method relies on ratings (explicit/implicit) provided by users to predict the likelihood of a user's liking an item [44].

Collaborative filtering assumes that if users liked the same things previously, the situation in the future won't change. This is why one of the methods of this type of recommender tries to group users or items in the same group based on a pattern of ratings.

The main challenge found in collaborative filtering is the sparsity of the ratings. Since users tend to avoid explicit ratings, many techniques try to leverage implicit ratings to better recommend items. There are two methods used in collaborative filtering. We explain each one in more detail in the following section.

Memory Based Approaches Memory-based or neighborhood-based collaborative filtering tries to find either users or items that are similar to other users/items [3]. Based on similar users or related items we can predict a rating score a user would give to a certain item. The similarity between items or users is calculated using multiple metrics, such as cosine similarity, Jaccard, and Pearson correlation.

- Jaccard Similarity: Used for comparing the similarity and diversity of sample sets. It is defined as the size of the intersection divided by the size of the union of the sample sets.
- Cosine similarity: Measures the angle between vectors. If the angle is 0°, then these vectors are similar and if the angle is 180°, then they are highly dissimilar.
- Pearson Similarity (Centered-Cosine similarity): It is the ratio between the covariance of two variables and the product of their standard deviations.

Memory Based Recommender Systems can be further divided into two groups.

User-to-User In this approach, we are looking for users similar to the user in question. Based on their rating patterns. Similar users are called neighbors, and their average ratings are used to predict how would the active user rate a specific item. The recommender system then can suggest items that similar users liked and the user in question didn't rate it yet. An example of User-to-User filtering is shown in Figure.2.24.

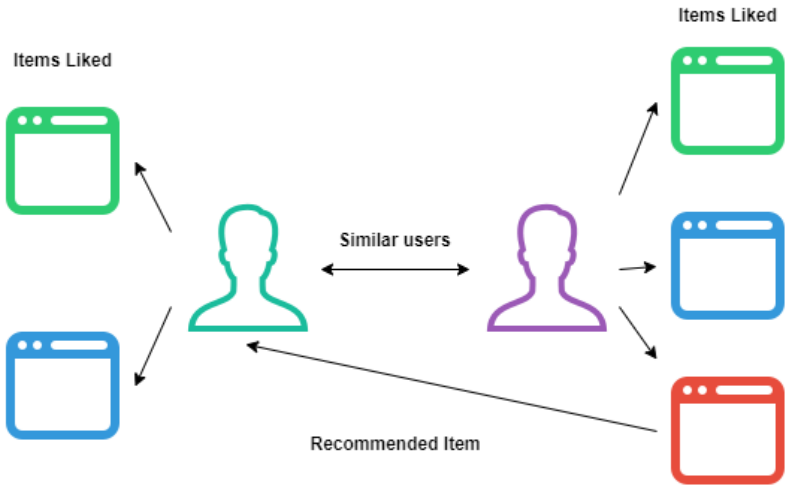

Figure 2.24 Example of User-Based Filtering. Adapted from [102]

Item-to-Item This method aims to predict the rating a user will give an item he did not yet see. This is done by finding a list of similar items rated by the same user and using them to predict the rating. A drawback of this method is that there tends to be a lower diversity in the recommendations as opposed to user-based CF. An example of Item-to-Item filtering is shown in Figure.2.25.

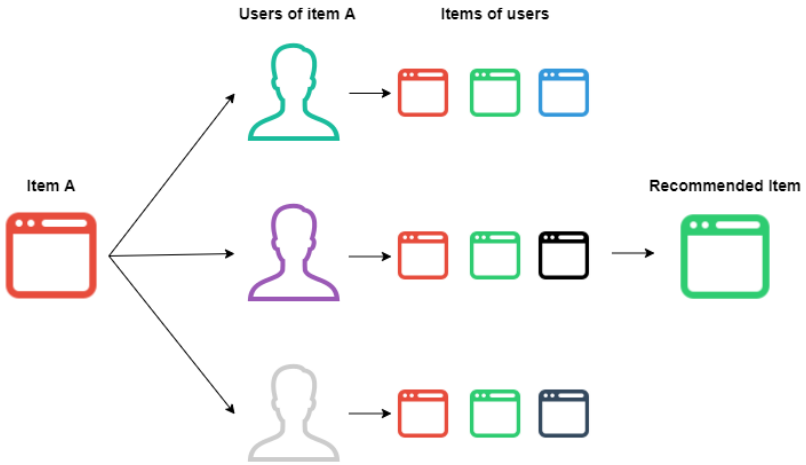

Figure 2.25 Example of Item-Based Filtering. Adapted from [102]

A general rule of thumb for deciding what approach to use is if you have more users than items in your platform the item-to-item approach works better, otherwise (number of items is more than users) user-to-user is the way to go [93]. As we have seen, the pipeline of memory-based filtering can either use user-based filtering to look at similar users or item-based filtering, to look at similar items. We list the advantages and drawbacks of using this method.

Advantages of Memory-based algorithms:

- Easy to implement and to add updated data.
- Obvious/Easy to explain recommendations.
- Doesn't require content metadata/descriptors. Only Ratings.

Disadvantages of Memory-based algorithms:

- Require Ratings (Can't handle well sparse rating matrices).
- Time and memory requirements scale significantly with the number of users and ratings.
- Cannot recommend for new users and items (Cold start Problem).

Model Based Approaches Machine learning and data mining methods are used to make predictions about a user's rating [3]. This predictive model can be a subject of the optimization process in case of multiple parameters are present. Memory Based Filtering requires the user-item dataset to make predictions, and Model-Based Filtering requires a fraction of that data thanks to dimensionality reduction techniques. One common approach for model-based CF is Matrix factorization (e.g., Singular Value Decomposition-SVD) which was used in the Netflix Challenge. Compared to Memory Based CF, the Model-based CF technique addresses the shortcomings of memory-based CF algorithms such as scalability, sparsity, and performance but at the cost of model building which can be expensive.

Moving on, collaborative filtering has some drawbacks. These issues include:

- Number of ratings: When a new item appears, the system can't recommend it because it doesn't have any ratings for the item. It will take some time to get enough ratings for the system to figure out what groups of users should be recommended the item.
- Sparsity With huge product bases, it's difficult to make sure that enough people explore all the options available. If some item hasn't been rated by a lot of people, the system won't have data on which to base the predictions.
- Gray sheep: To recommend items, the system has to group people with overlapping interests (finding neighbors). Many users will fall into these groups and enjoy the recommendations, but if some users do not consistently agree or disagree with some group, they will not be given high-quality recommendations.

As we have seen, collaborative filtering does not rely on content metadata or domain knowledge to provide recommendations. You only need the ratings and the interactions between items. In case we have both interactions and a collection of features and meta-data, we can use both in what is called a hybrid recommender system.

2.3.2.3 Hybrid Recommenders

It is noteworthy that each different type of recommender system uses a different type of data, whether its ratings for CF methods or features for content-based methods. While each method has its strengths and weaknesses, combining multiple methods can achieve much better results while benefiting from all the advantages each method presents. Hybrid approaches can be implemented in several ways: by making content-based and collaborative-based predictions separately and then combining them;

by adding content-based capabilities to a collaborative-based approach (and vice versa); or by unifying the approaches into one model. Hybridization presents an opportunity to overcome common problems such as cold start and the sparsity problem. There are generally three categories of hybrid recommender systems [74, 19]:

Monolithic hybridization design: A single algorithm that integrates multiple approaches by preprocessing and combining several knowledge sources. Figure.2.26 illustrates an example of monolithic hybridization.

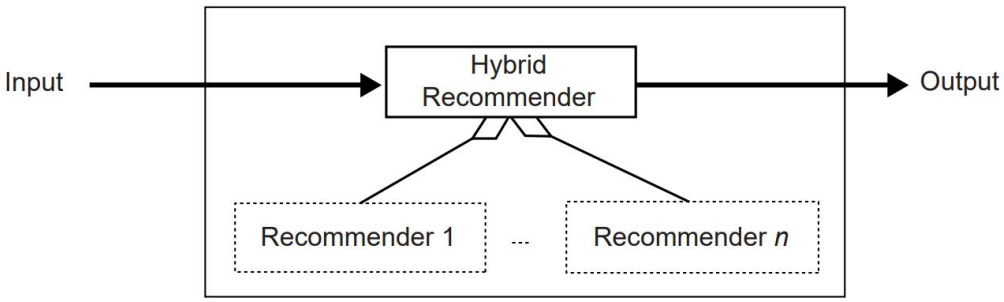

Figure 2.26 Monolithic hybridization design [74].

Parallelized hybridization design: This method operates independently, each approach produces its lists of recommendations which are later combined into a final set of solutions. Figure.2.27 illustrates an example of parallelized hybridization.

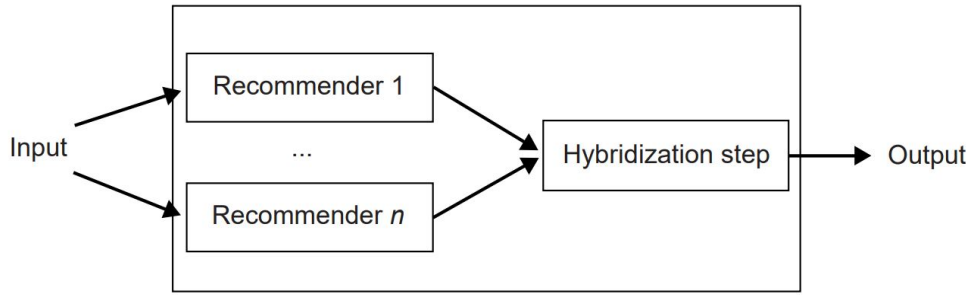

Figure 2.27 Parallelized hybridization design [74].

Pipelined hybridization design: In this approach, the output of one engine becomes part of the input data of the next engine. This is shown in Figure.2.28.

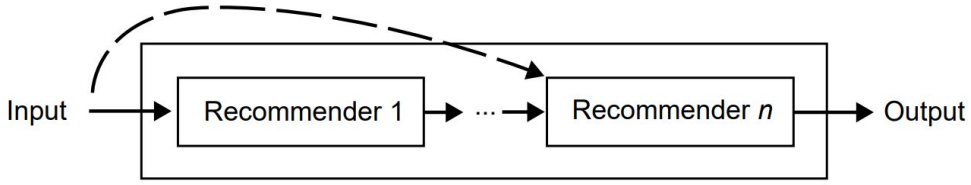

Figure 2.28 Pipelined hybridization design [74].

Recent approaches in hybridization are the use of deep learning to combine multiple types of recommendations (collaborative filtering, content-based, etc.). In Hybrid Deep Learning algorithms, users and items are modeled using both embeds that are learned using the collaborative filtering approach, and content-based features. Once embedding and features are computed, the recommendations can also be served in real time. A drawback of using deep learning for recommendations is that they need extensive hyper-parameter optimization to achieve better results than common techniques.

2.3.3 Evaluation Methods

Measuring the effectiveness of a recommendation algorithm is crucial when designing the evaluation part. While collaborative filtering might need unique metrics since it's somewhat different from content-based methods that rely more on text classification techniques. But before getting into what metric to use and when we need to further explain the types of recommender system evaluations. Recommender systems can be evaluated using either online or offline methods [61].

2.3.3.1 On-Line Evaluation

In online systems (e.g., e-commerce, MOOC, etc.) the user interactions with the presented recommendations are measured to test the effectiveness of the system. Interactions can be the clicks on recommended items, buy actions or watch time of recommended items, etc. This approach is sometimes less susceptible to bias because real users are naturally using the system. The main disadvantage is that such systems cannot be realistically deployed unless many users are already enrolled. Therefore, it is hard to use this method during the startup phase. One very known method used in the online evaluation is A/B testing which measures the direct impact of the recommender system on the user. The basic idea of these methods is to compare two algorithms as follows:

1. Segment the users into two groups A and B.

2. Use one algorithm for group A and another algorithm for group B while keeping all other conditions (e.g., the selection process of users) across the two groups as similar as possible.

3. At the end of the process, compare the conversion rate (or another payoff metric) of the two groups.

Another drawback that can drown here, is that online evaluation can only be assessed on some use cases that are related to the same domain or using the same data, thus can't be generalized to every use case. This can lead to mediocre performance in a different variety of settings. For this reason, offline methods are used when evaluating recommender systems.

2.3.3.2 Off-Line Evaluation

This method leverages historical datasets (e.g., ratings) to provide measures such as accuracy. Predictive measures address the subject of how close ratings of recommender systems are to the user ratings. They are a good choice for non-binary tasks. Mean Absolute Error (MAE) and Root Mean Squared Error (RMSE) is the most popular and easy-to-interpret predictive metrics.

Mean absolute error (MAE) is the average difference between the value predicted by the recommender $\hat{r}_{ui}$ and the actual value given by the user r_{ui} and $\hat{R}$ being the number of ratings. This is done by subtracting the predicted rating and actual rating for each user and taking the mean of all the errors to calculate MAE. One characteristic of this metric is if there are outliers or large error terms, it will weigh those equally with the other predictions.

$$MAE = \frac{1}{\left|\hat{R}\right|} \sum_{\hat{r}_{ui} \in \hat{R}} |r_{ui} - \hat{r}_{ui}| \tag{2.5}$$

Root Means Squared Error (RMSE) What this metric does essentially finds the difference between a predicted rating and a real rating. Having a lower error value means our model is predicting ratings similar to what the user gave. RMSE is a well-known metric and is used widely in the recommender systems research field. One characteristic of this metric is its vulnerability to being affected by outliers or bad predictions.

$$RMSE = \sqrt{\frac{1}{\left|\hat{R}\right|} \sum_{\hat{r}_{ui} \in \hat{R}} (r_{ui} - \hat{r}_{ui})^2} \tag{2.6}$$

Precision & Recall Classification metrics identify the degree of relevance or non-relevance of recommended items to the user [44]. The task here is adapted from the information retrieval field where we use the same principles shown in Table.2.3.

	Recommended	Not Recommended
Consumed	True Positive (TP)	False Negative (FN)
Not Consumed	False Positive (FP)	True Negative (TN)

Table 2.3 Confusion matrix of recommendation results [44].

- True positive (TP)—The item recommended and consumed by the user.
- False positive (FP)—The item was recommended but the user didn't consume it.
- False negative (FN)—The recommender didn't include the item in a recommendation and the user consumed it.
- True negative (TN)—The item wasn't recommended and the user didn't consume it.

We start with the first metric which is *Precision*. It determines the fraction of relevant items retrieved from the result. To calculate *Precision*, we use the following equation:

$$Precision = \frac{TP}{TP + FP} \tag{2.7}$$

The second metric is *Recall*, which determines the fraction of relevant items retrieved out of all relevant items. The following equation is responsible for calculating *Recall*.

$$Recall = \frac{TP}{TP + FN} \tag{2.8}$$

Note that when increasing *Precision*, *Recall* decreases as a result and vice versa. Metrics like the *F1* Metric attempts to combine *Precision* and *Recall* into a single value for comparison purposes [44].

Ranking metrics: The first item recommended is always the most important one, then the second is the second-most important, and so on. When evaluating, this can be considered thanks to ranking metrics. A variant of *Precision* exists which is *Precision@k* [76] where the k top elements are measured by taking the number of relevant items between the first k items.

One ranking metric is widely used which is the Discounted Cumulative Gain (DCG) [6]. This is a measure of ranking quality where highly relevant items are more useful when ranked first. Also, these metrics follow the assumption that highly relevant items are more useful than marginally relevant items, which are in turn more useful than non-relevant items.

$$nDCG_p = \frac{DCG_p}{IDCG_p} \tag{2.9}$$

With $IDCG_p = \sum_{i=1}^{|REL_p|} \frac{rel_i}{\log_2(i+1)}$ and $DCG_p = \sum_{i=1}^{p} \frac{2^{rel_i}-1}{\log_2(i+1)}$. Where REL_p is a list of top p relevant items (ordered by relevance). rel_i is the graded relevance of the result at position i.

Others metrics: Accuracy metrics aren't the metrics responsible for measuring recommenders' performance. People intrinsically enjoy variety and discovering things they have not seen before. In the real world, not every recommendation is based on similarity only. The main disadvantage of offline evaluations is that they do not measure the actual propensity of the user to react to the recommender system in the future. For example, the data might evolve, and the current predictions may not reflect the most appropriate predictions for the future. Moreover, measures such as accuracy do not capture the important features of recommendations, such as diversity and novelty, among other metrics listed below:

- Diversity: measures the spectrum of recommended items. Recommending items of diverse types are more diverse than items of the same type. Diversity can be based on users (how popular each item is, in the recommended list) or items (how different is each item in the recommended list).
- Novelty: measures how new, original, or unusual the recommendations are for the user. This sheds light on popular items being always at the top of recommendations while most items are in the long tail hidden by the immensity of a catalog.
- Serendipity: measures the surprise effect when finding things in your recommendations that you love but never knew you would.
- Coverage: measures how much available content we're covering when recommending items. Diversity leads to coverage because the better the diversity, the better the coverage.

2.3.4 Metaheuristic Approaches with Recommender Systems

In the past few years, recommender systems have been thoroughly applied to multiple domains such as e-commerce, movies and courses [41, 133, 42]. Approaches that exploit the combination of such recommendation algorithms are known as Hybrid Recommenders [19]. The advantage of this approach is that it limits the cons of each method used in the hybridization process all while inheriting their advantages. Standalone algorithms (CF or CBF) do not perform well when evaluated in terms of accuracy, novelty, and diversity. However, when combined together using one of the mentioned hybrid techniques, we're expecting improved and diversified results. Other approaches exploit the Pareto efficiency concept to combine such recommendation algorithms in a way that a particular objective is maximized without significantly hurting the other objectives. Recommender systems and the use of metaheuristics have been the focus of many academic researchers [20, 134, 132, 50]. We summarize these works in Table.2.4.

Most works, as shown in Table.2.4, combine collaborative filtering approaches (user-based, item-based, and model-based) with evolutionary algorithms such as Genetic Algorithms (GAs) in single and multi-objective modes. The most commonly handled objectives are accuracy, diversity, and novelty, with authors attempting to either obtain a solution with equal weights for each objective or a solution that follows user-specified weights.

A few works have mentioned the constraints discovered in the problem, and most of the time it is a single hard constraint that limits the recommended list (solution) to a fixed length. Because most works deal with evolutionary algorithms, we gathered information on the initialization strategies used and the genetic operators (both crossover CX and mutation MX).

In terms of the initialization phase, most works chose a random approach in which initial solutions are generated at random from the data, whereas others chose a different path in which existing recommendation approaches are used to generate solutions for users that are later optimized. One-point and two-point crossover operations are the most common for crossover operators CX, and similar findings are shown for mutation operators MX, which use one-point, random, and swap operators, sometimes with slight modifications to include repairing functions.

Finally, most works used traditional classification evaluation metrics such as (Precision, Accuracy, and Recall) with the Intra-List Similarity metric (ILS) [128] for diversity measurement. The Movielens [60] dominates the recommendation engine evaluation datasets. Research papers frequently select one of several available sizes (100K, 1M, 10M, or 20M movie ratings). We expand on these works in the following paragraph.

Authors	Rec. Approach	Opt. Approach	Objectives	Constraints	Initialization	Operators	Evaluation	Dataset
Alhijawi et al. [4]	CF	GA	3	2	Random	One-point (CX), Uniform, One-Point (MX)	MAE, Recall, Precision, and F1	HetRec 2011 and Movielens 100K
Rahul Katarya [81]	CF	ABC	1	-	Random	Not mentioned	MAE, Recall, Precision, and Accuracy	Movielens 100K
Da Silva et al. [28]	CF	GA	1	1	Random	Not mentioned	RMSE	Movielens 1M
Gasmi et al. [47]	CF	GA	1	1	Random	Two-point (CX), One-point (MX)	MEA and F1	Movielens 100K
Neysiani et al. [118]	ARM	GA	2	-	Random	One-point (CX), Random (MX)	Recall, Precision, Accuracy and F1	Movielens 1M
Chai et al. [20]	CF	MOIA	2	-	SVD	One-point (CX), One-point (MX)	Precision, Novelty and Diversity	Movielens 1M and Donation Dash
Wang et al. [132]	CF	NSGA-II	2	1	Random	One-point (CX), Swap (MX)	Accuracy and Diversity	Movielens 100K
Irfan et al. [70]	CF	NSGA-II	2	-	Random	Order Crossover (CX), Swap (MX)	Precision, Recall, and F1	Gowalla
Xie et al. [136]	Reinforcement Learning	-	-	-	Random	-	HIT@K, MAP and AUC (Offline), CTR and DW (Online).	LMOR (WeChat)
Zuo et al. [146]	CF	NSGA-II	3	-	-	Uniform (CX), Random (MX)	Accuracy, Novelty, Coverage and Hypervolume	Movielens 1M,
Ribeiro et al. [110]	CF	-	3	-	-	Two-point (CX), Uniform (MX)	Precision, Recall, Popilarity, Diversity and Novelty	MovieLens 1M, LastFM
Fortes et al. [45]	CF and CBF	NSGA-II, PSO	3	-	-	-	Accuracy, Novelty, and Diversity	Jester, Movielens 1M, Bookcrossing
Wang et al. [134]	CF	MOEA/D	2	-	Random	One-point (CX), One-point (MX)	Accuracy, Novelty, and Diversity	Movielens 100K, Jester

Table 2.4 Overview of related works.

(1) Alhijawi et al.,[4] applied Genetic Algorithms to collaborative filtering in order to recommend lists of items by combining both semantic relevancy and ratings. Their main contribution was finding a solution to cold-start and sparsity problems by considering the individual in the population as a potential recommendation list. In older work [5], the authors used the same approach but with different genetic operators (one point crossover/ one point mutation) and the same testing datasets.

(2) Rahul Katarya [81] proposed the use of an Artificial Bee Colony (ABC) to find the optimum value of center points for the KMeans algorithm. KMeans is used in a collaborative filtering algorithm to recommend movies to users. Their methodology consists of clustering users into groups based on their rating similarity.

(3) Da Silva et al.,[28] used a genetic algorithm to find the best-performing collaborative filtering approach. The goal is to find the lowest *RMSE* score. This was accomplished in two stages: first, recommender systems (CF) generate lists for the target user, and these lists are then fed into Genetic Algorithms that try to find the best combination of items to suggest. The authors proposed a possible improvement to this approach, which includes using item descriptors (content-based approach) or popularity to improve their relevancy.

(4) Gasmi et al.,[47] applied Genetic Algorithms to find users having similar characteristics to a target user, combined with a collaborative filtering approach to better predict users' ratings. Their approach is based on

contextual information gathered from user features (gender, age group, occupational category, and rating behavior). Authors indicated that choosing the right contextual feature weight is as hard as choosing the right content descriptors for items.

(5) Association Rules Mining (ARM) and Genetic Algorithms were combined in the work of Neysiani et al., [118]. The authors were successful in providing high-accuracy results by optimizing both support and confidence (ARM parameters). Their method was compared to the MOPSO (Multi-Objective Particle Swarm Optimization) algorithm. Despite producing better results than MOPSO, using ARM produced slow results because each rule must be checked in the database, which is impractical in real-world scenarios.

(6) Another multi-objective approach proposed by Chai et al.,[20]. In their work, both collaborative filtering (model-based) and multi-objective immune algorithm (MOIA) are combined to improve recommendation diversity without loss of accuracy. To generate initial recommendations, the authors used singular value decomposition (SVD). One limiting factor in this work is the fact that SVD can perform poorly against bigger datasets with high sparsity.

(7) Wang et al.,[132] combined multiple collaborative filtering techniques (item-based, user-based, and model-based) with NSGA-II. Similar to previous works, the authors aim to optimize accuracy and diversity simultaneously. Authors only tested their approach on 5 users from the Movielens dataset [60].

(8) Drifting away from movie recommendation, Irfan et al.,[70] used the NSGA-II algorithm and collaborative filtering methods to optimize two objectives (venue preference and location proximity). As suggested by the authors, adding more contextual features such as check-in time, user profiles, and interests could be a possible improvement in relevance.

(9) Xie et al.,[136] integrated a personalized approximate Pareto-efficient recommendation on the WeChat Top Stories section for millions of users. Their approach used reinforcement learning to find objective weights for the target user using list representation. Five metrics were used to evaluate models: click-through rate CRT, dwell-time scores DW for online evaluation and hit rate $HIT@K$, mean average precision MAP, and Area under the curve AUC for offline evaluation.

(10) Fortes et al.,[45] also adopted a similar technique, relying on user preferences concerning objectives weights during both the decision-making and optimization phases. Their approach aims to optimize accuracy, novelty, and diversity.

(11) The use of multiple recommendation engines is also developed in the work of Ribeiro et al.,[110], where a Pareto-efficient recommendation approach optimizes the weights of associated engines to provide items that are accurate, novel, and diverse.

(12) Zuo et al.,[146] proposed a multi-objective recommendation model to balance accuracy and diversity. Their approach relies on the NSGA-II algorithm. To reduce computing on each individual user, the authors adapted a grouping strategy that uses the KMeans algorithm to split users into several small clusters. These user groups receive similar recommendations.

(13) Another work by Wang et al. [134] used a Multi-objective Evolutionary Algorithm Based on Decomposition (MOEA/D) to recommend movies and jokes. Using the Movielens and jester dataset, their work successfully optimizes four different objectives (precision, recall, diversity, and novelty). The main recommendation approach used in item-to-item collaborative filtering.

(14) The work of Jugovac et al. [77] collects the tendencies of users, based on their past behavior, to provide a personalized recommendations list that adheres to the defined goals. Using a greedy re-ranking technique to match items with user profiles.

2.3.4.1 Positioning and motivations

Our work addresses different aspects that are missing or under-exploited in the aforementioned works:

- The use of multiple recommendation engines to initialize our solution population and provide more diversity.
- Working with real-world implicit ratings to train our model.
- We propose a customized mutation operator to improve the diversity of recommended items.
- Performance comparison of various Multi-Objective Evolutionary Algorithms (MOEA).

- Optimizing five conflicting objectives in the context of a real-world problem.
- The use of parameter tuning to optimize algorithms depending on user behavior and selected objectives.
- Work is being integrated into a production-ready environment.

2.4 Conclusion

We presented a brief overview of combinatorial problems and some of the resolution techniques used in the literature in this chapter. We also provided an introduction to the timetabling issue discussed in this thesis, as well as its differences from the literature. In the following Chapter, we model the entire problem and provide resolution steps. In addition to timetabling problems, a brief overview of recommender systems and their various types is discussed, with more information available in Chapter 4.

Chapter 3

Part I: MAPT - Mandarine Academy Professional Timetabling

Contents

In previous chapters, we defined the Professional Timetabling problem, and the problem found at Mandarine Academy. The existing timetabling tool at Mandarine Academy, DiLeap Logistic, proposed solutions that didn't respect company constraints. In this chapter, we present an approach that handles all constraints (hard and soft).

This task was done manually by a department of 30 persons, yet timetables were always invalid and took a lot of time to create (weeks) and a large amount of paper to provide a single solution. DiLeap Logistic provided an answer for rapid solution generation but lacked in the requirements area. Many solutions proposed by the tool didn't respect company constraints or objectives.

In this chapter, we present an approach that handles all constraints and objectives that depends on the client's needs. This chapter has been the subject of a publication in the IEEE Congress on Evolutionary Computation (CEC 2021) [55] and in the French Association for Operational Research and Decision Support (ROADEF-2021) [100].

The remainder of this chapter is organized as follows. Section 3.1 presents the problem definition, entities, and company guidelines. Objectives and hard/soft constraints are enumerated with their mathematical formulation.

In section 3.2 we present our proposed solution by using multi-objective genetic algorithms, for which we provide an encoding structure and two genetic operators (crossover and mutation). In Section 3.3, we propose a detailed experimental protocol that includes a description of the methods and data used, as well as details on determining the best settings for each algorithm and hardware configuration.

Results, given in Section 3.4, show the best parameters to use depending on the use case. Also, our approach yields better-diversified solutions and runs faster when compared to the manual approach.

The goal of developing a solution to such complex problems is to have that solution actively used by customers. As a result, it is critical that Mandarine Academy provides a production-ready environment for our model. We provide details of end-to-end course scheduling using DiLeap Logistic in Section 3.5.

3.1 Problem Description

We start this section by defining each entity and its relation to other actors in the planning process. This is followed by the found constraints and

objectives. We present the mathematical modeling in each section.

3.1.1 Entities

We identify the key entities that constitute our scheduling model. We start with entities that have the biggest impact on the issue we're modeling and detail the relationships with other entities. The entities hold the data needed to import, process, and provide a timetable. It is worth noting that we refer to a collection of Courses as Training or Events, and we use the symbol E to represent this. Figure.3.1 provides an overview of the relationship between key entities in the timetabling problem.

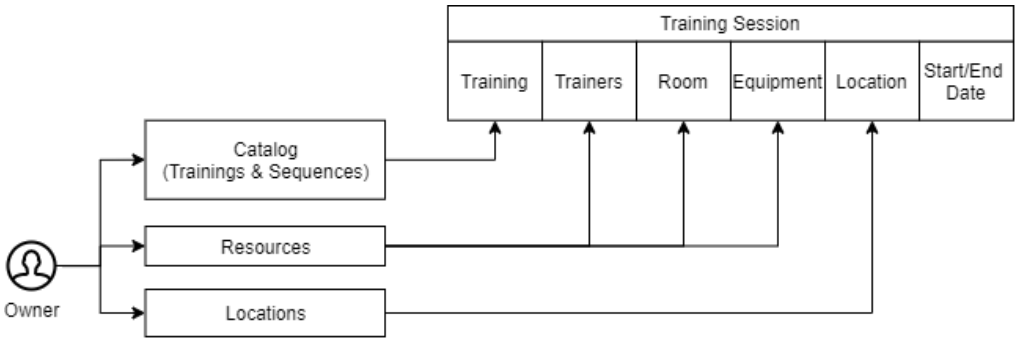

Figure 3.1 Relationship between entities involved in the scheduling process.

Session (Planning Day) The session is the central entity since most of the other entities exist only by and for it. It is the concrete and temporal organization of training and mobilizes the rooms, facilitators, materials, and trainees. In short, it is at the heart of the application. A session is a time unit of a day maximum on which will take place all or part of a previously selected training. This time constraint makes it possible to refine as well as possible all the material constraints which could arise from a training extended over several days (instructors changing from one day to another, room available only for part of the training, etc.). It can mobilize all the declared resources of the application. Therefore, in our problem definition, a session to schedule is characterized by:

- Owner: A session belongs to a proprietary.
- Course: A session may be based on a course.
- Sequence: A session may be based on the sequences included in a course.
- Trainers: A session can have one or more trainers (instructors).
- Rooms: A session can have one or more rooms.

- Equipment: A session can have one or more equipment.
- A date with starting/ending hours.
- A location and perimeter (specific business needs).

Proprietary (Owner) Since DiLeap Logistics is a shared platform, each proprietary has its different courses, resources, and parameters.

Skills An entity allowing the mastery of a discipline to be associated with an instructor (Trainer) or training (Course). It is, in fact, always associated with a mastery level.

- Associated with an instructor (Trainer), it indicates a skill that they possess (individual skills).
- Associated with a course or a sequence, it indicates what skill is required for these training courses to be conducted in good conditions once they are planned.

The levels of competence that are associated are completely configurable: they have a hierarchy that makes it possible to assess the mastery of the competence. The skills are also associated with a type of skill, which makes it easier to manage within different categories (e.g., a type of "office automation" skill for an Excel skill...).

Trainer (Tutor or Teacher) This is a user whose role is to provide the training sessions to which he/she is assigned: it is to him/her that the various planning display functions are mainly dedicated, which he can consult as a timetable. We describe Trainers (teachers) as a Set T which can be a natural person or a training institute. For this reason, an $overload\_factor_t \in \mathbb{Z}$ is assigned for each event e to indicate the number of simultaneous events a trainer t can attend. A trainer may be not available for certain days during the planning period because of different reasons (holidays, attendance at a seminar, etc.). These periods are known before the planning process. For each trainer $t \in T$, a type must be defined (Internal, external, etc..) for a trainer t by using $type_t \in TT$ where TT is a set of Trainer types. A trainer t can be assigned at minimum to a perimeter (company sector) $PT_t \subseteq P$ where P is a set of Perimeters. A trainer t can have a set of individual skills $SK_t \subseteq SK$ where SK is a set of Skills.

Room This entity represents a "physical" resource, which can be formally mobilized when planning a session. We denote a set of Rooms as R and a single room r. Because of their eminently concrete dimension, rooms have a whole set of attributes to provide:

A Room is an entity that represents a "physical" resource, which can be formally mobilized when planning a session. We denote a set of Rooms as R and a single room as r. Due to their concrete dimension, rooms have various attributes to provide:

1. Minimum and maximum capacities: denoted respectively as $min\_cap_r, max\_cap_r \in \mathbb{N}$, min_cap_r indicates the minimal number of allocated seats required to allocate the room r, while mmax_cap_r indicates the maximal number of allocated seats that room r can support.

2. Equipment: The equipment available in the room, will be considered during planning to offer the most relevant choices possible.

3. Overload factor: denoted as $overload\_factor_r \in \mathbb{Z}$, which indicates the number of simultaneous events a room r can hold.

4. Perimeter: $PR_r \subseteq P$ indicates the perimeter of room r (company sector).

5. Location: loc_r is defined to indicate the geographical position of room r.

6. Room type: Each room must have a type, which is useful for categorizing them and can be used for initial sorting during planning. The room type for room r is denoted by $type_r \in RT$, where RT is a set of room types (e.g., IT room, laboratory room, etc.).

7. Room configuration: A room r can come with a custom configuration (e.g., provides special seats for babies) denoted as $config_r \in RC$, where RC is a set of room configurations.

Finally, a room can have unavailability as well. These unavailabilities are known before the planning process.

Equipment A "physical" resource that can be formally mobilized when planning a session. Some trainings demand certain equipment for example a computer for an IT training. We define a set EQ of Equipment (e.g., computer, laser pointer, etc.). Equipment can be associated with a room (e.g., a table attached to the wall, which was not mentioned earlier but is relevant in this context) or can be associated directly with a training course or a sequence (both of which are not defined here but can be

understood as different types of training sessions). Each piece of equipment has a type, which is valuable for categorization and can be used for initial sorting during planning. We define for a device $eq \in EQ$ a type $type_{eq} \in EQT$ where EQT is a set of all device types. As for the room and the trainer, the equipment has unavailability, generated by the sessions to which it is linked or formally created by a user (maintenance, borrowing, etc.). This is also the reason why it has an overcapacity factor which allows the application not to necessarily declare it unavailable if we want to be able to associate it for a certain number of times over the same given period. Like the rest of the resources, a device eq has an overload factor defined as $overload\_factor_{eq} \in \mathbb{Z}$ that provides the maximum number of simultaneous events it can be used at.

Each of the following resources (Trainers, Devices, or Rooms) has a list of unavailabilities denoted as U_x, where x is a resource.

$$U_x = (d_1, st_1, et_1), .., (d_n, st_n, et_n)$$

where $d \in D$ and $st, et \in TS$. Here d, denotes a date, D is a set of dates, and TS is a set of timeslots with 15 minutes difference. Both st and et denote start/end time.

Catalog Each proprietary has its own catalog, which serves as a link between different underlying training entities. A catalog, as its name suggests, is a directory of training courses, organized according to a specific educational logic. The courses contained in a catalog are identified according to three types: the course, the learning path, and the sequence. We denote a set of Courses (Trainings or Events) with E.In this work, we will use the terms "training," "course," and "event" interchangeably to refer to the same concept. Figure.3.2 shows the relationship between a catalog, its courses, and sequences.

Course Course: includes data specific to its creation within a session at the time of planning. These data constitute the "Operating mode" of the course. We denote a set of Courses (Trainings or Events) with E. A course e has its own unique identifier along with the following details:

1. Course type: Each event or course $e \in E$ has one of two types (held in an actual room or online) denoted by $type_e \in ET$ where ET is a set of course types.

2. Perimeter: $PE_e \subseteq P$ indicates the company group (perimeter) to which this course belongs. P is a set of Perimeters.

3. A training can require certain types of resources:

(a) Trainers: We use $trainer\_type_e \in TT$ to indicate the required type of trainers. Where TT is a set of trainer types.

(b) Devices: $eqt_e \subseteq EQT$ indicates the preferred type of equipment for event e and EQT is a set of equipment types.

(c) Rooms: $room\_type_e \in RT$ or room configuration $room\_config_e \in RC$. RT is a set of room types and RC is a set of room configurations.

4. The planner provides a $session\_number_e$ that indicates the number of sessions to be planned for a specific event e. This means that a course can be scheduled multiple times within the event.

5. A $location_e \in L$ indicates where the event e must take place, L is a set of all locations. This is related to the rooms, as each location holds a specific set of rooms where events might be held.

6. A starting and ending hours st_e and $et_e \in TS$ are defined for event e as well as a duration in (days or hours) $duration_e \in \mathbb{R}$ is defined for each course e. Hours for each event, are known beforehand.

7. Trainings can indicate preferred dates or days. $SD_e \subseteq D$ (To avoid interruptions by weekends or holidays) or indicate imposed days $ID_e \subseteq D$ (ex: scheduled only on Wednesdays). We plan the events based on the preferred dates or days, and if there are imposed days, we schedule the events accordingly. The hours are also considered in the scheduling process.

8. A minimal and maximal seating capacity is defined per training $min\_cap_e \in \mathbb{Z}$, $max\_cap_e \in \mathbb{Z}$. If min_cap_e indicates the minimal number of registered attendees to consider training e in the scheduling progress. max_cap_e indicates the maximal number of registered attendees the training e can support. Knowing the minimal and maximal seating capacity helps in ensuring that the scheduled events have enough attendees to be considered valid (min_cap_e) and do not exceed the room's capacity (max_cap_e). This information is used during the scheduling process to avoid overbooking or underbooking events.

9. Trainings can require sometimes advanced trainers, a set of skilled trainers that are associated with event e is provided $ST_e \subseteq T$. In case the associated skilled trainers aren't available, trainers with specific individual skills can be considered. These specific skills of event e are denoted as $SK_e \subseteq SK$ where SK is a set of Skills.

10. Resources can be associated with an event e. We denote associated devices to event e by $AEQ_e \subseteq EQ$, for rooms we use $AR_e \subseteq R$ and trainers $AT_e \subseteq T$. These are different from skilled trainers SK.

11. Proprietary: The propriety to whom the course belongs.

A course might require certain skills. Thus, trainers who possess these skills can be proposed to run the training in good conditions.

Sequence A sequence is a digital representation of educational content that is provided by a trainer to learners. Unlike a course, a sequence is not a uniform training unit meant to be delivered in a single time block. Instead, it is a thematic sub-unit that must be associated with one or more courses in order to be scheduled. This means that the same sequence can be associated with multiple courses that may have different training objectives but all require the inclusion of this specific training module (e.g., an "Excel for Beginners" course and an "Office 365 Discovery" course can both have the sequence "Discover Excel Online"). This approach helps to prevent redundancy in declaring educational content and allows for more effective organization of various training modules within a course. However, it is not mandatory, and a course can be planned without having a specific sequence.

A sequence has a structure similar to a course, which enables it to declare a range of information that can be used for future scheduling of the course to which it belongs. These internal procedures allow for further refinement of what could have been stated earlier in the parent course. A sequence has its own unique identifier along with the same details found in a course's "Operating mode" except for a perimeter. For each training $e \in E$, sequences can be associated and are indicated by $SEQ_e \subseteq SEQ$. Sequences are considered sub-events and are defined in set SEQ.

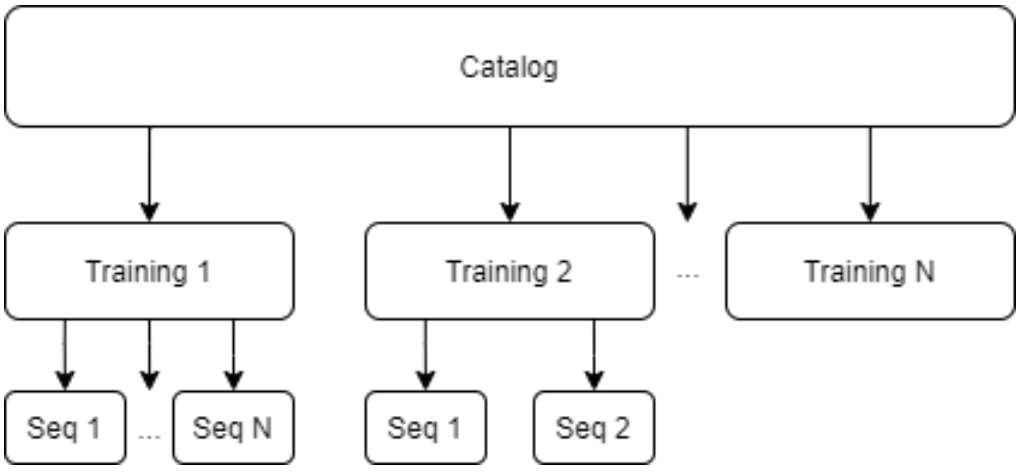

Figure 3.2 The Relation between Catalog, Trainings, and Sequences.

3.1.2 Constraints

In the following, we summarize the constraints which have been observed when scheduling the courses. Some constraints are standard for this kind of problem.

Generally, constraints are broken into 2 types “hard” and "soft" constraints. Hard constraints assure the operational feasibility of a schedule, there may be other constraints based on specific business rules and/or efficiency criteria that reflect qualitative aspects of a schedule and which are “softer” in nature. Solving a course scheduling problem involves finding at least an operationally feasible schedule. The following hard constraints are considered:

H1 Room type: Trainings may require a specific room type (IT, Lab, etc.). Events can only be placed in rooms having the same type $type_r$ as the event's e room type $room\_type_e$. This constraint is ignored if the course or sequence possesses a strongly associated room.

H2 Room location: Events must hold in rooms with the same location $location_r$ as the event location $location_e$. This constraint is ignored if the training is of the type 'web conference'.

H3 Room capacity: Only rooms with sufficient capacity can host trainings. $min\_cap_r \leq min\_cap_e$ and $max\_cap_r \leq max\_cap_e$. This constraint is ignored if the room is in strong association with the course.

H4 Type of equipment: Training may require specific types of equipment (computer, board, etc.). Device types must match those indicated by the event. This constraint is ignored if the material is in strong association with the course.

H5 Unavailability of resources (Room, Trainer, and Equipment): Resources may have unavailability dates. Thus, the timetable should not include unavailable resources during the scheduling phase.

H6 Specific skills: A trainer can't be chosen for training that requires either individual skills ($SK_t \notin SK_e$) or skilled trainers ($t \notin ST_e$). This constraint is ignored if a trainer is in strong association with the course.

H7 Planning horizon: Each training must be planned within the planning horizon indicated by the planner.

H8 Imposed days: Trainings may require specific days. For example, a three-day course has imposed days [Monday, Tuesday]. The training must take place: Monday, Tuesday, and Monday of next week.

Imposed days by the organizational staff for the event e are to be followed.

H9 Starting day: Trainings may require a starting day. The purpose of the starting day is to avoid splitting a course with a weekend between.

- Example 1: A four-day training might have imposed days [Monday] and a starting day [Monday] in this case, the course is scheduled Monday for four Weeks.
- Example 2: A three-day training might not have imposed days but can have a starting day [Monday]. In this case, the course is scheduled for Monday, Tuesday, and Wednesday of the same week.

H10 No resources (Trainer, rooms, and devices) can be placed in overlapping events more than their specified overload factor.

- Example: A session that begins at T1 = 8:05 and ends at T2 = 12:05 with a room R, trainer T or equipment cannot be used by another session. Resources remain blocked from T1 to T2.

H11 Fixed start and end times: Every course or sequence has a fixed start and end time. These hours must be respected and not changed.

H12 Sequences: Sequences are scheduled as a whole (following specific order) or not at all. For each event e having a non-empty set of sequences $SEQ_e \subseteq SEQ$ where $SEQ_e = \{seq_1, seq_2\}$ and $seq_2 \succ seq_1$ if and only if seq_2 is scheduled at after seq_1. This constraint ensures that the sequences are scheduled in the correct order.

H13 Perimeter: Only resources having the same perimeter as the event e can be selected. The perimeter refers to a specific company group.

H14 Session Number: The number of scheduled sessions for an event e cannot surpass $session\_number_e$ (indicated by the planner).

In this work, we consider a timetable that violates no hard constraints as valid. The following soft constraints are considered:

S1 Room configuration: Training may require a specific configuration for a room (school format for example).

S2 Type of trainer: A training course may require a specific type of trainer (Internal, Associate, etc.).

S3 Strong associations: A course or sequence may have associated resources (trainers, rooms, or equipment) that indicate that these resources are required for the training. To comply with the H5 constraint, resources, whether associated or not, must be available for selection. For example, if a physical training takes place in Bordeaux and has a strongly associated room in Paris, the room will not be considered due to the geographical constraint.

S4 Trainings with a duration longer than one day must not be interrupted by a weekend or long periods of days.

S5 A well-balanced event distribution is preferable. Each training should follow a list of preferable days to achieve balance. To measure how close a solution is to its balanced state we use the edit distance. It is expressed as the number of operations required to transform a string into another form [119]. The edit distance is a well-known concept in computer science and is used to measure the similarity between two strings. In this context, it is used to evaluate how close a solution is to its balanced state.

Assumptions

- Trainings are to be planned between 07:45 and 18:00.
- A day has a maximum duration of 7.8 hours. A course that has a duration equal to one day is equal to 7.8 hours. A half day is equal to 3.7 hours.
- The scheduling is done on a 5-day period (Monday, Tuesday, Wednesday, Thursday, and Friday).
- There is a lunch break between 12:00 and 14:00, but some trainings can start at those times.
- Training duration can range from half an hour to multiple days.
- The planning horizon can extend to multiple months.

The scheduling is done on a weekly basis, considering only weekdays (Monday to Friday). However, the planning for these trainings can be done for several months in advance. This means that while the trainings are scheduled within a 5-day period, the overall planning can cover a longer time frame, such as multiple months.

3.1.3 Objectives

There are several criteria to measure the quality of a schedule. Unlike most school timetabling problems, for instance, not all courses or sequences offered by the planner must be performed. An obvious objective is to maximize the number of courses as much as possible. Since the application has multiple owners, some desire efficient utilization of trainers and a fair distribution of workload among them.

Example: Marie and Paul have the same skills for the same courses, but Paul teached 150 courses, and Marie only 3 courses. There is no fair distribution in this case. To offer better flexibility, each planner can specify which objectives to be considered before launching the scheduling process. We are specifically looking to assign courses to days (not necessarily hours), trainers, and rooms, and potentially decide whether or not to schedule a particular course. The problem consists of several objectives, which can be selected by the platform users according to their preferences. The objectives are as follows:

(O1) Objective 1: Maximize the number of planned events:

$$\max f(x) = \sum_{i=1}^{n} Nbs_{e_i}$$

where Nbs_e is the total number of successfully scheduled sessions of event i and n is the number of events. The variable i is used as an index to represent each event in the summation. The objective function aims to maximize the total number of successfully scheduled sessions across all events. The sum is taken over all events, from event 1 to event n, and Nbs_{e_i} represents the number of successfully scheduled sessions for each specific event i.

(O2) Objective 2: Minimize the number of soft constraint violations.

$$\min g(x) = \sum_{i=1}^{n} (NbConfn_{e_i} + NbTrtn_{e_i} + NbARn_{e_i} + NbDistn_{e_i} + NbBdn_e)$$

where n is the number of events, $NbConfn_{e_i}$ is the total number of unique allocated rooms not having the same configuration as required by event $room\_type_e$ normalized by the total number of allocated rooms for event e. $Nbtrtn_{e_i}$ represents the total number of unique allocated trainers not having the same type as required by event $trainer\_type_e$, normalized by the total number of allocated trainers for event e. $NbARn_{e_i}$ presents the total number of unique allocated resources (Trainers, rooms, and equipment) that are not strongly associated with event e, where allocated resources $\notin AT_e, AR_e, AEQ_e$, normalized by the total number of allocated resources for event e. $NbDistn_{e_i}$ holds the total number of days (distance) between each instance in each session of event e, normalized by the total number of

instances in each session of event e. $NbBdn_e$ is the edit distance between event e schedule and the preferable well-balanced schedule e, normalized by the total number of events.

(O3) Objective 3: Minimize the number of external trainers. This indirectly maximizes the internal trainers' workload.

$$\min h(x) = \sum_{e}^{n} NbuT_e$$

where $NbuT_e$ is the total number of unique external trainers in all scheduled sessions of event e, n is the number of events.

(O4) Objective 4: Standard deviation will measure the workload deviations between internal trainers. To achieve a balanced workload distribution we seek to minimize the standard deviation. i denotes the index of event e,

$$\min \sigma = \sqrt{\frac{\sum_{i=1}^{n}(workload_{t_i} - \overline{\text{mw}})}{n-1}}$$

$workload_t$ is the total teaching hours of trainer t having type $type_t = Internal \in TT$, n is the total number of unique internal trainers and $\overline{\text{mw}}$ is the mean workload of all internal trainers and can be described as follows: $\overline{\text{mw}} = \frac{\sum_{j=1}^{n}(workload_{t_j})}{n}$.). Notice here that **O3** conflicts with this objective.

(O5) Objective 5: Minimize the number of classrooms. The goal of this objective is to reduce used classrooms, thus maximizing their utilization and indirectly reducing their overall cost (rent and maintenance). Incorporating explicit costs for each room could provide a more accurate representation, but handling costs with multiple objectives and constraints (soft or hard) can be challenging.

$$\min k(x) = \sum_{i=1}^{n} NbuR_{e_i}$$

. Where $NbuR_e$ is the total number of unique rooms in scheduled sessions of event e, n is the number of events.

A solution to this problem is a schedule that satisfies the selected objectives while adhering to the given constraints. Users can choose any combination of the objectives, and the platform will present the best solutions based on pseudo weights techniques. These solutions can be non-dominated solutions or Pareto fronts, depending on the user's preferences and the selected objectives.

3.2 Genetic Modeling

The initial step in GAs is encoding or solution representation in which a description of a solution's structure is defined. We detail in the next section the choice of encoding along with an overview of how we create our initial population.

3.2.1 Solution Encoding

When using a genetic algorithm, it is critical to select the best way to represent the solution. In general, encoding methods can be direct or indirect [22].

Starting with indirect methods, one example is the well-known binary encoding. Binary representations are commonly used due to their simplicity and memory-efficient design. However, in the real world, much effort goes into developing proper genotype representations because it is difficult to represent a solution with a binary string when dealing with complex problems and multiple constraints. This leads to the second encoding method, direct encoding. In the direct encoding method, the entire solution to a given problem is used as a chromosome. However, working with direct representation necessitates the development of a decoding scheme as well as a mechanism for monitoring the constraints and feasibility of solutions. It is important to note that classic genetic operators (crossover and mutation) can still be applied to direct encoding, but they may require adaptation to suit the specific problem representation.

We opted for a direct representation for the $MAPT$ (Mandarine Academy Professional Timetabling) problem.

Each solution has a set of event sessions ES belonging to their respective training.
Each session can have one or many session instances si where selected resources, dates, hours, sequences, and a location are specified.

- Event session instance $si_e = (R_e, d_e, st_e, et_e, l_e, seq_e)$, where:
  - R_e denotes the allocated resources set (Trainers, rooms, and devices).
  - d_e denotes the allocated day to event session instance si_e.
  - st_e and et_e denotes the allocated start and end times to si_e, which are imposed.
  - l_e denotes the allocated location to event session instance si_e.
  - seq_e denotes the specific sequence of the session instance.

- Event session $ES_e = \{si_{1e}, si_{2e}, \ldots, si_{ne}\}$, where n depend on e.
- Event session set $S_e = \{ES_{1e}, ES_{2e}, \ldots, ES_{me}\}$, where m depend on e.
- $sol = \{S_{e1}, S_{e2}, , \ldots, S_{ew}, \}$, where w depend on selected courses to be planned. S_{e1} is the set of sessions for the first course $e1$.

3.2.2 Initializing Population

As mentioned in [120], the initial population of genetic algorithms can be produced through a random, greedy, or heuristic approach.

We opted for a constructive heuristic with hard constraints incorporated in its design, this approach leads to a valid initial population that doesn't need extra fixing.

Algorithm 1 shows the proposed heuristic that uses a priority scheme at the start to assign courses E_{input}, with both 1) higher duration and 2) extra restrictions on days (imposed or starting days) a higher priority. Starting from the highest priority to the lowest, the heuristic selects every session of each training and a) assigns a time slot and b) resources that respect the defined constraints. The diversity between produced solutions is ensured by selecting different resources (if available) for different sessions of the same event. Note that in case of unavailable resources/time slots, the heuristic adapts by using an auxiliary memory to optimize the search process. In some cases, some events will be discarded as there aren't unlimited resources at each location.

Algorithm 1: Initializing Population: Constructive heuristic for MAPT problem initial solutions generation.

Data: E_{input} Courses Sorted by defined priority scheme.
Result: Sol (List of planned courses).

```
while still have courses to plan do
    current_session ← 0  E_dur ← Event duration (in days)
     E_Seqcnt ← Event sequence count  while
     current_session ≤ session_nbr do
        while (E_dur ≥ 0) ∨ (E_Seqcnt ≥ 0) do
            day ← valid days  resources ← valid resources
             session_instance ← day, resources
        session ← session_instance
         current_session ← current_session + 1  E_dur ← E_dur − 1
         E_Seqcnt ← E_Seqcnt − 1
    course ← course + session  Sol ← Sol + course
    return Sol
```

3.2.3 Genetic Operators

We list the different operators used to conduct our research, starting with the selection operator which is described in detail in Chapter 2, Section 2.1.1.4. There are two main categories of solution selection methods for genetic algorithms [30]:

- Deterministic methods: The selection of the fittest individuals in a population for recombination. This method can potentially lead to a local optimum, as it focuses on the best solutions for the current population.
- Stochastic methods: Select individuals randomly, without considering their fitness. This approach helps to maintain diversity in the population and avoid premature convergence to a local optimum.

The roulette wheel selection is a middle way since it creates a discrete probability distribution from which we identify the chromosomes. This explains why it is the most common selection method used in genetic algorithms. The roulette wheel selection is chosen for our experiments. For crossover and mutation, we provide more details in the following paragraph.

Crossover Operators like Partially Mapped Crossover (PMX), Cycle Crossover (CX) and Order Crossover (OX) are frequently used for scheduling problems [23]. To apply the aforementioned operators to the $MAPT$ problem, appropriate operators need to be designed. Since every solution must satisfy a range of hard constraints to be feasible, the classical operators will need to undergo modifications to produce valid solutions. We propose the use of a crossover operator called $MAPT_{co}$ and adapted from [16]. This operator was specially crafted for this problem, the pseudo-code is shown in Algorithm 2. In addition to the $MAPT_{co}$ crossover operator, we have also experimented with other crossover operators such as PMX, CX, and OX. The results of these experiments will be presented later in the thesis, where we compare the performance of these operators in solving the $MAPT$ problem. Both $MAPT_{co}$ and PMX include company business logic, concerning their adaptivity with the new proposed encoding.

Algorithm 2: Pseudo code of the $MAPT_{co}$ crossover operator, adapted from [16].

Data: P_1, P_2 (Parents)
Result: O_1, O_2 (Off-springs)
$N \leftarrow$ total number of events.
$E_1 \leftarrow$ randomly select a number where $E_1 > 1$ and $E_1 \leq N$.
$E_2 \leftarrow N - E_1$.
$O_1 \leftarrow$ randomly select E_1 number of events from P_1 and randomly select E_2 number of events from P_2 not already selected from P_1.
$O_2 \leftarrow$ randomly select E_2 number of events from P_2 and randomly select E_1 number of events from P_1 not already selected from P_2.
return O_1, O_2

The $MAPT_{co}$ crossover operator is designed to ensure that the off-springs inherit some characteristics from both parents while maintaining the feasibility of the solutions. The random selection of events from both parents helps in exploring the search space effectively.

A possible conflict with constraints can happen for the crossover operators. The following example demonstrates. Parent 1 and Parent 2 don't share the same number of planned sessions per event. In the case of a crossover operation, a child solution might end up having certain sessions with unavailable resources since the order was changed. The child's solution, in this case, is not valid. To overcome this, a repairing function is implemented where the time slot/resources are changed for the invalid sessions. In case when this doesn't work (no availability) the infeasible session is discarded.

Mutation Like crossover, the mutation is applied following a defined probability p_m. Classical operators (Bit Flip, Swap, and Random mutation) are commonly found in the literature.

A custom operator named $MAPT_{mo}$ is proposed to be compared with a classical operator. $MAPT_{mo}$ was adapted from [16] and the classical Random mutation. However, $MAPT_{mo}$ has business logic incorporated in its design that permits it to work on a specific catalog's properties (resources and time slots).

The operator selects an event session randomly and then decides whether to generate a random valid (hard constraints) time slot or resources (new rooms, equipment, or trainers). Two granularity of mutation can be used with $MAPT_{mo}$ that impacts the solution's state:

- Small (Default): By selecting a small granularity, changes inside a solution are applied at the Event session instance si_e level. this permits changes to resources, time slots, location, etc.

- Large: By selecting a large granularity, changes inside a solution are applied at the Event Session ES_e as a whole. This permits huge changes to the overall schedule for a particular event e.

3.3 Experimental Design

In this section, we focus on the experimental protocol implementation used at Mandarine Academy to tackle the timetabling problem. Figure.3.3 provides an overview of steps considered in the experimental design. In the first step, we describe the historical data used in the experiments. We discuss the performance of using the constructive heuristic to generate initial solutions for each test size. In order to provide a fair comparison, we also explain how we tune the parameters of Multi-Objective Genetic Algorithms (MOGAs) with irace both 3 and 5 objectives. The last subsection evaluates and interprets the findings of selected metaheuristics on several runs.

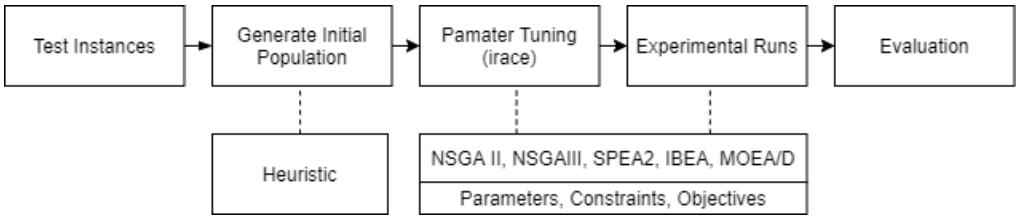

Figure 3.3 Overview of the experimental protocol used in MAPT problem.

3.3.1 Dataset

As we explained before, since this problem is different from academic timetabling, evaluating our approach using academic data sets such as (ITC 2007 or ITC 2019) won't be relevant, since our objectives incorporate business requirements and aren't purely academic as the focus here is organizational more than academic.

Rather than using synthetic data, we use historical data for experiments. Mandarine Academy and its partners in DiLeap Logistic supplied us with some historical data (2019-2020) in order to test our approach. Problem instances were created using a total of 152 different training. Table.3.1 offers an overview of our test instances. Note here that there are no duplicate courses inside each test instance.

In total, 3 different problem sizes (small, medium, and large) are provided. For each problem size, the number of sessions, training, and planning window is increased. This was implemented to simulate how the algorithms will behave when having a larger number of events to satisfy. Note that the duration is expressed in days and each size has 20 different test instances.

This real-world data is published online under the title Mandarine Academy Professional Timetabling Dataset (MAPTD)[54]. The data is aimed at benchmarking timetabling solvers whether it is in education or other professional sectors. There are two kinds of records in MAPTD:

- Input files that are used as testing the approach. There are 3 groups (Small, Medium, and Large) each with 20 different instances, totaling 60 test sets, to simulate different real-world scenarios.
- Second records are training files that include the information of each entity involved in the scheduling process. Such as courses, teachers and rooms availabilities, locations, etc. Note that both records have no redundant values.

Table 3.1 Characteristic of test instances used in our experiments.

Size	Small	Medium	Large
Sessions	100	500	1050
Trainings	20	50	70
Duration	36.34	97.24	135
AvailableDays	60	152	244
Trainers	269	269	269
Rooms	230	230	230
Devices	14	14	14
TestInstances	20	20	20

3.3.2 Parameter Tuning

In order to fairly compare the performance of each algorithm, the i-race package [94] with its elitist-iterated racing approach is used to find the best-performing parameters. Elite configurations are returned based on their average best hypervolume indicator (HV) across different test instances. More information about i-race is provided in Section 2.1.4.

The initialization of i-race consists of importing all the considered parameters with their possible values. Because the algorithms in our experiments are population-based, common parameters are found, such as:

- Population size (Pop): Number of generated solutions at each generation.
- Crossover operator (C_o)
- Crossover probability (C_p): Probability for parent solutions to recombine.

- Mutation operator (M_o)
- Mutation probability (M_p): Probability for a child solution to be mutated.

For each parameter we list the possible values it can have, this is shown in Table.3.2. Parameter values for the population size were chosen based on the number of events in our test instances. The classical PMX crossover operator will be compared to our proposed operator $MAPT_{co}$. For mutation operators, the Swap mutation is compared to our proposed operator $MAPT_{mo}$. A probability of 1.0 for both genetic operations means that all offspring are produced by crossover or mutation. A probability of 0.0 indicates that we are mimicking the older generation. We simulate genetic algorithms against various cases, providing low to high probability values for mutation and crossover, to see how the performance varies.

Table 3.2 Parameters settings considered for tuning phase.

Parameters	*Values*
POP	50, 75, 100, 125, 150
CX	PMX, $MAPT_{co}$
CP	0.0, 0.1, 0.2, 0.3, 0.5, 0.8, 1.0
MX	$SWAP$, $MAPT_{mo}$
MP	0.1, 0.3, 0.5, 0.7, 0.8, 0.9, 1.0

The termination criteria chosen for the final experiments are 4, 8, and 12 hours of computing time. This is the time duration for algorithms to execute until termination. The choice of such duration is aligned with the company's interest in running these heavy calculations overnight, to provide a solution, the next day for complex problems with urgency.

3.3.3 Computational Environment

We conduct the experiments on a private cloud cluster. We use 5 virtual machines (VM) each composed of 24 cores of 1 GHz and 100 GB RAM each. We installed SLURM [117] an open-source, fault-tolerant, scalable cluster management, and job scheduling system. It is responsible for allocating access to resources (compute nodes) to users, monitoring jobs, and finally, it is arbitrating resources by managing a queue of pending work.

3.4 Experimental Results

For this task, we will be using jMetalPy [10] a python framework for single/multi-objective optimization that offers both parallel computing capabilities and easy-to-use visualization tools. The framework can be used to develop custom evolutionary algorithms/operators and define new problems.

The initial population used in both the tuning phase as well in experiments was produced by the proposed constructive heuristic. The average number of planned sessions per test size is 100 (100%) for small instances, 498 (98%) for medium instances, and 980 (93.3%) for large instances.

3.4.1 Parameter Tuning

Elites' configurations provided by i-race for each algorithm (NSGAII, NSGAIII, SPEA2, MOEA/D, and IBEA) are computed using the hypervolume indicator. These algorithms are adapted from jMetalpy to our problem, and the earlier demonstrated algorithm is only used to generate valid initial solutions using a constructive heuristic. A non-parametric test is applied to results by using Spearman's rank correlation coefficient, Kendall's concordance coefficient, variance, and Friedman's test (rank).

Table.3.3 shows the best configuration found for each algorithm on the small problem instances using 3 objectives (O1, O2, and O4). The chosen stopping criterion for our experiments is the elapsed computing time, which was set to a maximum of 4 hours per instance. Note that MOEA/D has additional parameters which are the neighbor size (NS), neighborhood selection probability (NSP), and max number of selected solutions (MSS). The values for these parameters respectively are (NS: 33, NSP: 0.8, and MSS: 16). Table.3.3 shows the best configuration found for each algorithm on the small problem instances considering all objectives (O1, O2, O3, O4, and O5) simultaneously to find non-dominated solutions. MOEA/D's additional parameter values are (NS: 15, NSP: 0.7 and MSS: 8).

Across all returned configurations $MAPT_{mo}$ is selected, this confirms that it surpasses the classical Swap mutation for this specific problem. After all, $MAPT_{mo}$ uses business logic to change solutions. However, this isn't the case for $MAPT_{co}$ our proposed crossover operator that wasn't included in the elite configurations, instead, the classical PMX operator was chosen.

For the following sub-section, each algorithm will use its best configuration against all problem sizes (small, medium, and large) and with 4, 8, and 12 hours of computing time. This will test the configurations found to see how the performance evolves when we let algorithms run for longer periods against more complex problems.

Table 3.3 I-race results (Small instances) using 4 hours as a stopping criterion for 3 objectives (O1, O2, and O4) and 5 objectives.

	POP		CX		CP		MX		MP	
OBJ	3	5	3	5	3	5	3	5	3	5
$NSGAII$	125	125	PMX	PMX	0.7	0.7	$MAPT_{mo}$	$MAPT_{mo}$	0.2	0.2
$NSGAIII$	150	75	PMX	PMX	0.8	0.8	$MAPT_{mo}$	$MAPT_{mo}$	0.1	0.3
$SPEA2$	150	150	PMX	PMX	0.8	0.7	$MAPT_{mo}$	$MAPT_{mo}$	0.3	0.4
$MOEA/D$	125	125	PMX	PMX	0.9	0.8	$MAPT_{mo}$	$MAPT_{mo}$	0.3	0.4
$IBEA$	150	125	PMX	PMX	0.6	0.8	$MAPT_{mo}$	$MAPT_{mo}$	0.3	0.8

3.4.2 Small Dataset Results

We performed 30 independent runs for each algorithm using 3 objectives (O1, O2, and O4) and 5 objectives (O1, O2, O3, O4, and O5). We showcase the results for each problem size (small, medium, and large) with multiple stopping conditions (4, 8, and 12 hours of computing time). The main performance metrics used in our experiments are the Hypervolume (HV) [140] to maximize, the Generational Distance (GD) [127], Inverse Generational Distance (IGD) [127] and the Epsilon-Indicator (ϵ) [145] are to be minimized. For each metric, the **average best value** and the standard deviation are provided.

We are comparing the hypervolumes in the objective space to evaluate the performance of the algorithms. The Hypervolume (HV) metric is responsible for measuring the convergence and diversity of our solutions, making it a better quality estimator than other indicators. We used the Friedman Test to ensure the statistical relevance of the results. With a p-value of $p < 0.05$, results are compared using only the HV metric and not all the indicators.

Performance analysis using 3 Objectives Table.3.4 shows a performance comparison of all algorithms on small problem instances using 3 Objectives (O1, O2, and O4). The computing time mentioned in the following analysis is per instance, and the experiments were conducted on individual instances with the specified stopping criteria values.

Starting with 4 hours of computing time, better results are provided by IBEA followed by NSGA II, SPEA2, and NSGA III in terms of the HV metric. The difference between these algorithms is minimal. Taking into account the other performance indicators (GD, IGD, ϵ) we clearly see no clear advantage from any algorithm, as they all behave similarly with the exception of MOEA/D which has the worst results compared to the rest. After studying the previous findings, we decided to launch new experiments

using two additional stopping criteria values, 8 and 12 hours of computing time per instance. For 3 objectives and 4 hours of computing time, IBEA had the top spot for HV in second place we can see NSGAII and SPEA2 having similar HV scores with NSGAIII coming in 4th place with similar EP indicator.

When looking at the results of 5 objectives and 4 hours of computing time per instance found in Table.3.5, we can see IBEA still in the lead, with NSGAIII in second place followed by NSGAII and SPEA2. When averaging their HV scores, we find that IBEA and NSGAIII are our top 2 contenders. We decided to use these two methods for the remainder of our experiments.

Moving on, with stopping criteria of 8 hours of computing time per instance, better results are provided by IBEA in terms of HV, NSGA III is behind by a small difference. Considering other performance indicators (GD, IGD, ϵ), NSGA III has better GD and IGD compared to IBEA which was able to obtain a better ϵ aside from the hypervolume. For the 12 hours stopping condition, with a negligible difference, IBEA is still in the lead in terms of HV. For other performance metrics, NSGA III keeps having better results in terms of GD and IGD, while IBEA is holding its ground with better ϵ and HV.

Convergence analysis using 3 Objectives Observations taken from Fig. 3.4, show the evolution of the hypervolume indicator for each algorithm. Starting with 4 hours stopping criterion, most algorithms were able to converge faster, with IBEA and NSGA III in the lead. More computing time is needed as all algorithms are still converging. Notice that the time taken to improve the quality of solutions is slower and worse using MOEA/D which confirms its lower performance compared to the rest of the algorithms for this specific problem. Though both algorithms start with similar performance, IBEA was able to converge faster than NSGA III. From 8 to 12 hours of computing time, we can clearly see NSGA III still converging while IBEA is starting to slowly stabilize (stagnate).

Table 3.4 Performance comparison ($Mean_{\underline{standarddeviation}}$) using small instances and 3 Objectives for over 30 independent runs.

Small Test Instances (3 Objectives)					
$T(Hours)$	4				
$Algorithm$	$NSGAII$	$NSGAIII$	$SPEA2$	$MOEA/D$	$IBEA$
HV	$0.89_{\underline{0.026}}$	$0.88_{\underline{0.021}}$	$0.89_{\underline{0.020}}$	$0.73_{\underline{0.021}}$	$\mathbf{0.90}_{\underline{0.017}}$
GD	$\mathbf{1.51}_{\underline{0.019}}$	$1.52_{\underline{0.011}}$	$\mathbf{1.51}_{\underline{0.017}}$	$1.54_{\underline{0.017}}$	$1.55_{\underline{0.019}}$
IGD	$1.38_{\underline{0.041}}$	$1.41_{\underline{0.032}}$	$\mathbf{1.37}_{\underline{0.034}}$	$1.50_{\underline{0.036}}$	$1.47_{\underline{0.026}}$
ϵ	$-0.86_{\underline{0.018}}$	$-0.86_{\underline{.017}}$	$-0.86_{\underline{0.019}}$	$-0.86_{\underline{0.014}}$	$\mathbf{-0.87}_{\underline{0.023}}$
$T(Hours)$	8		12		
$Algorithm$	$NSGAIII$	$IBEA$	$NSGAIII$	$IBEA$	
HV	$0.94_{\underline{0.012}}$	$\mathbf{0.96}_{\underline{0.010}}$	$\mathbf{0.96}_{\underline{0.012}}$	$0.96_{\underline{0.009}}$	
GD	$\mathbf{1.53}_{\underline{0.020}}$	$1.67_{\underline{0.021}}$	$\mathbf{1.55}_{\underline{0.023}}$	$1.68_{\underline{0.014}}$	
IGD	$\mathbf{1.40}_{\underline{0.040}}$	$\mathit{1.58}_{\underline{0.042}}$	$\mathbf{1.39}_{\underline{0.035}}$	$1.61_{\underline{0.048}}$	
ϵ	$-0.90_{\underline{0.02}}$	$\mathbf{0.95}_{\underline{0.010}}$	$-0.93_{\underline{0.024}}$	$\mathbf{-0.95}_{\underline{0.011}}$	

Table 3.5 Performance comparison ($Mean_{\underline{standarddeviation}}$) using small instances and 5 Objectives for over 30 independent runs.

Small Test Instances (5 Objectives)					
$T(Hours)$	4				
$Algorithm$	$NSGAII$	$NSGAIII$	$SPEA2$	$MOEA/D$	$IBEA$
HV	$0.80_{\underline{0.026}}$	$0.83_{\underline{0.021}}$	$0.80_{\underline{0.020}}$	$0.61_{\underline{0.021}}$	$\mathbf{0.85}_{\underline{0.017}}$
HV (3 OBJ)	$0.87_{\underline{0.016}}$	$0.88_{\underline{0.021}}$	$0.87_{\underline{0.015}}$	$0.68_{\underline{0.037}}$	$\mathbf{0.91}_{\underline{0.014}}$
GD	$\mathbf{1.90}_{\underline{0.014}}$	$1.95_{\underline{0.016}}$	$1.91_{\underline{0.015}}$	$1.99_{\underline{0.034}}$	$2.00_{\underline{0.015}}$
IGD	$\mathbf{1.71}_{\underline{0.037}}$	$1.76_{\underline{0.042}}$	$\mathbf{1.71}_{\underline{0.040}}$	$1.96_{\underline{0.039}}$	$1.90_{\underline{0.007}}$
ϵ	$-0.83_{\underline{0.023}}$	$-0.85_{\underline{0.021}}$	$-0.83_{\underline{0.021}}$	$-0.76_{\underline{0.029}}$	$\mathbf{-0.88}_{\underline{0.019}}$
$T(Hours)$	8		12		
$Algorithm$	$NSGAIII$	$IBEA$	$NSGAIII$	$IBEA$	
HV	$0.89_{\underline{0.029}}$	$\mathbf{0.91}_{\underline{0.014}}$	$\mathbf{0.92}_{\underline{0.018}}$	$0.91_{\underline{0.016}}$	
HV (3 OBJ)	$0.93_{\underline{0.026}}$	$\mathbf{0.95}_{\underline{0.010}}$	$\mathbf{0.96}_{\underline{0.016}}$	$0.95_{\underline{0.012}}$	
GD	$\mathbf{2.01}_{\underline{0.039}}$	$2.15_{\underline{0.030}}$	$\mathbf{2.05}_{\underline{0.034}}$	$2.16_{\underline{0.020}}$	
IGD	$\mathbf{1.79}_{\underline{0.062}}$	$\mathit{2.04}_{\underline{0.077}}$	$\mathbf{1.84}_{\underline{0.068}}$	$2.07_{\underline{0.083}}$	
ϵ	$-0.91_{\underline{0.029}}$	$\mathbf{-0.95}_{\underline{0.009}}$	$-0.94_{\underline{0.014}}$	$\mathbf{-0.95}_{\underline{0.010}}$	

Performance analysis using 5 Objectives Table.3.5 shows a performance comparison of all algorithms on small problem instances using 5 Objectives.

Starting with 4 hours of computing time, better results are provided by IBEA followed by NSGA III in second place. SPEA2 and NSGA II have similar scores (third place). We aggregated the same 3 Objectives used

in the previous experiment (O1, O2, and O4) to see if aggregating yields better results than running experiments with only 3 Objectives. Results found in $HV(3OBJ)$ show a small difference when compared to those found in Table.3.4.

For other performance indicators (GD, IGD, ϵ) we clearly see no clear advantage from any algorithm, as they all behave similarly except for MOEA/D which has the worst results compared to the rest. NSGA II was able to hold a better GD across both experiments, similarly, SPEA 2 was able to maintain a better IGD even after switching to 5 Objectives. The clear winner by a small margin is IBEA for both 3 and 5 Objectives in the 4 hours time frame.

For 8 hours, NSGA III is behind IBEA by a small difference in terms of HV. The same results are present in HV for 3 Objectives aggregated from the total 5 Objectives. For the 12 hours experiments, better results are provided by NSGA III even when the difference isn't huge. This is also true for $HV(3OBJ)$. Considering other performance indicators (GD, IGD, ϵ) for 8 and 12 hours, NSGA III has better GD and IGD compared to IBEA which was able to only obtain a better ϵ.

Convergence analysis using 5 Objectives By observing the convergence graph found in Fig.3.5 for 5 objectives and Fig.3.6 for 3 out of 5 objectives using small problem instances, we notice at the 4 hours mark, that each algorithm is still converging with both IBEA and NSGA III at the lead. The gap between IBEA and NSGA III gets wider in the middle of the experiments (between 4-8 hours). Around 12 hours of computing time elapsed, we clearly see IBEA's performance stabilizing, while NSGA III is still improving, and it surpasses IBEA at the very end of the experiments.

Seems that 8 hours of computing time already gives a good HV, so the extra 4 hours might not be worthwhile after all. Now after seeing both algorithms tested under different computing times, we will be testing them with medium test instances to see if their performance will hold against more complicated problems. (e.g., more complex instances).

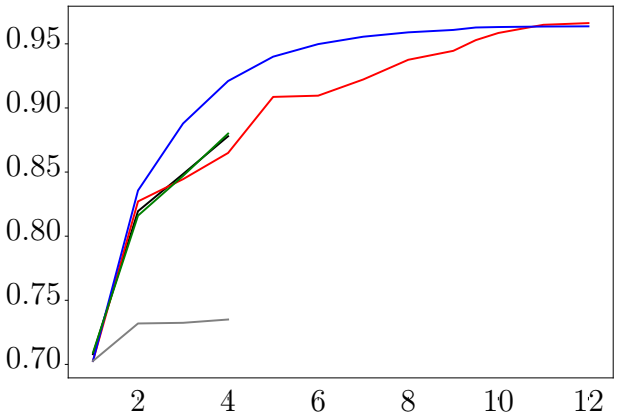

Figure 3.4 Evolution of the hypervolume indicator (Y-axis) over 12 hours of computing time (X-axis) using the Small problem instances (30 runs) and 3 Objectives with NSGAIII (Red), IBEA (Blue), NSGAII (Black), SPEA2 (Green) and MOEA/D (Grey).

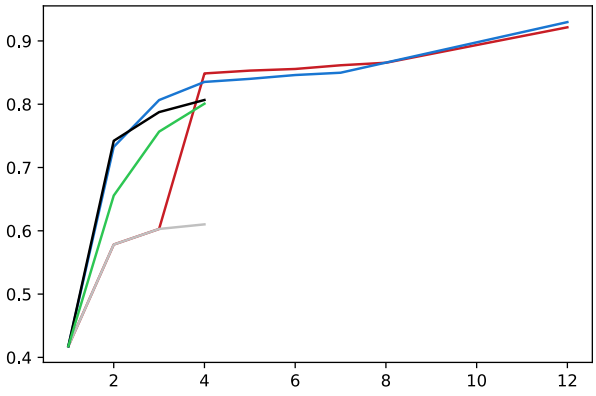

Figure 3.5 Evolution of the hypervolume indicator (Y-axis) over 12 hours of computing time (X-axis) using the Small problem instances (30 runs) and 5 Objectives with NSGAIII (Red), IBEA (Blue), NSGAII (Black), SPEA2 (Green) and MOEA/D (Grey).

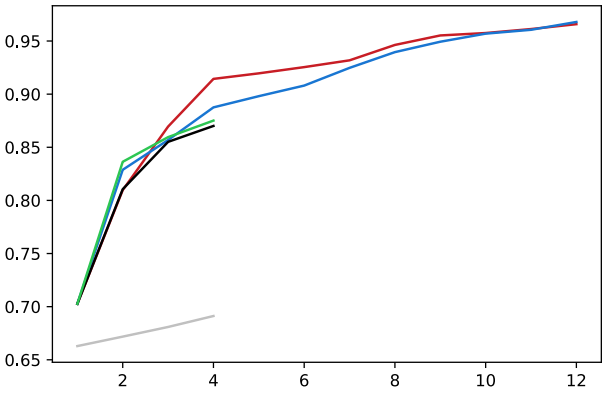

Figure 3.6 Evolution of the hypervolume indicator (Y-axis) over 12 hours of computing time (X-axis) using the Small problem instances (30 runs) and 3 out of 5 Objectives with NSGAIII (Red), IBEA (Blue), NSGAII (Black), SPEA2 (Green) and MOEA/D (Grey).

3.4.3 Medium Dataset Results

Performance analysis using 3 Objectives Table.3.6 shows a performance comparison of both NSGA III and IBEA on Medium problem instances and using 3 Objectives for a duration of 4, 8, and 12 hours.

Starting with 4 hours and despite the small difference, NSGA III was able to maintain a better HV. Even after adding more computing time (8 hours), performance from both algorithms didn't change much. This time, IBEA has slightly better performance than NSGA III. The final experiments (12 hours) show a very small performance increase that barely justifies the additional 4 hours of computing time. IBEA dominates medium instances experiments by having the best values for every metric (again, with a small difference).

At first glance, and when compared to Table.3.4 that uses Small problem instances, we clearly see that both algorithms struggle to achieve higher performance when faced with more complex problem instances. Now, this can be caused by multiple factors, such as both algorithms might need to undergo a tuning phase using the medium instances to find suitable configurations.

Convergence analysis using 3 Objectives Convergence graphs of IBEA and NSGA III found in Fig. 3.7 for 3 Objectives and a duration of 4, 8, and 12 hours, both algorithms are still converging even after adding extra time in between. While NSGA III kept the lead for the first 10 hours,

it was IBEA who surpasses the latter and achieves better performance in 12 hours duration.

Table 3.6 Performance comparison ($Mean_{\underline{standarddeviation}}$) using medium instances and 3 Objectives for over 30 independent runs.

Medium Test Instances (3 Objectives)						
$T(Hours)$	4		8		12	
Algorithm	$NSGAIII$	$IBEA$	$NSGAIII$	$IBEA$	$NSGAIII$	$IBEA$
HV	$\mathbf{0.68}_{\underline{0.003}}$	$0.67_{\underline{0.006}}$	$0.68_{\underline{0.004}}$	$\mathbf{0.70}_{\underline{0.010}}$	$0.70_{\underline{0.005}}$	$\mathbf{0.73}_{\underline{0.013}}$
GD	$\mathbf{1.50}_{\underline{0.0008}}$	$1.50_{\underline{0.001}}$	$1.50_{\underline{0.001}}$	$\mathbf{1.49}_{\underline{0.001}}$	$1.49_{\underline{0.007}}$	$\mathbf{1.48}_{\underline{0.004}}$
IGD	$1.42_{\underline{0.034}}$	$\mathbf{1.38}_{\underline{0.021}}$	$1.39_{\underline{0.021}}$	$\boldsymbol{1.34}_{\underline{0.016}}$	$1.37_{\underline{0.025}}$	$\mathbf{1.33}_{\underline{0.012}}$
ϵ	$-0.72_{\underline{0.012}}$	$\mathbf{-0.73}_{\underline{0.006}}$	$-0.73_{\underline{0.005}}$	$\mathbf{-0.74}_{\underline{0.007}}$	$\mathbf{-0.74}_{\underline{0.006}}$	$\mathbf{-0.74}_{\underline{0.008}}$

Table 3.7 Performance comparison ($Mean_{\underline{standarddeviation}}$) using medium instances and 5 Objectives for over 30 independent runs.

Medium Test Instances (5 Objectives)						
$T(Hours)$	4		8		12	
Algorithm	$NSGAIII$	$IBEA$	$NSGAIII$	$IBEA$	$NSGAIII$	$IBEA$
HV	$\mathbf{0.59}_{\underline{0.015}}$	$0.55_{\underline{0.015}}$	$0.63_{\underline{0.015}}$	$\mathbf{0.65}_{\underline{0.015}}$	$0.65_{\underline{0.016}}$	$\mathbf{0.68}_{\underline{0.010}}$
HV (3 OBJ)	$\mathbf{0.66}_{\underline{0.005}}$	$0.64_{\underline{0.002}}$	$\mathbf{0.70}_{\underline{0.010}}$	$0.69_{\underline{0.012}}$	$0.71_{\underline{0.018}}$	$\mathbf{0.72}_{\underline{0.011}}$
GD	$\mathbf{1.88}_{\underline{0.004}}$	$1.90_{\underline{0.002}}$	$\mathbf{1.87}_{\underline{0.004}}$	$1.90_{\underline{0.001}}$	$\mathbf{1.87}_{\underline{0.004}}$	$1.91_{\underline{0.002}}$
IGD	$\mathbf{1.80}_{\underline{0.020}}$	$1.83_{\underline{0.005}}$	$\mathbf{1.79}_{\underline{0.014}}$	$\boldsymbol{1.85}_{\underline{0.008}}$	$\mathbf{1.79}_{\underline{0.017}}$	$1.87_{\underline{0.006}}$
ϵ	$\mathbf{-0.70}_{\underline{0.007}}$	$-0.66_{\underline{0.010}}$	$-0.69_{\underline{0.009}}$	$\mathbf{-0.70}_{\underline{0.011}}$	$-0.69_{\underline{0.016}}$	$\mathbf{-0.70}_{\underline{0.013}}$

Performance analysis using 5 Objectives Table.3.7 shows a performance comparison of both NSGA III and IBEA on medium problem instances using 5 Objectives for a duration of 4, 8, and 12 hours.

Similar to results found using only 3 Objectives in Table.3.6, NSGA III is maintaining superiority over IBEA in the 4-hour range and has achieved the best results in every metric. Switching to 8 hours of computing time, a really small increase in HV is achieved by both algorithms. While IBEA was able to overthrow NSGA III when comparing the HV of all 5 Objectives, NSGA III has shown a slightly better HV when aggregating 3 Objectives from the overall 5 Objectives. For the remainder of the metrics, NSGA III has the best metrics in terms of GD, IGD, and ϵ. In the final experiments with 12 hours of computing time, we can see a slow increase in HV for both algorithms. NSGA III has a better GD and IGD, and IBEA has a slightly better ϵ and HV. Now, even when performance is not at the same level as what we have seen when using both algorithms on the small test instances, at least both algorithms are still improving.

Convergence analysis using 5 Objectives Hypervolume evolution graphs found in Fig. 3.8 and Fig. 3.9 for 5 and 3 out of 5 objectives respectively, shows how IBEA was able to keep improving with time, surpassing NSGA III at the end. We also notice that both algorithms showcase a similar behavior in the final half of the experiments where they slow their improvement rate dramatically. This also indicates that both algorithms need additional computing time as they are still converging, and thus perhaps better performance can be achieved.

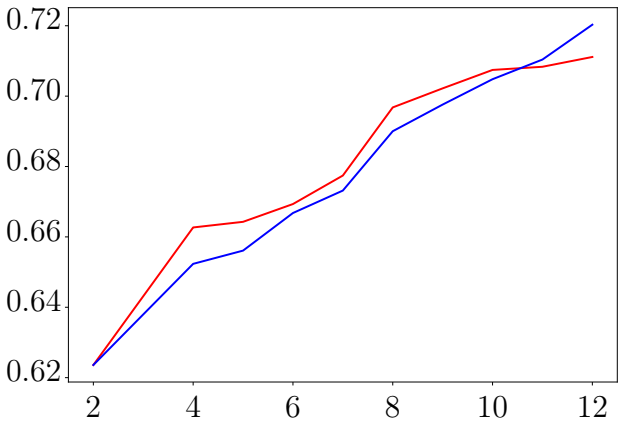

Figure 3.7 Evolution of the hypervolume indicator (Y-axis) over 12 hours of computing time (X-axis) using the medium problem instances (30 runs) and 3 Objectives, with NSGAIII (Red) and IBEA (Blue).

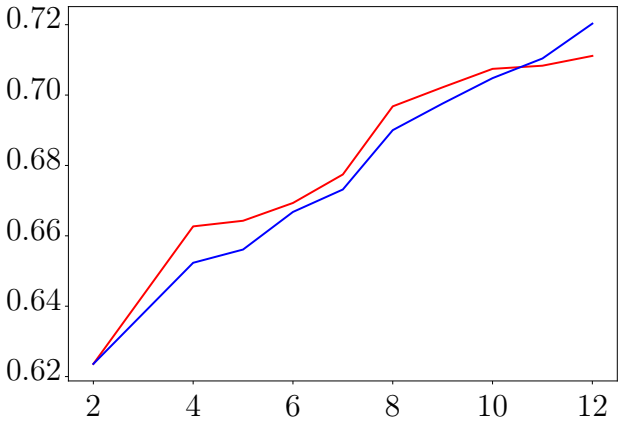

Figure 3.8 Evolution of the hypervolume indicator (Y-axis) over 12 hours of computing time (X-axis) using the medium problem instances (30 runs) and 5 Objectives, with NSGAIII (Red) and IBEA (Blue).

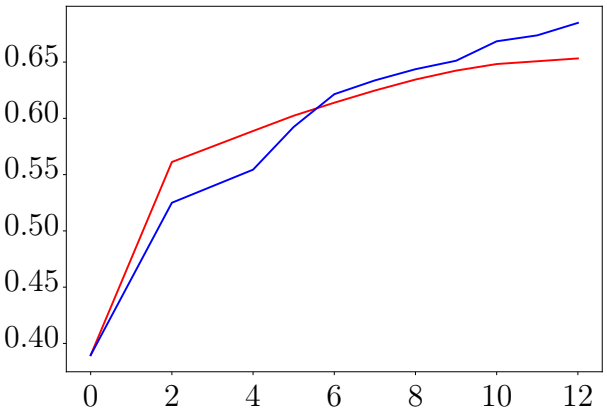

Figure 3.9 Evolution of the hypervolume indicator (Y-axis) over 12 hours of computing time (X-axis) using the medium problem instances (30 runs) and 3 out of 5 Objectives, with NSGAIII (Red) and IBEA (Blue).

3.4.4 Large Dataset Results

Performance analysis using 3 Objectives Table.3.8 shows the performance of IBEA and NSGA III using Large problem instances and 3 Objectives for a duration of 4, 8, 12, and 24 hours.

Similar performance behavior to those found in Table.3.6 with Medium instances. This might confirm that the performance of both algorithms is problem-size-dependent since both algorithms use their best configurations based on tests applied only to small instances.

On two separate occasions (4 and 12 hours) NSGA III and IBEA have similar HV scores. For the rest of the metrics, both algorithms either had the same values or in rare cases an edge over the other. For the 24 hours experiments, IBEA is a clear winner here in terms of HV and ϵ while NSGA III has better GD and IGD. Even though the difference is small, both algorithms continue to improve.

As seen before, both algorithms might need to undergo a tuning phase using large instances in order to find elite configurations that perform well.

Convergence analysis using 3 Objectives The convergence graph of IBEA and NSGA III for 4 hours stopping criterion wasn't provided as both algorithms didn't improve their initial HV for the whole allowed computing time, so the results found in Table.3.8 for 3 Objectives are the initial and final scores.

Fig 3.10 shows the HV evolution of IBEA and NSGA III. Both NSGA III and IBEA require additional computing time as they are both still converging. While NSGA III was trying to close the gap with IBEA, this was successful with around 12 hours of computing time, but after that, IBEA returns to the lead and surpasses NSGA III.

Table 3.8 Performance comparison ($\text{Mean}_{\underline{standarddeviation}}$) using large instances and 3 Objectives for over 30 independent runs.

Large Test Instances (3 Objectives)				
$T(Hours)$	4		8	
$Algorithm$	$NSGAIII$	$IBEA$	$NSGAIII$	$IBEA$
HV	$\mathbf{0.60}_{\underline{3.15e-05}}$	$\mathbf{0.60}_{\underline{2.75e-05}}$	$0.63_{\underline{0.03}}$	$\mathbf{0.64}_{\underline{0.009}}$
GD	$\mathbf{1.48}_{\underline{0.0001}}$	$\mathbf{1.48}_{\underline{3.26e-07}}$	$\mathbf{1.47}_{\underline{0.008}}$	$\mathbf{1.47}_{\underline{0.0009}}$
IGD	$\mathbf{1.47}_{\underline{6.66e-16}}$	$\mathbf{1.47}_{\underline{6.66e-16}}$	$1.42_{\underline{0.047}}$	$\mathbf{\mathit{1.37}}_{\underline{0.008}}$
ϵ	$\mathbf{-0.68}_{\underline{3.61e-05}}$	$\mathbf{-0.68}_{\underline{3.15e-05}}$	$-0.70_{\underline{0.021}}$	$\mathbf{-0.71}_{\underline{0.009}}$
$T(Hours)$	12		24	
$Algorithm$	$NSGAIII$	$IBEA$	$NSGAIII$	$IBEA$
HV	$\mathbf{0.66}_{\underline{0.007}}$	$\mathbf{0.66}_{\underline{0.009}}$	$0.68_{\underline{0.008}}$	$\mathbf{0.70}_{\underline{0.009}}$
GD	$\mathbf{1.46}_{\underline{0.002}}$	$1.47_{\underline{0.001}}$	$\mathbf{1.44}_{\underline{0.01}}$	$1.45_{\underline{0.002}}$
IGD	$1.38_{\underline{0.016}}$	$\mathbf{1.36}_{\underline{0.010}}$	$1.34_{\underline{0.01}}$	$\mathbf{1.33}_{\underline{0.01}}$
ϵ	$\mathbf{-0.72}_{\underline{0.006}}$	$\mathbf{-0.72}_{\underline{0.006}}$	$-0.72_{\underline{0.004}}$	$\mathbf{-0.73}_{\underline{0.007}}$

Table 3.9 Performance comparison ($Mean_{\underline{standarddeviation}}$) using large instances and 5 Objectives for over 30 independent runs.

Large Test Instances (5 Objectives)				
$T(Hours)$	4		8	
$Algorithm$	$NSGAIII$	$IBEA$	$NSGAIII$	$IBEA$
HV	$\mathbf{0.46}_{\underline{0.07}}$	$0.39_{\underline{1.89e-05}}$	$\mathbf{0.57}_{\underline{0.02}}$	$0.49_{\underline{0.08}}$
HV (3 OBJ)	$\mathbf{0.61}_{\underline{0.015}}$	$0.60_{\underline{2.91e-05}}$	$\mathbf{0.63}_{\underline{0.008}}$	$0.61_{\underline{0.011}}$
GD	$\mathbf{1.89}_{\underline{0.006}}$	$1.90_{\underline{7.33e-07}}$	$\mathbf{1.88}_{\underline{0.005}}$	$1.89_{\underline{0.001}}$
IGD	$\mathbf{1.86}_{\underline{0.036}}$	$1.89_{\underline{1.33e-15}}$	$\mathbf{1.81}_{\underline{0.019}}$	$\mathit{1.86}_{\underline{0.027}}$
ϵ	$\mathbf{-0.68}_{\underline{0.018}}$	$-0.66_{\underline{2.22e-16}}$	$\mathbf{-0.70}_{\underline{0.008}}$	$-0.68_{\underline{0.013}}$
$T(Hours)$	12		24	
$Algorithm$	$NSGAIII$	$IBEA$	$NSGAIII$	$IBEA$
HV	$\mathbf{0.59}_{\underline{0.01}}$	$\mathbf{0.59}_{\underline{0.02}}$	$0.64_{\underline{0.009}}$	$\mathbf{0.65}_{\underline{0.010}}$
HV (3 OBJ)	$0.65_{\underline{0.008}}$	$\mathbf{0.69}_{\underline{0.010}}$	$0.68_{\underline{0.009}}$	$\mathbf{0.69}_{\underline{0.009}}$
GD	$\mathbf{1.87}_{\underline{0.004}}$	$1.89_{\underline{0.002}}$	$\mathbf{1.88}_{\underline{0.005}}$	$1.90_{\underline{0.003}}$
IGD	$\mathbf{1.80}_{\underline{0.022}}$	$1.84_{\underline{0.008}}$	$\mathbf{1.81}_{\underline{0.019}}$	$1.87_{\underline{0.006}}$
ϵ	$\mathbf{-0.70}_{\underline{0.008}}$	$-0.69_{\underline{0.010}}$	$-0.690_{\underline{0.008}}$	$\mathbf{-0.693}_{\underline{0.010}}$

Performance analysis using 5 Objectives Table.3.9 shows a performance comparison of both NSGA III and IBEA on Large problem instances using 5 Objectives for a duration of 4, 8, 12, and 24 hours.

NSGA III was able to achieve the best results in terms of HV (for 4 and 8 hours) and similar results with IBEA on 12 hours of computing time. While NSGA III dominated HV when aggregating 3 Objectives, IBEA was able to achieve a higher HV when more computing time is available. For the rest of the performance metrics (GD, IGD, and ϵ) NSGA III held its first place in each experiment. For 24 hours, IBEA performs better than NSGA in terms of HV for 5 Objectives, HV for 3 Objectives, and ϵ.

Unfortunately, the use of medium and large instances showed that our selected elite configurations will not work optimally unless given a longer available computing time which is impractical.

Convergence analysis using 5 Objectives Hypervolume evolution figures found in Fig.3.11 and Fig.3.12 demonstrate both algorithms' performances. IBEA struggles to improve against NSGA III in 5 objectives experiments and it is improving at a slow pace compared to NSGA III. NSGA III performed way better than IBEA in terms of convergence rate until around 10 hours, then IBEA took the lead. When looking at only 3 Objectives (aggregated from 5 objectives), NSGA III had the lead for more than half the experiment time and kept close readings when was surpassed by IBEA.

Both experiments with 3 or 5 objectives showed exceptional performance when faced with small test problems, while this was not the case for medium and large instances, IBEA and NSGA III were able to offer diverse and improved solutions always. Further improvement is always possible and re-running tuning experiments with consideration for more complex problems might show different results from what we have seen already.

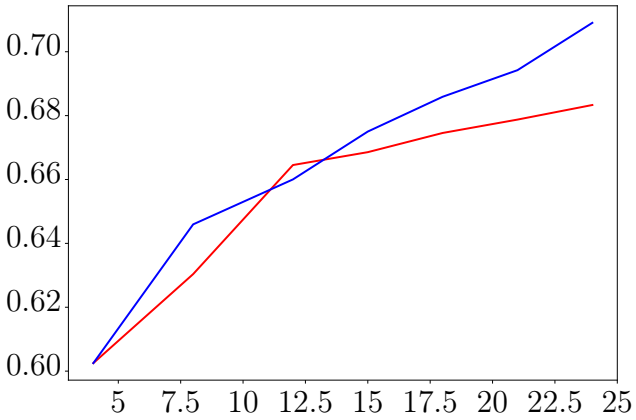

Figure 3.10 Evolution of the hypervolume indicator (Y-axis) over 24 hours of computing time (X-axis) using the large problem instances (30 runs) and 3 Objectives, with NSGAIII (Red) and IBEA (Blue).

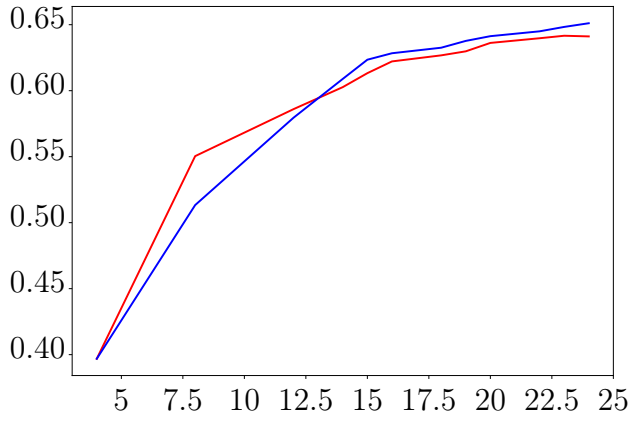

Figure 3.11 Evolution of the hypervolume indicator (Y-axis) over 24 hours of computing time (X-axis) using the large problem instances (30 runs) and 5 Objectives, with NSGAIII (Red) and IBEA (Blue).

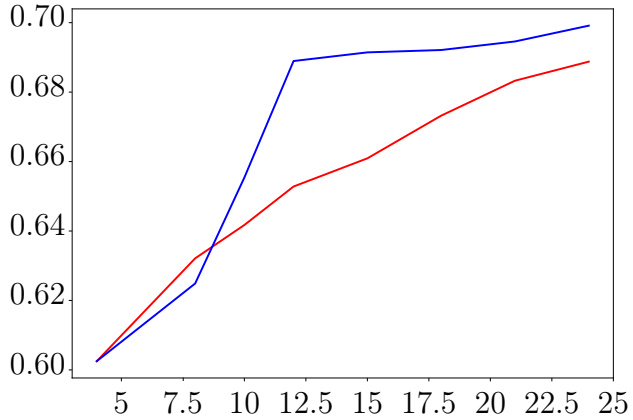

Figure 3.12 Evolution of the hypervolume indicator (Y-axis) over 24 hours of computing time (X-axis) using the large problem instances (30 runs) and 3 out of 5 Objectives, with NSGAIII (Red) and IBEA (Blue).

3.5 Deploying Model in Production

In this section, we will go through how to integrate the proposed approach into DiLeap Logistic. After performing experiments on both the construction heuristic and the metaheuristic based on Genetic Algorithms. We need

to think about how planners can use the solution in a more user-friendly manner. We must also assist planners in comprehending the success metrics returned by our method. This is a section of the literature that is rarely covered; integrating solutions for timetabling problems for school administrations is more prevalent in literature, than for commercial usage.

The next step in the Mandarine Academy Professional Timetabling problem is to put our approach into production. This is an important phase since numerous factors and challenges related to the subject must be investigated. The specifics are covered in the following sections.

3.5.1 Graphical Interfaces for Scheduling

Making the scheduling procedure accessible to the majority of users might be difficult. Because the majority of the tests performed in this study were done using Python code, this procedure will be translated into graphical pages that regular users can comprehend. The suggested graphical interfaces are divided into 3 pages.

Problem input page : This is the page in charge of reading the input data such as training sessions, planning periods, and the required locations. Figure.3.13 shows the graphical interface offered to DiLeap Logistic platform operators for uploading the input spreadsheet file.

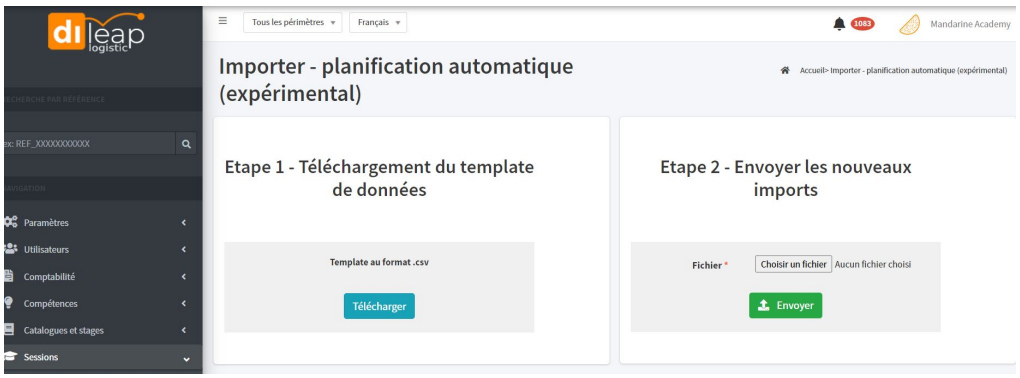

Figure 3.13 Example of input file interface in DiLeap Logistic.

Problem settings page : This is the most crucial page, as it is responsible for the algorithm's execution behavior. While DiLeap Logistic does not yet provide a configuration page for tweaking the current greedy method, planners do not have the option to select constraints or objectives. The suggested page would address the objective selection step first, in which planners can choose which objectives to examine from a supplied list, as well as whether to maximize or minimize selected objectives. Next, during the constraint selection phase, planners can pick which constraints

(hard and soft) to take into account during the planning process. This is an important step since constraints determine the validity of the solutions. Essentially, planners may choose whether or not to make a timetable valid. Planners cannot access complex metaheuristic parameters such as genetic operator probabilities, population size, and so on.

Performance evaluation page : The third and last page is the final solution assessment. An interface is provided to planners for determining weights for each considered objective in order to pick a subset of suggested Pareto solutions. Multi-Criteria Decision Making (MCDM) is another name for this multi-objective decision-making procedure. Pseudo-Weights are one of the solutions considered in this step [33]. This is covered in further detail in Chapter 2 Section 2.1.2.4.

Currently, our approach shows solutions graphically, allowing the planner to accept or reject suggested sessions. This is seen in Figure.3.14. In addition, a dashboard displaying information about planned sessions, mobilized resources and pending actions makes it easier for planners to follow updates about the overall training process. As seen in Figure.3.15 the dashboard is far from complete as the above propositions are being studied now. The overall architecture of the proposed Mandarine Academy Professional Timetabling approach is shown in Figure.3.16.

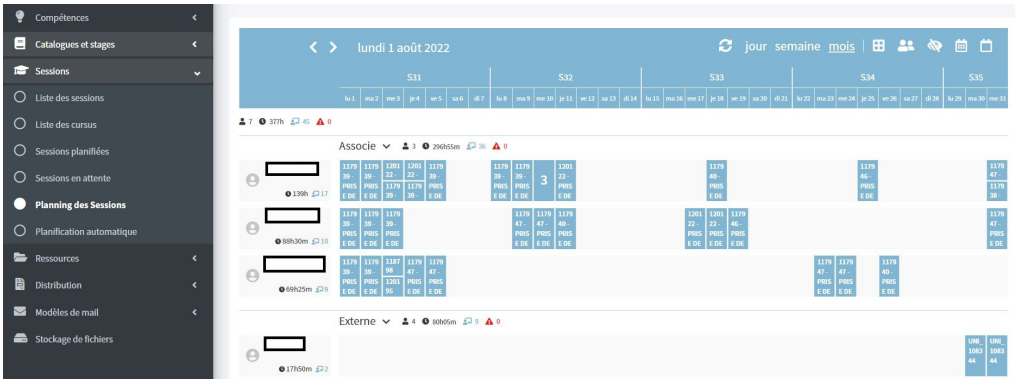

Figure 3.14 Displaying planned sessions in DiLeap Logistic.

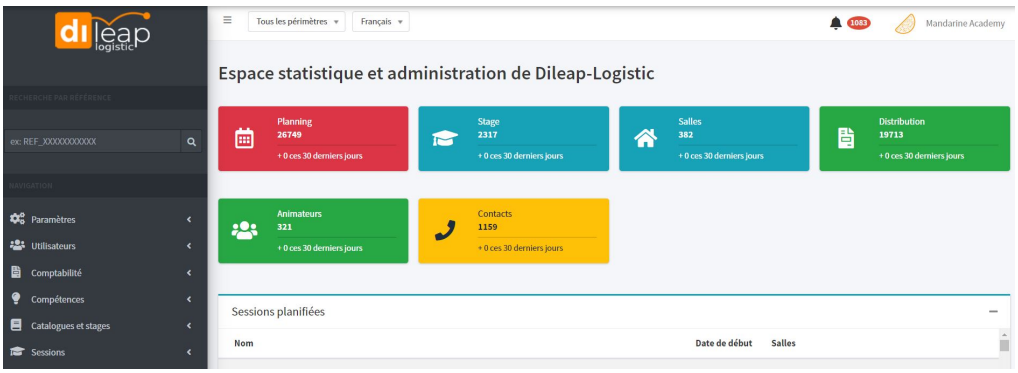

Figure 3.15 Dashboard in DiLeap Logistic.

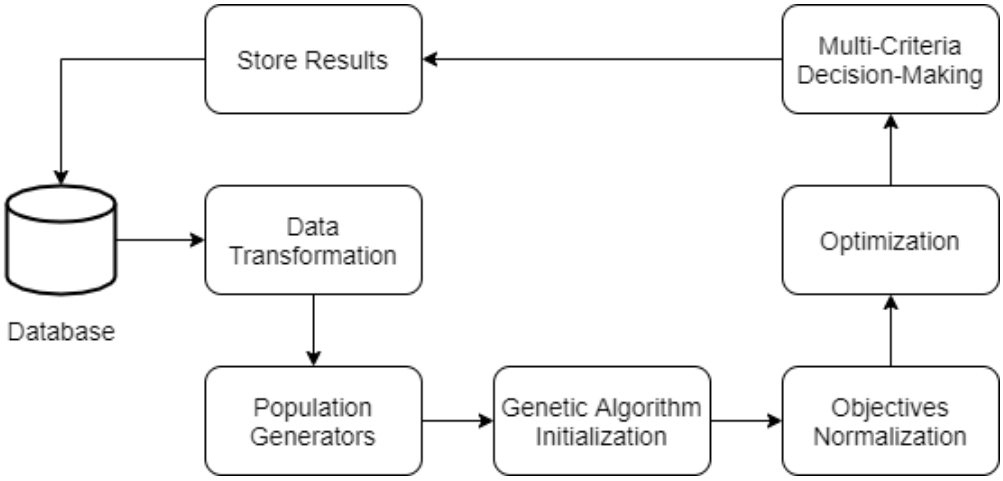

Figure 3.16 Proposed approach architecture.

3.5.2 Performance of Construction Heuristic and Greedy Heuristic

We examined the design and implementation of a construction heuristic responsible for providing timetables that adhere to defined hard constraints in Chapter 3 Section 3.2.2. The objective of implementing a construction heuristic was to create a solution that fulfills the predefined hard constraints that DiLeap Logistic's current greedy approach failed to accomplish.

The greedy approach gives planners quick results, making the process significantly faster than the manual approach. Despite its benefits, it does not respect predefined constraints (it can mobilize resources that are unavailable or in the incorrect location). The construction heuristic seeks to tackle this problem because planners want to spend as little time as possible repairing erroneous training sessions supplied by the greedy method. Time

spent adjusting and deleting faulty timetables can be saved by delivering valid solutions that fulfill company requirements.

The construction heuristic produced valid solutions for the Genetic Algorithms employed in our last experiments. To further test the capabilities of the construction heuristic we ran multiple tests using real-world data under different settings to compare its performance with the existing greedy algorithm within DiLeap Logisitc. Both methods are compared to varying numbers of trainings, ranging from 10 to 100 unique training. With a maximum of 2500 training sessions and a planning time ranging from two months to a full year. We compare the two techniques based on run time, number of planned training sessions, number of valid planned training sessions, and number of hard constraints broken. Table.3.10 displays the results of the described 25 runs.

Table 3.10 Comparison between Construction heuristic and current planning algorithm (greedy approach) on 25 test instances under DiLeap Logistic.

				Run time		**% of planned Se**		**% of valid Se**		**Violated hard constraints**	
#	**Ne**	**Se**	**Days**	*New*	*Old*	*New*	*Old*	*New*	*Old*	*New*	*Old*
1	10	50	60	9,4"	2,3"	50 % (25)	90% (45)	50% (25)	28% (14)	0	H1 (3)
2	10	100	90	25"	3,9"	50% (50)	90% (90)	50% (50)	30% (30)	0	H5 (2)
3	10	150	120	1' 6"	45"	50% (75)	90% (135)	50% (75)	30% (45)	0	H5 (3)
4	10	200	180	2' 4"	59"	40.5% (81)	90% (180)	40.5% (81)	36.5% (73)	0	H5 (4)
5	10	250	360	7' 6"	1' 31 s	50% (125)	90% (225)	50% (125)	39.2% (98)	0	H7 (3)
6	25	125	60	1' 39"	5.2"	43.2%(54)	96% (120)	43%(54)	38% (48)	0	H1 (4), H2 (4), H5 (3)
7	25	250	90	2' 3"	9.2"	39%(98)	96% (240)	39%(98)	34% (85)	0	H1 (4), H2 (4), H5 (7)
8	25	375	120	4', 8"	13"	40% (149)	96% (360)	40% (149)	33% (123)	0	H1 (4), H2 (4)
9	25	500	180	11' 25"	18.2"	43% (217)	96% (480)	43% (217)	38% (193)	0	H1 (4), H2 (4), H5 (10)
10	25	625	360	12' 21"	26"	52% (325)	96% (600)	52% (325)	47% (292)	0	H1 (4), H2 (4), H7 (3)
11	50	250	60	2' 35"	14"	39% (98)	98% (245)	39% (98)	34% (85)	0	H1 (7), H2 (6), H3 (4) , H5 (14)
12	50	500	90	4' 57"	24"	40% (199)	98% (490)	40% (199)	34% (170)	0	H1 (7), H2 (6), H3 (4), H5 (37)
13	50	750	120	6', 11"	33"	44% (331)	98% (735)	44% (331)	33% (253)	0	H1 (7), H2 (6), H3 (4), H5 (23)
14	50	1000	180	8' 26"	56s	44% (441)	98% (980)	44% (441)	37.5% (375)	0	H1 (7), H2 (6), H3 (4), H5 (24), H7 (31)
15	50	1250	360	14' 6"	76,7s	44% (550)	98% (1225)	44% (550)	34% (422)	0	H1 (7), H2 (6), H3 (4), H5 (20), H7 (3)
16	75	375	60	3', 57"	38"	38% (141)	93% (350)	38% (141)	24% (90)	0	H1 (8), H2 (8), H3(1), H5 (13)
17	75	750	90	7'31"	46"	38% (284)	93% (700)	38% (284)	24% (191)	0	H1 (8), H2 (8), H3 (1), H5 (64)
18	75	1125	120	11' 5"	1' 15"	38% (428)	84% (1050)	38% (428)	22% (284)	0	H1 (8), H2 (8), H3 (1), H5 (55)
19	75	1500	180	15' 56"	1' 53"	34% (511)	93% (1400)	34% (511)	29% (431)	0	H1 (8), H2 (8), H3 (1), H5 (61), H7 (30)
20	75	1875	360	53' 23"	1' 40"	48% (902)	93% (1750)	48% (902)	13% (246)	0	H1 (8), H2 (8), H3 (1), H5 (73), H7 (3)
21	100	500	60	3'	40"	44% (221)	92% (460)	44% (221)	15% (76)	0	H1 (12), H2 (10), H5 (19)
22	100	1000	90	11' 42"	1' 20"	39.5% (395)	92% (920)	39.5%(395)	13% (133)	0	H1 (12), H2 (10), H5 (64)
23	100	1500	120	15' 34"	2' 20"	35% (525)	92% (1380)	35% (525)	20% (304)	0	H1 (12), H2 (10), H5 (83)
24	100	2000	180	34'	3' 40"	37% (749)	92% (1840)	37% (749)	16% (324)	0	H1 (12), H2 (10), H5 (99), H7 (47)
25	100	2500	360	90'	5' 21"	42% (1107)	92% (2300)	42% (1107)	14% (345)	0	H1 (12), H2 (10), H5 (89), H7 (2)

Starting with the first evaluation criterion, run time, the greedy approach (old) emerges as the obvious victor. The most complex computations occurred at test run number 25 when the greedy technique takes around 5 minutes to produce solutions, while the construction heuristic (new) takes 90 minutes. While both run times are significantly less than the manual technique, the construction heuristic falls short in this case since it must validate the correctness of offered solutions, whilst the greedy approach does not.

The proportion of planned sessions is the second criterion. When compared to the construction heuristic, the greedy technique achieves a high number of scheduled sessions (>90%). On the other hand, the approach falls short in terms of the percentage of valid training sessions. Starting with test numbers 20 to 25, we see that the greedy approach plans 13%-20% of total sessions, whereas the construction heuristic plans 35%-48% of total sessions. While this is true for more difficult and complex settings, both approaches can perform similarly in initial tests with a slight advantage to the construction heuristic.

The final decision criterion is the number of hard constraints violated. The construction heuristic passed perfectly with 0 broken constraints. This is not the case for the greedy technique, which breaks more and more hard constraints as test complexity increases. Overall, the construction heuristic solves the timetabling problem more successfully than the greedy strategy.

3.6 Conclusion

We propose a solution for the professional course timetabling problem encountered at Mandarine Academy by using popular multi-objective evolutionary algorithms (NSGA II, NSGA III, SPEA 2, MOEA/D, and IBEA).

We proposed a mathematical formulation of our objective functions in addition to hard/soft constraints involved in the planning process.

A constructive heuristic has been designed to produce valid timetables. The heuristic was able to plan most of the trainings in each test instance and provided a high-quality initial population that respects defined hard constraints. Unfortunately, the solutions lack diversity due to the heuristic that creates only feasible solutions.

To optimize soft constraints of valid solutions returned by the heuristic, we constructed several Multi-Objective Genetic Algorithms (MOGAs), including (NSGA II, NSGA III, SPEA2, IBEA, and MOEA/D). The genetic operators for mutation and crossover have been altered to incorporate specific business logic related to the problem at Mandarine Academy. This alteration led to the creation of two custom operators, $MAPT_{co}$ for crossover and $MAPT_{mo}$ for mutation.

A tuning phase using all mentioned algorithms on the small problem instances shows that our proposed mutation operator $MAPT_{mo}$ outperformed the classical Swap operator. For the crossover operator, PMX was chosen based on its better results compared to our proposed operator $MAPT_{co}$.

Experimental results show that both IBEA and NSGA III gave the best results in terms of Hypervolume, Generational Distance, Inverse Generational Distance, and ϵ-Indicator. By using evolutionary algorithms, we were able to obtain more diversified solutions that are significantly better than their initial state. When observing convergence plots, a promising improvement can be made by considering the additional computational time in our experiments in order for the algorithms to handle complex instances. However, even for the research aspect alone, increasing the computational time and resources may not always be feasible.

From a business point of view, this is infeasible as it can come at the price of computational costs or delays to receive results. The use of MOEA/D in our experiments turned out to be a bad investment since it only gave the worst results. This confirms that this algorithm was not suited for this specific problem.

For future work, a different experimental protocol that includes parameter tuning using medium and large instances and not only small test problems will be realized. By using parameters adapted to problem size, we might see a different winner in terms of algorithm performance. Another approach would also be testing random initialization instead of the

construction heuristic to see how it might affect the planning process. This chapter has been the subject of a publication in the IEEE Congress on Evolutionary Computation (CEC 2021) [55] and in the French Association for Operational Research and Decision Support (ROADEF 2021) [100]. The dataset used for the Mandarine Academy Professional Timetabling (MAPT) problem is available online [54].

Chapter 4

Part II: MARS - Mandarine Academy Recommender System

Contents

In Chapter 1 we identified problems related to the e-learning platform (Mooc-Office365-Training) operated by Mandarine Academy through the

study of user ratings/behavior. The first challenge in our work is to handle the cold start problem discussed in detail in Chapter 2, the problem impacts both users and items. The second challenge includes the sparsity problem which manifests in the lack of user data. This may be caused by a variety of elements, such as the lack of suitable user interactions, improper display of them via the User Interface (UI), or the failure of users to express their interests clearly. Finally, the last challenge is the learner drop-out rates which is alarming. Users view 4 videos on average before leaving a platform, according to our research. These 3 challenges raised questions about what can be done and led us to consider recommender systems to increase the number of educational videos viewed by Mandarine Academy users. In this chapter, we outline our approach for automatically delivering content to users based on their preferences using recommender systems. We address the cold start and the sparsity problem, 2 significant issues arising from Mandarine Academy data. We thoroughly explore the data used and go into detail about the entities involved in the recommendation process. This chapter has been the subject of a publication in the Genetic and Evolutionary Computation Conference (GECCO 2022) [56], the French Association for Operational Research and Decision Support (ROADEF) 2022 Congress [57]. The dataset used in this research was made public for benchmarking recommender systems at Harvard Dataverse [53].

This chapter is structured as follows. In Section 4.1, we propose a formulation of the Mandarine Academy Recommender System (MARS) problem as a Multi-Objective Optimization Problem (MOP). We describe entities that play a crucial role in the functioning of a recommender system. In addition, mathematical modeling of hard constraints and company objectives is presented. In Section 4.2, we cover the steps taken to model the Mandarine Academy Recommender System problem within a Multi-Objective Genetic Algorithm (MOGA). This includes solution encoding, initial population generations, and genetic operators (mutation and crossover). In Section 4.3, we define our experimental protocol, starting with a description of the data used in our experiments, followed by a parameter tuning phase of Multi-Objective Genetic Algorithms such as NSGAII, NSGAIII, SPEA2, IBEA, and MOEA/D. Finally, in Sections 4.4 and 4.5 we compare the performance metrics of each algorithm, discuss the choice of genetic operators and analyze the quality of both the final solutions and optimization process. We also compare older and newer graphical interfaces from a user's perspective and discuss the impact on the learning experience.

4.1 Recommendation as a Multi-Objective Optimization Problem

In this section, we propose a formulation of the Mandarine Academy Recommendation Problem as a Multi-objective Optimization Problem. We start by defining entities responsible for both the input and output of recommender systems. Followed by modeling of business goals and company restrictions (hard constraints).

4.1.1 Entities

A description of the main entities involved in the recommendation process is given below. This includes the data structure, data types, and statistical information in addition to high-quality features. Recommendation systems are extensively used for suggesting new items to users, based on their preferences (ratings). We can see the major entities that play a crucial role in our approach, which are users, items, and ratings. Figure.4.1 provides an overview of the relation between mentioned entities.

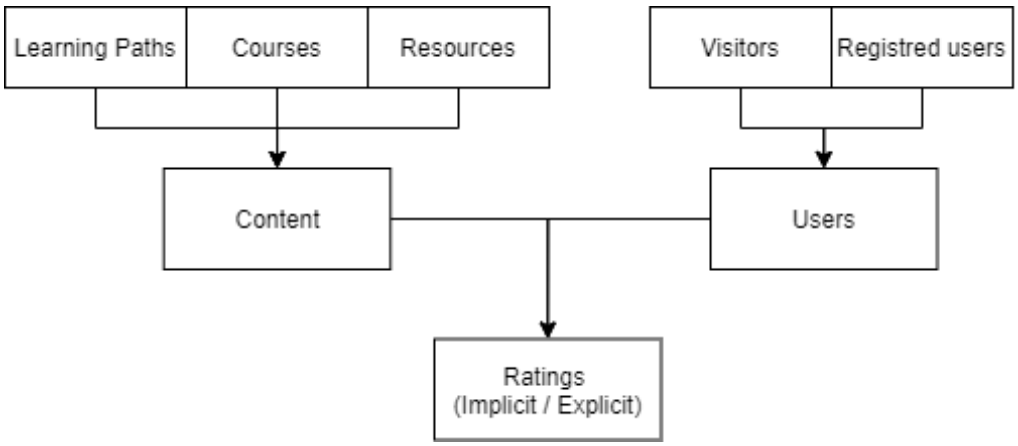

Figure 4.1 Relationship between entities involved in the recommendation process.

Users and Visitors This entity represents the heart of the recommending process. Users and visitors are the powering engines of the platform, they leave behind interactions that express their feelings towards the content or the service. Using Mooc-office365 data, we found that more than 130K are registered on the platform with 92.3% using the French version and 7.7% using the English version. Around 3.5K are monthly connected users. Some of the attributes that describe users are Registration date, Profile update date, Country/city, Time zone, Role, last login, Privilege, Company codes, Job codes, Domain codes, and Partner codes. However,

to protect users' personal information and reinforce privacy measures, only the user's unique identifier and job category are used.

Resources (Tutorials, use cases) Tutorials and use cases are available in a video format with the option to choose subtitles in 11 different languages. These short videos focus on specific software, skills, or jobs. Typically, these resources are integrated into courses, and one resource can be used in multiple courses. With a collection of more than 1300 tutorials and 113 webcasts (French version), users might find it challenging to locate relevant content. Our goal is to provide an ordered list containing resources for users.

Table 4.1 Overview of resource features (French version of Mooc-office365-training).

Feature	**Feature Desc**	**Type**	**Missing**
Resource ID	Unique identifier	*Int*	*0%*
Language	Content language	*Category*	*0%*
Title	Content title	*Text*	*0%*
Description	Content description	*Text*	*0%*
Views	Number of views	Int	0%
Creation Date	Content upload date	*Date*	*0%*
Duration	Duration in seconds	*Int*	*0%*
Type	Tutorial or Use Case	*Category*	*0%*
Level	Content difficulty	*Category*	*75.94%*
Job	Related professions	*Category*	*77.53%*
Software	Related software	*Category*	*7%*
Theme	Related theme	*Category*	*12.54%*

Table.4.1 provides an overview of important features that describe resources (Tutorials and use cases).

Courses and Learning Paths Courses aim to deliver a certain level of skills (e.g., Beginner in Microsoft Word). They contain different types of items, ranging from modules (structured lists of tutorials and use cases), quizzes, SCORM, documents to read, webcasts, etc. The learning experience here is defined by the company's pedagogical team as each step is defined prior to course upload.

Learning paths on the other hand tend to structure multiple courses together. They aim to deliver a path to master a suite of skills, acquire knowledge about a certain theme/ sector or provide necessary skills for

certain jobs. There are 41 learning paths and 146 courses currently in Mooc-Office365 (french version). Both courses and learning paths share the same attributes as resources. These features are presented in Table.4.2 and Table4.3 for courses and learning paths respectively.

Table 4.2 Overview of course features (French version of Mooc-office365-training).

Feature	**Feature Desc**	**Type**	**Missing**
Course ID	Unique identifier	*Int*	*0%*
Language	Content language	*Category*	*0%*
Title	Content title	*Text*	*0%*
Description	Content description	*Text*	*0%*
Views	Number of views	Int	0%
Creation Date	Content upload date	*Date*	*0%*
Duration	Duration in seconds	*Int*	*0%*
Level	Content difficulty	*Category*	*86%*
Job	Related professions	*Category*	*89%*
Software	Related software	*Category*	*14%*
Theme	Related theme	*Category*	*19%*

Table 4.3 Overview of learning path features (French version of Mooc-office365-training).

Feature	**Feature Desc**	**Type**	**Missing**
Path ID	Unique identifier	*Int*	*0%*
Language	Content language	*Category*	*0%*
Title	Content title	*Text*	*0%*
Description	Content description	*Text*	*2.5%*
Views	Number of views	Int	0%
Creation Date	Content upload date	*Date*	*0%*
Duration	Duration in seconds	*Int*	*0%*
Level	Content difficulty	*Category*	*47.5%*
Job	Related professions	*Category*	*70%*
Software	Related software	*Category*	*12.5%*
Theme	Related theme	*Category*	*25%*

4.1.2 Constraints

Our goal is to provide an ordered list containing items for users. This list can be the same for visitors or personalized per user. Each personalized recommendation will be the result of an optimization process run especially on each user to provide the best personalization levels. Since we are working on an optimization problem, we must define our hard constraints in order to determine if a solution is feasible or not. Hard constraints can be defined as conditions to optimization problems that a solution must satisfy. If a solution respects all conditions, it's called a feasible or a valid solution.

The problem at hand exhibits combinatorial traits, as we need to select a subset of items from a large pool of tutorials and webcasts to recommend to each user. The combinatorial nature of the problem becomes more complex when considering multiple conflicting business objectives and a large number of users, making it an optimization problem.

- Recommended items inside a recommendation list L must be unique and contain no duplicates.
- Length of a recommendation list L must not exceed a fixed length k.

4.1.3 Objectives

In Chapter 2 we covered offline and online evaluation metrics for recommender systems. Offline methods use historical data to show how closely predictions match actual user ratings. The majority of literary works focus on accuracy metrics like RMSE or Precision, as seen in Table.2.4, making the use of offline methods essential for evaluating the quality of recommendations.

But as we've seen in Table.2.4, accuracy isn't the only factor influencing how good recommendations are. Mandarine Academy hopes to pique users' interest in subjects other than what they are already studying. The company wants to highlight the variety and novelty of recommendations to aid users in finding potentially interesting items. Such metrics will track how often users discover content they hadn't previously considered looking for but that they now find to be interesting.

The business can specify which metrics will be taken into account before starting the recommendation process to provide greater flexibility. The various goals that can be used by the Mandarine Academy recommender engine are described in more detail below.

(Objective 1) Maximize similarity with user profile. This is done by calculating the overall cosine similarity between all items in the user profile and items in the recommended list. Cosine similarity is a widely used metric in information retrieval and recommendation systems, measuring the cosine

of the angle between two vectors in a multi-dimensional space. A higher score means higher similarity. The cosine similarity between two items is calculated as follows:

$$csim(a, b) = \frac{a \cdot b}{\|a\|\|b\|} \tag{4.1}$$

Where a and b are the feature vectors of the two items being compared. The overall similarity between the recommended list and the user profile is then calculated using the following equation:

$$Psim = \frac{\sum_{L}^{i=0} \sum_{L}^{j=0} csim(r_i, u_j)}{L} \tag{4.2}$$

Where L is the recommended list (solution) and r_i is the item number i from L. The user profile is expressed as n where u_j is the item number j from n. $csim$ is the item-item cosine distance matrix.

(Objective 2) Maximize diversity which is responsible for how dissimilar recommended items are for a user. This can be achieved by using the Intra-List Similarity metric (ILS) [128]. In a nutshell, we are calculating the average cosine similarity of all items in a list of recommendations. Note that this objective is conflicting with the first objective.

$$Rdiv = \frac{\sum_{L-1}^{i=0} \sum_{L}^{j=1} csim(r_i, u_j)}{tc} \tag{4.3}$$

Where L is the recommended list (solution) and r_i is the item number i from L. The user profile is expressed as n where u_j is the item number j from n. $csim$ is the item-item cosine distance matrix. tc is the item pairs count.

(Objective 3) Maximize novelty. In this objective, we are trying to recommend less popular items and focus on items having a smaller number of views that were added recently to the catalog. A scoring function that sums the number of views and number of days since release, returns the median. The smaller the median, the more novel the items are.

$$Rnov = \frac{\sum_{L}^{i=0} ns(r_i)}{L} \tag{4.4}$$

Where L is the recommended list (solution) and r_i is the item number i from L. ns is the novelty score.

(Objective 4) Minimize Root Mean Square Error (RMSE). Refer to Chapter 2, Section 2.3.3 for more details about RMSE. The rating predicted by the recommender is $\hat{r}_{ui}$ and the actual value given by the user is r_{ui} while $\hat{R}$ is the number of ratings.

$$RMSE = \sqrt{\frac{1}{\left|\hat{R}\right|} \sum_{\hat{r}_{ui} \in \hat{R}} (r_{ui} - \hat{r}_{ui})^2} \tag{4.5}$$

(Objective 5) Maximize the Normalized Discounted Cumulative Gain (nDCG). We will be using **nDCG@5** which corresponds to the number of relevant results among the top 5 recommended items. Refer to Chapter 2, Section 2.3.3 for more details about nDCG.

$$nDCG_p = \frac{DCG_p}{IDCG_p} \tag{4.6}$$

With $IDCG_p = \sum_{i=1}^{|REL_p|} \frac{rel_i}{\log_2(i+1)}$ and $DCG_p = \sum_{i=1}^{p} \frac{2^{rel_i}-1}{\log_2(i+1)}$. Where REL_p is a list of top $p = 5$ relevant items (ordered by relevance). rel_i is the graded relevance of the result at position i.

4.2 Genetic Modeling

The previous section demonstrates the different objectives associated with evaluating both business and prediction performance metrics. The different objectives present a competitive behavior, this is demonstrated in objective 1 which seeks to maximize precision, while objectives 2 and 3 seek to provide novel and diverse items which can alter the precision scores. This brings the need for not a single solution but multiple solutions since we don't want to sacrifice the performance of one objective to the detriment of others. As demonstrated, we are faced with many competing objectives to optimize solutions of multiple unique items, which classifies this problem as a Multi-Objective Combinatorial Optimization Problem (MOCOP). Since we're interested in diverse solutions, both Dominance and Indicator-based methods which have proven efficiency in handling multi-objective problems [71, 55] are considered. These methods include algorithms such as:

- Non-dominated Sorting Genetic Algorithm II (NSGA-II) [121].
- Non-dominated Sorting Genetic Algorithm III (NSGA-III) [35].
- Multi-Objective Evolutionary Algorithm by Decomposition (MOEA/D) [138].
- Indicator-Based Evolutionary Algorithm (IBEA) [142].
- Strength Pareto Evolutionary Algorithm (SPEA-2) [143].

A detailed review of MOCOPs and Dominance and Indicator-based methods is provided in Chapter 2 Section 2.1.3. The above algorithms fall under the Genetic Algorithms (GAs) category, which begins with the choice of the chromosome encoding (solution representation) of the problem to be solved. Following this we further expand on our initialization strategy to provide a high-quality initial population for our experiments, followed by genetic operators.

4.2.1 Solution Encoding

Following the works of [4, 118, 20], we define a solution as a list of unique item identifiers denoted by the symbol L. This list represents a set of recommendations for a single user. With the first item being the first to be recommended. The list will have a fixed length of k and will include items specific to each user. This solution can be seen from the end-user perspective as the example shown in Figure.4.2.

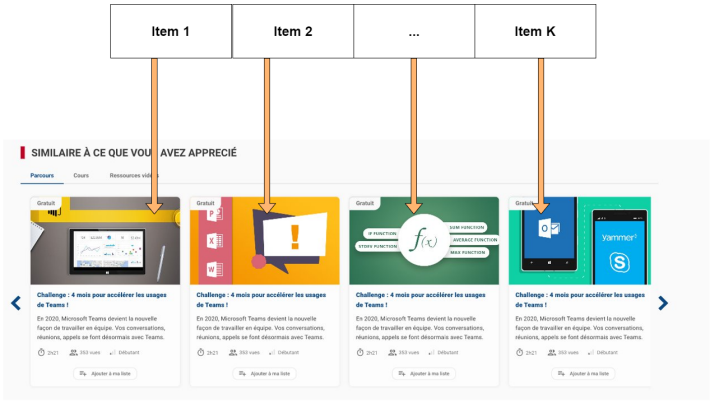

Figure 4.2 Encoded solution seen from a front-end perspective.

4.2.2 Initializing Population

After defining a structure that represents our solution, we proceed to generate multiple solutions for our problem. A set of solutions is called a population.

In order to generate a good initial population that contains various recommendations for each user, we took advantage of available data (interactions and content descriptors) to create multiple recommendation engines. For our initial population, we will be using the following approaches:

- Random.
- Content-Based Filtering (CBF).
- Association Rules (FP-Growth).
- Collaborative Filtering (CF) - Item Based.
- Collaborative Filtering (CF) - User Based.
- Collaborative Filtering (CF) - Model Based (ALS, SVD, SVD++, NMF, SlopeOne).

Collaborative Filtering (CF) Methods: The implementation of Item/User/Model-based collaborative filtering algorithms was facilitated by the Surprise python library, which is specifically designed for recommender system implementation [68]. The Item and User based methods apply a k-nearest neighbor (KNN) algorithm [108] around items or user's profile to retrieve neighbors/similar items/users. Using the Surprise library, we were able to compare the performance of model-based algorithms:

- CF-Model Based (SVD): The famous Singular Value Decomposition (SVD) algorithm [86] popularized by Simon Funk during the Netflix Prize.
- CF-Model Based (SVD++): An optimized SVD algorithm to enhance the accuracy of prediction by generating implicit feedback [85].
- CF-Model Based (NMF): Based on Non-negative Matrix Factorization [115].
- CF-Model Based (SlopeOne): Based on the SlopeOne algorithm found in [88].
- CF-Model Based (ALS): Alternating Least Squares (ALS) [66] matrix factorization implemented in PySpark and designed primarily for implicit interactions.

The procedure follows standard machine learning steps, in which we divide our user ratings into train and test sets, fit estimators with training data, and execute prediction and evaluation on test data. For Model and User/Item-based methods, results are in the form of (user id, and recommended items). Item-based methods can also return (item id, related items). Section 4.3.1 goes through the specifics of each model and the training/testing process.

Association Rules (FP-Growth): Content-based Filtering (CBF) and Association Rules (AR) is implemented differently than collaborative filtering methods. Beginning with association rules, the algorithm attempts to discover frequent items in historical data. The FP-Growth algorithm [91] is implemented in our work using the Pyspark library. FP-Growth was chosen due to its superior performance and efficiency in finding frequent item sets when compared to older methods such as Apriori [69]. The goal is to uncover all co-occurrence relationships, also known as associations, between items found in historical data.

Market basket data analysis is a typical application of association rules that tries to uncover how items purchased by clients in a supermarket (or store) are related. Similarly to collaborative filtering approaches, we fit the

FP-Growth model with training data and evaluate test data after splitting user ratings. The user-specific item-sets function returns results in the form of (user id, recommended items), while the frequent item-sets method returns results in the form of (item id, similar items).

Content-Based Filtering (CBF): Finally, for content-based filtering, Natural Language Processing (NLP) techniques are used to compare the textual information (title, description, subtitles) of each item. User ratings are only used here to find what a user likes. Actual predictions aren't based on user ratings but on item features. NLP approaches to study human languages, specifically how to process and interpret the text. NLP is frequently used for text classification tasks, which involve providing categories to textual content.

Textual information can be transformed into numerical features that our models can understand using NLP approaches. When combined with the cosine similarity metric we can calculate distances between item features, eventually creating an item-to-item similarity matrix. This permits the selection of any item and identifies others with similar characteristics. The following steps were taken to process and transform textual features into numerical features:

1. Text processing: This includes removing special characters, HTML tags, and stop-words. Only considers numbers or letters (alphanumeric).

2. Tokenization: The process of converting a text into a numerical data format appropriate for machine learning. It essentially divides sentences, words, and characters.

3. TF-IDF: Term Frequency-Inverse Document Frequency[109] is a very popular text vectorization approach that combines 2 concepts, Term Frequency (TF) and Document Frequency (DF). TF indicates how important a specific term is in a document by counting its occurrences. DF indicates how common the term is.

After applying text processing and vectorization methods, our features are transformed to meet machine learning requirements. We apply the cosine similarity distance on features to generate the item-to-item matrix. Results are in the form of (user id, and recommended items) using the similarity of items within a target user's profile (e.g., history of seen items). Recommendations can also come in the form of (item id, or similar items) using the similarity matrix alone. The SpaCy library[64] was used to implement the above NLP tasks. It's an open-source Python library for parsing and comprehending large amounts of text. It contains a multitude

of efficient implementations of common algorithms, with available models catering to specific languages (English, French, German, etc.).

From each of the above approaches, recommendation lists are generated per user. In the event that an algorithm is unable to provide personalized recommendations, we have defined a fallback method, which is the "Random" approach. The random approach essentially returns a random item from the catalog.

4.2.3 Genetic Operators

We list the different operators used to conduct our research, starting with the selection operator which is described in detail in Chapter 2, Section 2.1.1.4. The roulette wheel provides a middle way for selecting appropriate solutions from a population of solutions. The strategy attempts to identify solutions that aren't necessarily the fittest, but at the same time, also not choosing solutions in a totally random way. For crossover and mutation, we provide more details in the following paragraph.

Crossover Operators like One-Point and Two-Point crossover are widely used in the literature [67, 4, 20, 118]. The idea behind such operators is simple, the One-Point crossover selects a single point in both parents and swaps the elements after that point, while the Two-Point crossover selects two points and swaps the elements between those points. As a result, some elements from parent 2 may become redundant in parent 1, requiring a repairing mechanism to ensure a valid solution.

Mutation Random mutation is frequently found in the literature [118], [20] along with 1-point mutation [4], 2-point mutation [132] and Uniform mutation [4]. The idea behind this is simple, changing items inside the solution with other items. We propose a custom mutation operator named $MARS_{mo}$ to be compared to classical operators. The concept behind $MARS_{mo}$ is to choose N, with $(1 \leq N \leq k/2)$, elements from a candidate solution. The selected elements are then randomly swapped with either (1) Similar items (Content-Based or Item-Based), (2) Random (3) Novel (Recently added to the catalog) items. The pseudo-code of $MARS_{mo}$ can be seen in Algorithm 3.

4.3 Experimental Design

This section describes the experimental protocol implementation used at Mandarine Academy to tackle the recommendation problem. Figure.4.3 depicts the many steps considered in the experimental design. Our first task

Algorithm 3: Pseudo-code of $MARS$ custom mutation operator $MARS_{mo}$.

Input: Solution, Mut_{pr}
Output: Solution
`/* Select a method for replacing items */`
1 $ReplaceMethod \leftarrow Random(Similarity, Random, Novelty)$ **for** ***each*** *item in Solution* **do**
2 **if** $Mut_{pr} \leq Random([0.0, 1.0])$ **then**
`/* Replace an item using a replacement method */`
3 Solution(item) ← Replace Method(item)

is to collect, clean, and transform the input data that will be used throughout experiments. We present findings and discuss the decisions made when choosing the data and candidate recommender engines to generate the initial solutions. We chose a parameter-tuning phase where Multi-Objective Genetic Algorithms are compared under diverse objective sets to obtain best performing settings. The last subsection evaluates and interprets the findings of selected metaheuristics on several runs.

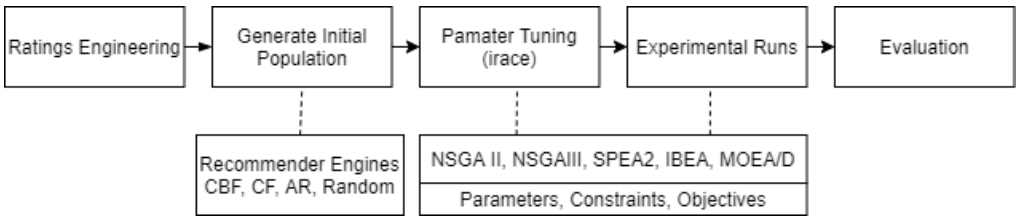

Figure 4.3 Overview of the experimental protocol used in MARS problem.

4.3.1 Dataset Engineering

In this subsection, we detail the considerations taken into the data selection process. Back in Chapter 1 Section 1.1.2, we showcase initial findings upon investigating usage data provided by Mandarine Academy on its product Mooc-Office365[1] (French version). In the following experiments, we used the French catalog as it's the most accessed by a large margin (93% of visitors). Observations were taken from early 2018 to late 2020.

Observing Table.1.2 for explicit interactions, we see that not only roughly 1% of consumers have directly given their feedback, but also a big portion of the catalog seems unreached. Explicit ratings are insufficient to create user profiles and if models are trained on a small number of users, the reach

[1]https://mooc.office365-training.com/

of recommendations will decrease. However, Table.1.3 illustrates that implicit interactions, specifically page views, have a greater reach among users and content. Because our method focuses on resources (tutorials and use cases), we examine both resource page visits and view portions (resource view time). Table.4.4 includes statistical analysis for further examinations.

Figure.4.4 and Figure.4.5 illustrate a count plot of grouped viewed resource pages and videos per user (Avg. Count/User). Observations show that both distributions are right-skewed (positive). Also, a high number of pages/videos are seen by a small number of users compared to the majority of users. A user will visit 21 resource pages on average while watching an average of 8 items. We discovered clear evidence of outlier occurrence by visualizing figures and calculating averages, which we handled by utilizing the Interquartile Range (IQR) approach. It's a commonly used rule that a data point is an outlier if it is more than 1.5*IQR.

Figure.4.6 and Figure.4.7 illustrate the change in distribution plots and the Avg. Count/User after conducting outlier elimination. Although both interactions involve a limited number of items consumed by a user, there is no ground truth that shows whether seeing a content page several times leads to improved user satisfaction. Our assumption is that increased or decreased page views have no effect on a user's learning experience. Furthermore, this is not true for resource view time, which can imply increased interest as viewing duration grows. For example, let's take some learners who spent a certain amount of time viewing a tutorial; the longer a user observes, the more they learn and update their knowledge. The opposite is also true: if a user does not view enough of the video, they will not obtain a complete comprehension of the subject. Some extreme cases, such as advanced users watching a few portions of a video just to learn about a small detail, aren't considered in our hypothesis.

Table 4.4 Overview of statistical information on implicit resource ratings.

	Resource Page View	**Resource View Time**
Unique Users	28418	9789
Unique Items	1309	1287
Number of Rows	253827	68,894
Avg. Count/User	21	8
Unique Users (IQR)	18519	8652
Unique Items (IQR)	1179	1099
Number of Rows (IQR)	111000	26649
Avg. Count/User (IQR)	4	4

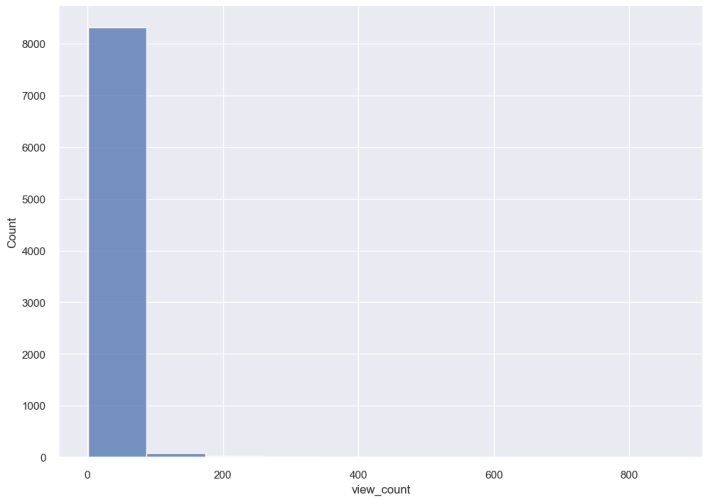

Figure 4.4 Distribution of user and resource page visit count.

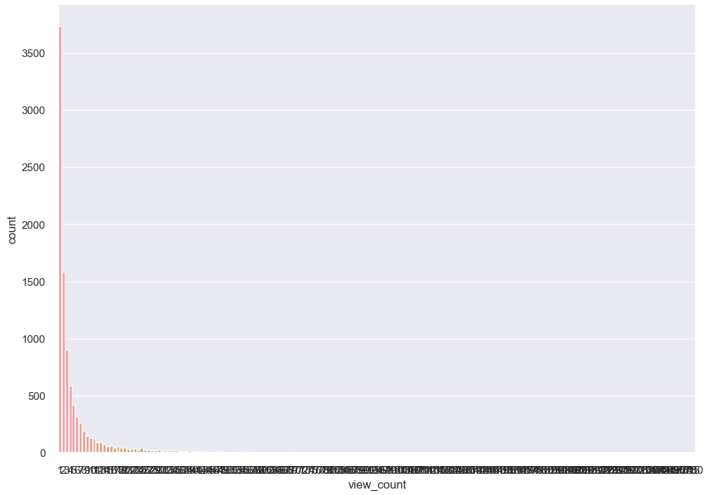

Figure 4.5 Distribution of user and video watch time count.

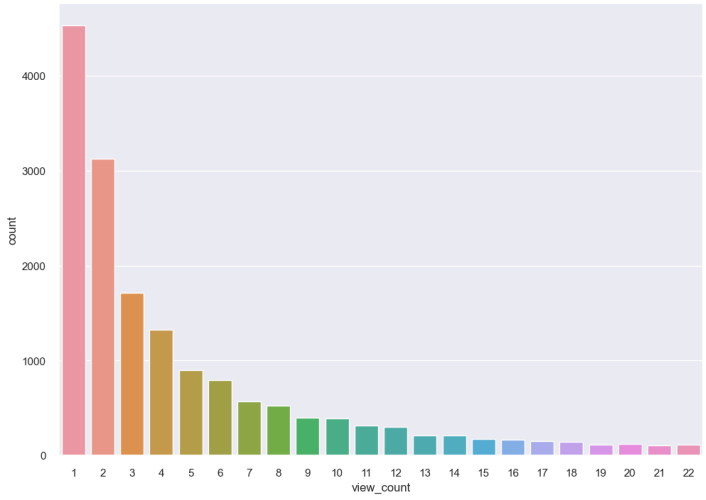

Figure 4.6 Distribution of user and resource page visit count after outlier removal.

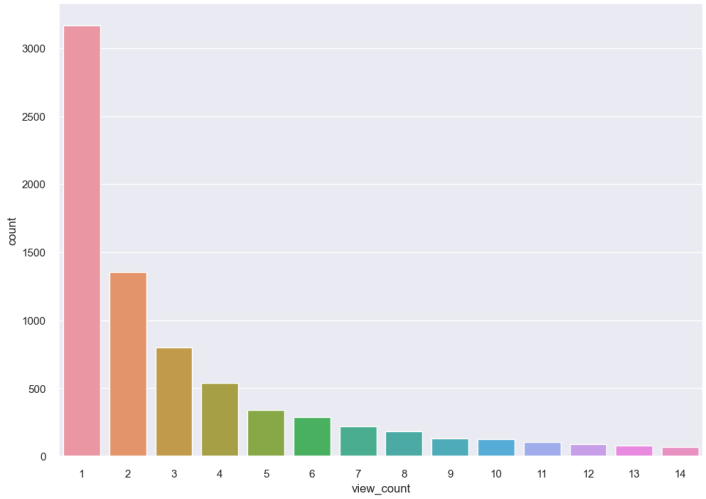

Figure 4.7 Distribution of user and video watch time count after outlier removal.

Table 4.5 Implicit Interactions (View Portions) from Mooc-Office-365 (French) starting 2018 to late 2020.

User ID	Item ID	View Portion	State	Date
792	995	7%	Not considered	2018-01-21
3475	516	30%	In progress	2018-05-25
...	...	...	...	...
687	520	80%	Finished	2020-08-14
542	498	45%	In progress	2020-09-16

Presenting a new scoring system We investigate the different features and values contained in the data set in Table.4.5 since we found resource view portions to be a better fit than resource page view.

The dataset has a total of 68,894 ratings (prior to IQR) ordered from oldest to most recent. The company uses the following states to describe viewing events:

- (1) Not considered: View portion from 0% to 10% of the video.
- (2) In progress: View portions from 11% to 69% are considered equal.
- (3) Finished: View portions from 70% to 100% considers the user has finished watching the item.

In the previous paragraph, we positioned our hypothesis on the degree of viewing an item and its impact on the overall impression. The assumption is that longer viewing times indicate a higher user interest. However, the current scale doesn't capture user interest and fails to correctly identify important items. The fact that view times from 11% to 69% are considered equal doesn't make sense if we apply the previous example of learners. Learning on 11% of the content is not the same as learning on 69%, hence a new scoring system that integrates past observations must be explored in order to better capture user behavior. We detail below a new scoring system that builds on the previous scale but incorporates more levels of appreciation. The new approach translates the percentages of viewing time according to new states. The assumption is that longer viewing times indicate a higher user interest:

- (1) No interest: viewings from 0% to 20% of the video.
- (2) Small interest: viewings from 21% to 40% of the video.
- (3) Medium interest: viewings from 41% to 60% of the video.
- (4) High interest: viewings from 61% to 80% of the video.

- (5) Finished: viewings from 81% to 100% of the video.

This introduces 5 different levels that have varying degrees of importance and reassembles the classical 5-star rating system found in explicit ratings. A comparison between the current and proposed scale is presented in Fig.4.8 and Fig.4.9.

Both current and new methods reveal that the majority of visitors are about to or have completed viewing the content. This means that the great majority of people have finished their lectures. However, we can still interpret small groups of users with low viewing time. Because this isn't a recurring occurrence, we feel it could be due to poor user experience (UX), as consumers may stumble into irrelevant content and return to seek something more manageable. Perhaps the title wasn't clean enough because descriptions aren't always available, or the video content isn't clear enough for the user to understand.

Including such low-rated items in the user profile can lead to bad recommendations. Nevertheless, the use of high ratings to train our recommender systems might seem a better solution. Learning from high-rated content will provide a better-personalized learning experience for users as the aim here will be to provide content that is most likely to match their habits/tastes, which manifests in highly viewed items. We define a threshold of 50% or more viewing time per resource for a rating to be considered in the user's profile.

To address the issue of making sure the user has actually stayed in front of the video when it is considered "finished," we can add an additional metric to the scoring system that takes into account user engagement during the video playback. This can be achieved by tracking user interactions, such as pausing, rewinding, or adjusting the volume, which can provide insights into whether the user was actively engaged with the content. By incorporating user engagement metrics into the scoring system, we can better determine if a user has truly finished watching a video and remained engaged throughout its duration. However, given the current setting and technical difficulties, such a feature is considered in future stages of research. The dataset used in this research was made public for benchmarking recommender systems at Harvard Dataverse [53].

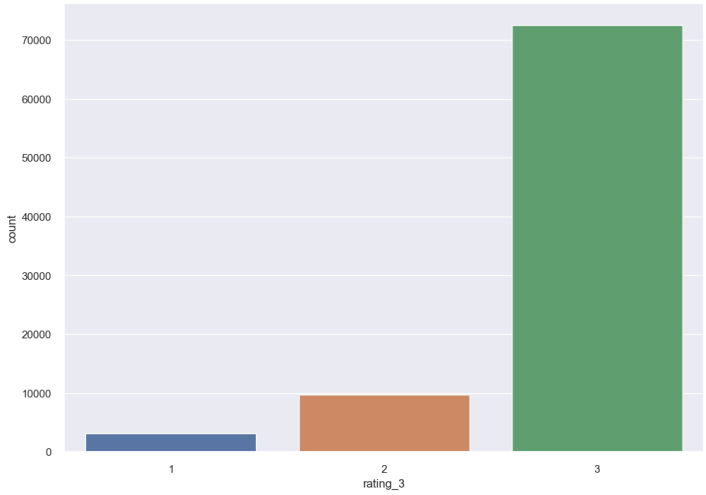

Figure 4.8 Count of implicit interactions (View Portions) per user (old method).

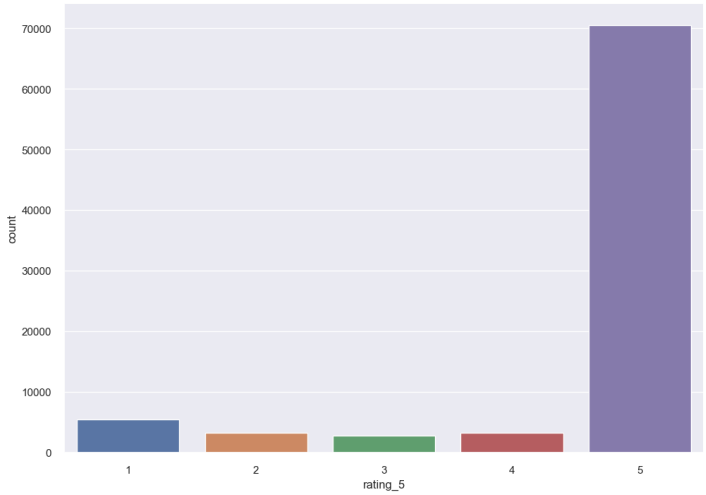

Figure 4.9 Count of implicit interactions (View Portions) per user (new method).

Initial population generation We previously selected implicit ratings, proposed a new scale, and excluded low-rated items from the data. For the time being, data wrangling is complete, and we can move on to train the models we discussed in Section 4.2.2 about establishing a high-quality initial population of solutions using several recommendation engines:

- Random.
- Content-Based Filtering (CBF).

- Association Rules (FP-Growth).
- Collaborative Filtering (CF) - Item Based.
- Collaborative Filtering (CF) - User Based.
- Collaborative Filtering (CF) - Model Based (ALS, SVD, SVD++, NMF, SlopeOne).

Each approach has a specific logic that uses to provide recommendations. While collaborative filtering methods take advantage of user ratings, content-based filtering requires knowledge of item features. Further details on the functioning of each approach are seen in Chapter 2 Section 2.3.2. In our case, we will be picking each approach except for the Model-Based collaborative filtering method where we will be working with both ALS and one more candidate.

Each approach has its own logic for making recommendations. While collaborative filtering approaches rely on user evaluations, content-based filtering necessitates knowledge of item characteristics. More information on how each strategy works is found in Chapter 2 Section 2.3.2. In our scenario, we will select each method except the Model-Based CF methods, where we will work with both ALS and one more candidate.

Since we're using the surprise library, the best-performing model-based approach will be chosen among candidates (SVD, SVD++, NMF, SlopeOne). Since they fall under the same category, the logic behind each algorithm is similar, for this reason, we use it to compare their performance under implicit data.

After splitting the data into dependent and independent variables. We divide the dataset into a 20% test set and an 80% train set. 10-Fold cross-validation technique illustrated in Figure.4.10 is utilized. Using K-fold cross-validation, we may train and test the model on distinct subsets of the data several times. This way, we may not only analyze the performance of our model, but our model will also perform better because it was trained on more data. To evaluate each run, we chose the most popular predictive metrics for recommender systems seen in Chapter 2 Section 2.3.3, which are Mean Absolute Error (MAE) and Root Mean Squared Error (RMSE). In brief, both metrics calculate the average differences between predicted ratings and actual ratings.

The preliminary results are displayed in Figure.4.11, which compares the performance of model-based collaborative-filtering algorithms using the RMSE, MAE, and MSE metrics. Naturally, SVD++ achieved the best prediction metrics, after all, it's designed to enhance the accuracy of predictions using implicit ratings. Unfortunately, SVD++ does not have the quickest training or prediction times. In terms of prediction metrics, the difference between SVD++ and SVD is not significant when compared to

fit/test times. Using SVD instead of SVD++ allows for faster model training, which speeds up the entire process. Because we are not testing in production mode scenarios, we are considering the SVD++ method.

Users will receive personalized results using the ALS and SVD algorithms, as well as other techniques (CBF, AR, Random, User/Item-Based). Now that we've decided on the final recommendation engines that will be used to generate initial solutions, we can go on to the next phase, which is to talk about the Multi-Objective Genetic Algorithms (MOGAs) that will be utilized to solve the optimization problem.

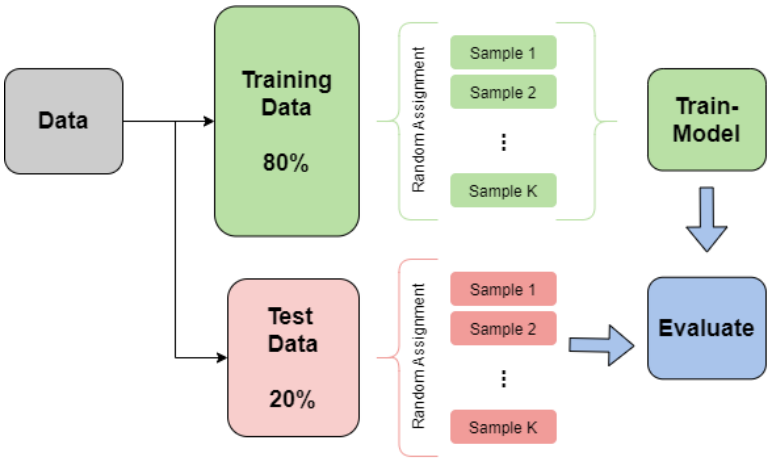

Figure 4.10 Experimental protocol for the initial population.

	Memory Based (CF)		Model Based (CF)			
Algorithm	Item-Based	User-Based	SVD	SVD++	NMF	SlopeOne
RMSE	0.919	0.891	0.896	**0.888**	0.992	0.934
MSE	0.844	0.795	0.803	**0.789**	0.985	0.874
MAE	0.490	**0.478**	0.515	0.497	0.665	0.499
Fit Time (s)	**0.343**	1.863	3.778	63.933	3.636	0.375
Test Time (s)	0.687	1.048	**0.037**	0.515	0.039	0.477

Figure 4.11 Results of running collaborative filtering approaches for initial population generation.

4.3.2 Experiments: Performance Optimization

Table.4.4 shows insights on statistical information of resource view portions point to a hidden potential problem. Our approach's **Objective 5** tries to maximize the **nDCG@5**, which means this can only apply to people who have seen at least 6 items. Because certain objectives are incompatible with all users, we must partition our data before using it as input to meta-

heuristics. Following this discovery, we divide our dataset into 2 groups based on user profile length:

1. Group 1: Consists of 7223 users having less or equal to 5 items in their viewing history.

2. Group 2: Consists of 2566 users having more than 5 items in their viewing history.

Because not all objective functions can be calculated by each group, having two groups necessitates using only a subset of all available objectives. We adapt our method to possible future trends by assigning various objectives to each user group. In this situation, the flexibility of objective selection allows personalized recommendations to be delivered to both groups despite differences in their profiles.

Choice of Multi-Objective Evolutionary Algorithms: As demonstrated in Section 4.1.3, we are faced with many competing objectives. We have discussed the need to apply Multi-Objective Dominance and Indicator-Based methods due to their efficiency and ability to provide diverse solutions. These methods include algorithms such as Non-dominated Sorting Genetic Algorithm II (NSGA-II)[121], Non-dominated Sorting Genetic Algorithm III (NSGA-III)[35], Indicator-Based Evolutionary Algorithm (IBEA)[142], and Strength Pareto Evolutionary Algorithm (SPEA-2)[143]. Another popular approach is the Multi-Objective Evolutionary Algorithm by Decomposition (MOEA/D) [138], which decomposes the original problem into a set of subproblems and solves them simultaneously.

A detailed review of MOCOPs and Dominance and Indicator-based methods is provided in Chapter 2 Section 2.1.3. The above algorithms fall under the Genetic Algorithms (GAs) category.

Parameter Tuning: Instead of arbitrarily selecting predefined parameters and applying them to all algorithms, another protocol is utilized to give a fair performance comparison. It employs the i-race package [94], which implements an iterated racing strategy to automatically find optimal settings. The package focus on improving optimization algorithms and machine learning models. For further details on irace please refer to Chapter 2 Section 2.3.3.

In essence, irace operates by first receiving as input a parameter domain definition. This contains the target algorithms that will be tuned and parameters with possible values to be considered. For the MARS problem Table.4.6 details different parameters with their description and ranges. Because the algorithms in our experiments are population-based, common parameters are found, such as:

- Population size (Pop): Number of generated solutions at each generation.
- Crossover operator (C_o)
- Crossover probability (C_p): Probability for parent solutions to recombine.
- Mutation operator (M_o)
- Mutation probability (M_p): Probability for a child solution to be mutated.
- Kappa (Kp): Fitness scaling value for indicators (IBEA).
- NSP: Neighborhood Selection Probability (MOEA/D).
- $MNRS$: Max Number of Replaced Solutions (MOEA/D).
- NS: Neighbor Size (MOEA/D).

Table 4.6 Parameters settings considered for tuning phase.

Parameter	Possible Values
Pop	10, 50, 100, 200, 500, 1000
Cx	1-Point, 2-Point
Cx_p	0.1 - 1.0
Mx	Random, $MARS_{mo}$
Mx_p	0.1 - 1.0
Kp	0.1 - 1.0
NSP	0.1 - 1.0
$MNRS$	10, 50, 100, 200, 500, 1000
NS	10, 50, 100, 200, 500, 1000

Note also that we will be comparing the proposed custom mutation operator $MARS_{mo}$ with the classical $Swap$ mutation.

Performance evaluation of MOEA: Table.4.7 provides an overview of the different parameters considered for each experiment using irace. The first experiment will focus on a subset of objectives (O1, O2, and O3) and will be only applied on **Group 1** that concerns users having less or equal to 5 items in their viewing history. The second experiment will focus on all objectives (O1-O5) applied on **Group 2**. The fact that **Group 2** has more items in their profile is the reason for such a decision since we can

use a portion of their history as a test to calculate O4 (RMSE) and O5 (nDCG@k).

In both experiments, a fixed computing time limit of 1 hour is defined as a stopping criterion. This means that for each user, a total time of 1 hour is provided to look for the best recommendations. Concerning results, k was set to 10 items (max number of recommended items per user).

Before we can assess "how good" any single run of a Multi-Objective Evolutionary Algorithm (MOEA), we must first grasp two concepts. The first is convergence, which indicates how "close" we come to find the best solution. The second metric is diversity which measures if solutions are fully spread throughout the set or are clustered together. Elite configurations are returned based on their average best Hypervolume (HV) metric [145] across different test instances. The HV metric is capable of measuring both the convergence and diversity of our solutions. The higher the HV value, the better our solutions are.

Table 4.7 Overview of experimental configuration for parameter tuning phase.

Experiment	**1**	**2**
Number of Objectives	*3 (O1, O2 and O3)*	*5*
User Group	*Group 1*	*Group 2*
Performance Metric	*Hypervolume*	*Hypervolume*
K (Number of recommendations)	*10*	*10*
Time Limit	*1 Hour*	*1 Hour*

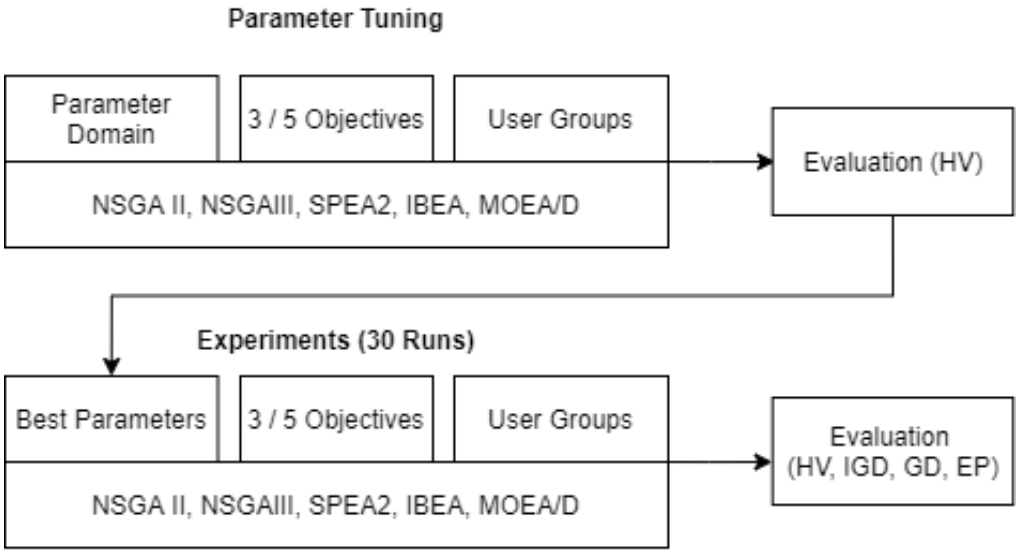

Figure 4.12 Overview of the experimental configuration of both parameter tuning and following experiments for MARS.

Figure.4.12 depicts the design of experiments starting with the parameter tuning phase (irace) and its results (best parameters). The output of irace will constitute the input for final experimentation where we compare MOEA algorithms to find the best-performing approach. Despite the similarities of both experiments, notice that for the second one, we are running a total of 30 executions and calculating multiple evaluation metrics instead of just one. The following quantitative metrics can provide an evaluation mechanism for both convergence and diversity [145]: Hypervolume (HV) (Max), Generational Distance (GD) (Min), Inverse Generational Distance (IGD) (Min), Epsilon-Indicator (ϵ) (Min). Details of each evaluation mechanism can be found in Chapter 2 Section 2.1.4.

Computational Environment We conduct the experiments on a private cloud cluster. We use 5 virtual machines (VM) each composed of 24 cores of 1 GHz and 100 GB RAM each. We installed SLURM [117] an open-source, fault-tolerant, scalable cluster management, and job scheduling system. It is responsible for allocating access to resources (compute nodes) to users, monitoring jobs, and finally, it is arbitrating resources by managing a queue of pending work.

4.4 Experimental Results

4.4.1 Results of Parameter Tuning

Starting with Table.4.8 which shows the elites configurations provided by irace for each algorithm (NSGAII, NSGAIII, SPEA2, MOEA/D, and IBEA) using implicit interactions for three objectives (O1, O2, and O3).

Table 4.8 Elites configurations provided by i-race using 3 objectives on implicit dataset (Group 1).

Parameter	NSGAII	NSGAIII	SPEA2	MOEA/D	IBEA
Pop	10	10	10	500	10
Cx	1-Point	1-Point	1-Point	2-Point	1-Point
Cx_p	0.3	1.0	0.9	0.7	0.1
Mx	Random	Random	Random	$MARS_{mo}$	Random
Mx_p	1.0	0.6	1.0	0.9	0.9
Kp	-	-	-	-	0.2
NSP	-	-	-	1.0	-
$MNRS$	-	-	-	1000	-
NS	-	-	-	500	-

Observations indicate that the $1-Point$ crossover operator has been chosen over the $2-Point$ crossover operator by most algorithms. This can be explained by the fact that the crossover operator in this test set does not impact objectives performance, so selecting a simpler operator could be the reason. Only $MOEA/D$ chose $MARS_{mo}$ as the mutation operator, while the rest of the algorithms used the random mutation operator. This can be attributed to a variety of factors, including the allowed computing time and length of solutions k aside from the number of objectives.

Table 4.9 Elites configurations provided by i-race using 5 objectives on implicit dataset (Group 2).

Parameter	NSGAII	NSGAIII	SPEA2	MOEA/D	IBEA
Pop	10	10	10	100	100
Cx	1-Point	2-Point	2-Point	2-Point	2-Point
Cx_p	0.1	0.1	0.6	0.3	0.6
Mx	$MARS_{mo}$	$MARS_{mo}$	$MARS_{mo}$	$MARS_{mo}$	$MARS_{mo}$
Mx_p	1.0	0.8	1.0	1.0	0.9
Kp	-	-	-	-	1.0
NSP	-	-	-	0.8	-
$MNRS$	-	-	-	500	-
NS	-	-	-	100	-

When looking at Table.4.9 for the 5-Objectives irace runs, most elite configurations have chosen the $2-Point$ crossover operator over $1-Point$. This confirms our previous assumption, that crossover operators, are chosen depending on their role in improving the objectives. Since the additional objectives in this experiment have an interest in item ordering, a change in elite configuration was anticipated. Similarly, all algorithms selected $MARS_{mo}$ as a mutation operator, indicating that this operator has superior performance, particularly in complex settings.

4.4.2 Performance Results of MOEAs

As indicated in Figure.4.12 the output of irace which is elite parameters will be used as an input for the final experiment. We average the error metrics of 30 algorithm executions. The methodology attempts to account for the majority of situations while also providing an overall picture of how each approach performs. In the next paragraph, we examine the results of the first experiment that handles both Group 1 of users and 3 Objectives (O1, O2, and O3).

Table 4.10 Performance comparison (**average best value** and the standard deviation) using 3 Objectives for over 30 independent runs.

Algorithm	HV_{3OBJ}	GD	IGD	EP
NSGAII	0.77 0.05	1.23 0.002	1.019 0.016	0.147 0.024
NSGAIII	**0.91** 0.04	1.24 0.001	1.018 0.018	0.101 0.025
SPEA2	0.80 0.03	1.24 0.0007	**1.002** 0.035	0.146 0.025
MOEA/D	0.68 0.07	1.23 0.013	1.076 0.059	0.233 0.042
IBEA	0.85 0.06	**1.22** 0.014	1.070 0.021	**0.093** 0.037

Results over 3 objectives for User Group 1 Table. 4.10 shows the performance comparison of each algorithm for 30 independent runs. Starting with HV_{3OBJ} column, results show that $NSGAIII$ has a maximum score of **(0.91)**. In second place $IBEA$ followed by $SPEA2$ with a score of **(0.85)** and **(0.80)** respectfully.

Taking into account other performance indicators (GD, IGD, ϵ) which must be minimized and starting with the GD column shown in Table. 4.10. Most algorithms obtained similar scores with $IBEA$ achieving the lowest score **(1.22)**. However, looking at the IGD column, $SPEA2$ was able to obtain a value of **(1.002)** and create a gap with the rest of the algorithms. Note however that both GD and IGD are easier metrics to meet compared to ϵ. For which, $IBEA$ was able to achieve the lowest ϵ score with **(0.093)** followed by both $NSGAIII$ and $SPEA2$.

Table 4.11 Performance comparison (**average best value** and the standard deviation) using 5 Objectives for over 30 independent runs.

Algorithm	HV_{5OBJ}	HV_{3OBJ}	GD	IGD	EP
NSGAII	0.74 0.05	0.84 0.034	**1.41** 0.003	**1.06** 0.03	0.07 0.04
NSGAIII	0.79 0.03	0.84 0.037	1.44 0.0006	1.07 0.03	0.10 0.08
SPEA2	**0.81** 0.06	**0.85** 0.026	1.43 0.004	1.12 0.02	**0.06** 0.01
MOEA/D	0.47 0.02	0.57 0.029	1.45 0.03	1.31 0.07	0.29 0.02
IBEA	0.68 0.04	0.74 0.018	1.46 0.01	1.17 0.01	0.15 0.0008

Results over 5 objectives for User Group 2 In Table. 4.11 we shift our focus to 5 Objectives performance results, starting with HV which is shown in the HV_{5OBJ} column. $SPEA2$ and $NSGAIII$ achieved good scores of **(0.81)** and **(0.79)** respectively.

The previous findings are compared with the results of the 3 Objectives experiment in Table. 4.10. Since objectives 1, 2, and 3 are already included, we aggregate their values. This is indicated by HV_{3OBJ} column

in Table. 4.11. Both $SPEA2$ **(0.85)** and $NSGAIII$ **(0.84)** kept a robust performance. $NSGAII$ obtained a score of **(0.84)** outperforming its previous HV score in Table. 4.10. However, $IBEA$ didn't perform well compared to the HV_{3OBJ} experiment.

$NSGAII$ obtained the lowest GD score **(1.41)** not far from other algorithms. Surprisingly though, $NSGAII$ was also able to obtain the lowest IGD score of **(1.06)** followed by $NSGAIII$. When considering the ϵ column, $SPEA2$ achieved a performance similar to $NSGAII$ with scores of **(0.06)** and **(0.07)** respectfully.

Setting aside the HV_{5OBJ} results, $NSGAII$ has shown good results considering the many-objective problem setting. But, $SPEA2$ and $NSGAIII$ continued to perform marginally better, and $NSGAII$ kept a steady performance in both experiments. This makes the previous algorithms well fit for our future experiments.

Convergence study with graphs: Moving on to discuss the evolution of the hypervolume HV indicator for each algorithm over time. We start with $3OBJ$ performance charts shown in Fig. 4.13.

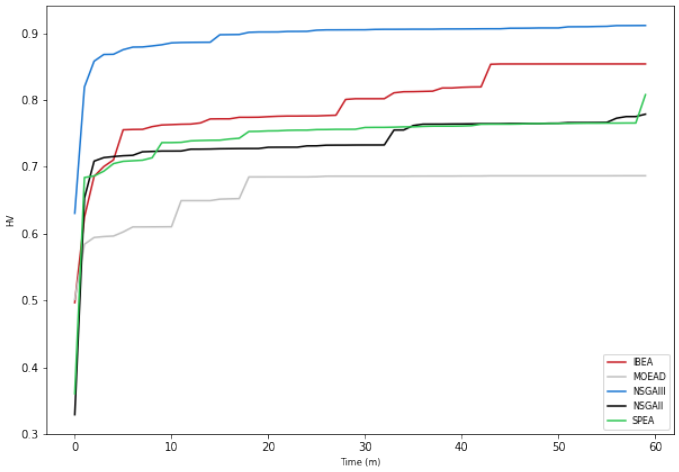

Figure 4.13 Evolution of the HV indicator (Y-axis) for $3OBJ$ over 1 hour of computing time (X-axis) using all algorithms (30 Executions).

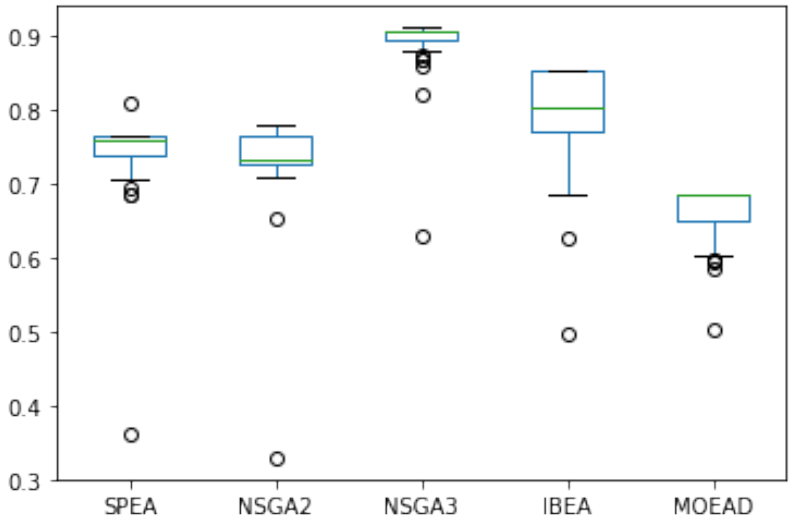

Figure 4.14 Box-plot of HV indicator for $3OBJ$ experiments using all algorithms (30 Executions).

Each algorithm $IBEA$, $MOEA/D$, $NSGAIII$, $NSGAII$, and $SPEA$ has its respective colors (Red, Grey, Blue, Black, Green). Same as our findings in Table. 4.10, $NSGAIII$, $IBEA$, and $SPEA2$ are in the lead when looking at the end of the graph. $NSGAIII$ was able to maintain its superiority from the beginning, while both $IBEA$ and $SPEA2$ lacked behind in the first third of the experiment time. Further details that show the distribution, skewness, and averages of convergence data of $3OBJ$ are illustrated in Figure.4.14.

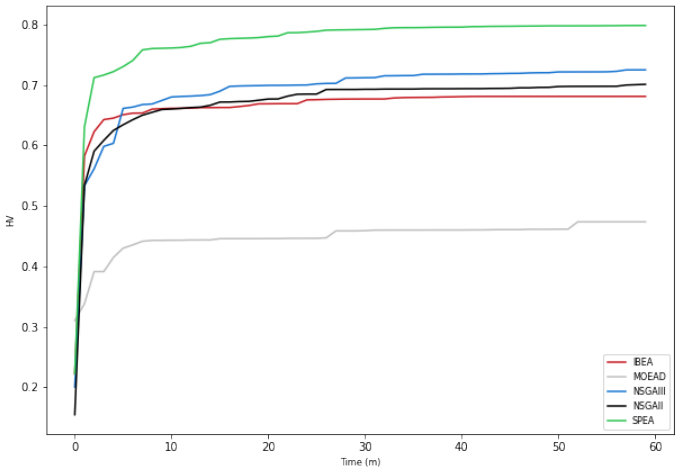

Figure 4.15 Evolution of the HV indicator (Y-axis) for $5OBJ$ over 1 hour of computing time (X-axis) using all algorithms (30 Executions).

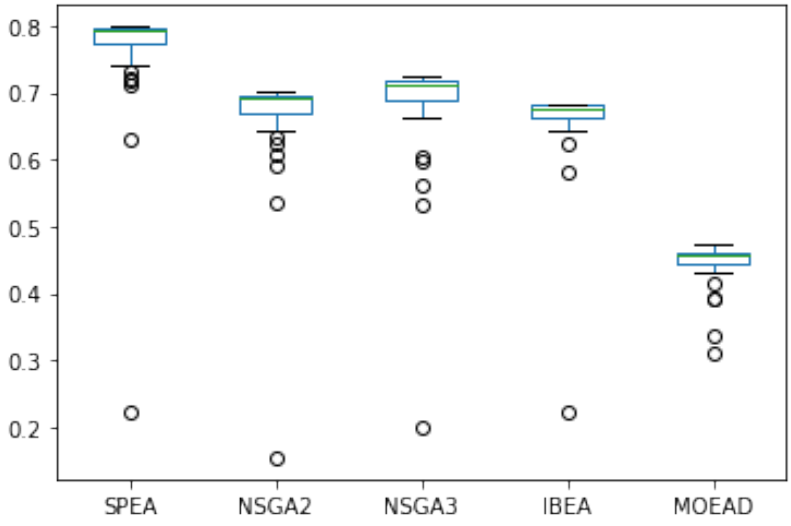

Figure 4.16 Box-plot of HV indicator for $5OBJ$ experiments using all algorithms (30 Executions).

This behavior changes when looking at the $5OBJ$ graph in Fig.4.15. $SPEA2$ keeps the lead from the start compared to other algorithms. From both experiments, we can see that most algorithms are not improving at the same level as at the beginning of the experiments. This indicates a stagnation state and algorithms aren't likely to improve considerably. A summary of convergence data for the $5OBJ$ experiment is depicted in Figure.4.16.

Examining Figure.4.13 and Figure.4.15 reveals that good results are obtained in around 5 minutes of computing time for both experiments. While most approaches continue to improve after that time, the extra computing time is not justified. The advantage of cutting training time early is the ability to quickly update models and serve recommendations to users.

Qualitative evaluation using real users Following the observations on the quality of the optimization process and metaheuristic performance, we shift focus to understanding and explaining the obtained best solutions. We randomly picked 2 users from the test data; the details of each user profile are provided below.

Starting with User A belonging to Group 1 which consists of 7223 users having less or equal to 5 items in their viewing history. User A has seen a total of 4 videos, further details are shown in Table.4.12. Note that we are only considering highly viewed items (*leq* 50% of video time) in the dataset.

Table 4.12 User A (Group 1) profile history of seen learning videos.

Title	Category	Duration (minutes)	Number of Views
What is OneDrive for business?	One Drive	0:42	1569
Share documents	One Drive	2:56	1051
How to use Outlook Online	Outlook	1:51	819
Discover Outlook Interface	Outlook	2:54	609

We explore the results returned to User A from each algorithm used in previous experiments. A reminder on User A results which were evaluated on 3 Objectives (Similarity, Novelty, and Diversity), only the first 5 items are displayed since they will also be the first seen by a user before deciding whether to click or explore the rest of the results. Details of recommendations tailored for User A are found in Table.4.13.

Table 4.13 Personalized recommendations of User A on 3 objectives experiments (including duration and views).

Algorithm	**HV**	**#1 Item**	**#2 Item**	**#3 Item**	**#4 Item**	**#5 Item**
NSGAII	**0.77**	Create Tasks (Outlook) 1:37 - 261	Share Calendar (Outlook) 1:44 - 257	Add Contacts (Outlook) 1:35 - 288	Intro Shifts (Shifts) 0:41 - 230	Interfaces in Shifts (Shifts) 0:41 - 230
NSGAIII	**0.91**	Group working (Groups) 1:59 - 320	Settings for Skype (Skype Business) 1:23 - 117	Manage Photos (Windows) 2:17 - 154	Live Event (Teams) 1:35 - 209	Categoris (Outlook) 2:22 - 285
SPEA2	**0.80**	Create E-mail (Outlook) 0:44 - 160	Forwading mails (Outlook) 0:44 - 160	Ranking (Lists) 0:51 - 242	Outlook Online (Outlook) 2:22 - 285	Tasks (Outlook) 0:44 - 160
IBEA	**0.85**	Print worksheet (Excel) 2:01 - 286	keyboard Shortcuts (Windows) 2:17 - 154	Profile (Office365) 0:43 - 142	Phone Calls (Teams) 1:35 - 845	Sahre Docs (OneDrive) 0:42 - 1569
MOEA/D	**0.68**	Filters (PowerBi) 4:41 - 15	Print Schedule (Shifts) 0:41 - 230	Interface (Bookings) 2:46 - 290	Calendarr (Bookings) 2:55 - 322	Services (Bookings) 2:03 - 283

Observations over User A profile in Table.4.12 shows interest in a small set of tools (One Drive and Outlook), one specialized in file management on the cloud, and the second in emails, scheduling, and organizational tasks. The personalized recommendations are displayed in Table.4.13 shows a diverse selection of tools that provides a range of similar services. We start with results returned by the best-performing approaches in this experiment which are NSGAIII and IBEA. While some of the items in both solutions are identical to those already present in User A's profile ("Outlook" and "One Drive"), we also discover other unexpected services ("Teams", "Skype", and "Windows"). The novel and diverse results may provide a new exploration of the catalog. It can also teach users about new services they weren't aware of or can be integrated into their workflow. This boosts user productivity. Considering that both NSGA III and IBEA had the best metrics, it's no surprise that results exhibit a nice balance between the 3 selected objectives (Similarity, Diversity, and Novelty).

For the remaining algorithms, SPEA2 and NSGAII exhibit a similar trend. Their results are balanced between similar items ("Outlook") and items with new tools that might complement the existing knowledge ("Shifts" and "Lists"). Lastly, MOEA/D had the most diverse content, as none of its suggestions are comparable to any of those in User A's profile.

Moving on to User B belonging to Group 2 which consists of 2566 users having more than 5 items in their viewing history. User B has seen a total of 7 items, further details on each item are shown in Table.4.14.

Table 4.14 User B (Group 2) profile history of seen learning videos.

Title	Category	Duration (minutes)	Number of Views
OneDrive for Business	One Drive	0:42	1569
Discovery	Share Point	2:10	379
Intro to PowerApps	Power Apps	1:12	526
Intro to Power BI	Power BI	2:52	406
Kaizala for beginners	Kaizala	1:06	193
Scheduling a team meeting	Teams	1:09	330
Appointments on Outlook	Outlook	0:57	455

We explore the results returned to User B from each algorithm used in previous experiments. User B results are calculated using 5 Objectives (Accuracy, Novelty, Diversity, RMSE, and nDCG@5), only the first 5 items are displayed since they will also be the first seen by a user before deciding whether to click or explore the rest of the results. Details of recommendations tailored for User B are found in Table.4.15.

Table 4.15 Personalized recommendations of User B on 5 objectives experiments (including duration and views).

Algorithm	**HV**	**#1 Item**	**#2 Item**	**#3 Item**	**#4 Item**	**#5 Item**
NSGAII	**0.74**	Downloads (Office365) 0:56 - 274	Evenets (Teams) 1:09 - 330	Together Mode (Teams) 1:09 - 330	Titles (PowerBI) 2:54 - 486	Document protection (Teams) 1:14 - 317
NSGAIII	**0.79**	Avg. in Math (Excel) 1:52 - 351	Document protection (OneDrive) 2:56 - 1060	Acces Rights (OneDrive) 2:02 - 928	Filters (PowerBI) 3:26 - 364	Extensions (Edge) 1:31 - 251
SPEA2	**0.81**	Intro (Office365) 0:43 - 148	Sharing (OneNote) 1:51 - 213	Intro (Lists) 1:37 - 308	Excel Lists (Lists) 1:42 - 307	Automate (Lists) 2:18 - 318
IBEA	**0.68**	Online (PowerPoint) 1:25 - 277	Fonts (PowerPoints) 1:07 - 337	Intro (PowerApps) 1:12 - 526	Pin Docs (SharePoint) 2:32 - 349	Collab. Lists (Teams) 2:42 - 344
MOEA/D	**0.47**	Installation (Office365) 2:25 - 173	Pending (Lists) 0:52 - 100	Event stats (Teams) 1:07 - 340	Pin Docs (SharePoint) 2:32 - 349	Cond. Formatting (Excel) 1:41 - 482

Observations of User B's profile in Table.4.14 displays interest in a wide range of tools without indicating any particular preference other than the aim for introductory videos. This might indicate that the user is expanding their knowledge on a suite of tools or the overall catalog. Both users A and B have different learning approaches. The personalized recommendations are displayed in Table.4.15 and show a diverse selection of tools that contain similar items to the user profile. We begin with the SPEA2 and NSGAIII results, which are the techniques that performed the best across 5 objective experiments.

In comparison to other methods, SPEA2 findings demonstrate more diversified themes (such as "Office 365" and "Microsoft Lists") and give many more introductory videos. NSGA III shows more similarity with User B's profile with little diversification compared to SPEA2. Similarities between the categories ("SharePoint," "Power-Apps," and "Teams") are seen in results returned by IBEA, with the addition of ("Power-Point") videos to add diversity. The 3rd element is present in the user profile, and this indicates that the algorithm (IBEA) managed to predict that the user will be watching the video "Intro. to Power-Apps".

Due to the fact that we train using a portion of the user profile and evaluate nDCG@5 on the remaining portion, it is possible to find identical items in both user profiles and recommendations. In comparison to SPEA2, the remaining algorithms, MOEA/D and NSGAII, display similarities to User B's profile and place less importance on introductory videos.

4.5 Deploying Model in Production

To provide results in a production environment where actual users utilize the platform, the approach will be trained on selected user profiles to avoid the extra-computing. Our goal is to optimize the hypervolume score while reducing the model computation time because this process can take a while, especially when there are a lot of users.

We learned in Section 4.4.2 that a reasonably good hypervolume score can be attained within 5 minutes of computation. This is seen in both Figure.4.13 and Figure.4.15. This means that calculations will run for a maximum of 5 minutes per user before returning recommendations. To prevent potential issues with model training or restricting access to users, recommendation update times are set to hours where there is the lowest usage rate.

In order to provide results in production scenarios, the approach will be trained on user profiles. Because this process can take a long time, especially with a large user base, we try to keep our runs as short as possible while maximizing the HV metric. Examining Figure.4.13 and 4.15 reveal that good results are obtained in around 5 minutes of computing time for both experiments. While most approaches continue to improve after that time, the extra computing time is not justified.

4.5.1 Graphical Improvements for Users

Back in Chapter 1 Section 1.1.2.3, we based our assumptions that lower explicit ratings are partly due to graphical issues since it's the only visible type of feedback input to users. We confirmed our hypothesis after investigating the graphical interface available for both registered users and visitors. In content pages the was a clear lack of rating methods. For the existing interactions (like and share buttons) there was no text explaining the function of a barely visible icon. Chen et al. [21] insist on improving visibility and readability for users, whether for recommendation results or the content they are viewing.

4.5.1.1 Ratings

Since explicit ratings are essential for both feedback information and also input to the recommender system, we propose a new design for existing and new interactions. The design is shown in Figure.4.17 where graphical changes include more readable text font and size, in addition to informative icons and button indications (on-hover effects).

Figure 4.17 Proposed design for explicit interactions.

Furthermore, a "Feedback" option is provided for users who are having difficulty with a certain item. By indicating whether there are any issues with audio, video, or subtitles, the user can have control over the state of the catalog. This also gives the company an early warning system if there was unseen errors in both technical and educational matters.

The "Bookmark" feature is another addition to explicit ratings. It enables users to save content to be viewed later. The purpose of this feature was initially to help users save videos that seemed interesting without spending additional time looking for them in other sessions. This feature is implemented and still in the early stage of collecting information before assessing its importance.

The last addition is a pop-up (dialog box) after a video has finished addressing the lack of explicit ratings related to the content. This feature aims to collect feedback from users after they have completed their learning. The proposed pop-up design is shown in Figure.4.18.

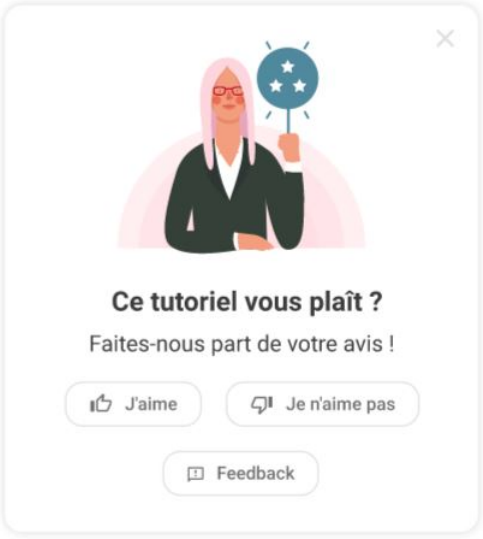

Figure 4.18 Design of a dialog box shown at the end of a video.

4.5.1.2 Recommendation Placement

Graphical improvements weren't limited to small tweaks but also major changes in the main pages (Home and Content Page). Our propositions take into consideration both registered users and visitors. Previously the home page provided a limited set of the newest courses and tutorials for both visitors and users. The current home page design is depicted in Figure.4.19. Since the goal of the main page is to provide the shortest possible path to learning we propose a new page design that offers a more efficient way to access a multitude of content types. The suggested page design is illustrated in Figure.4.20. Users/visitors can now directly select certain tools and required skills for specific jobs or certification videos. We took advantage of existing population generators to create standalone recommendations for both users and visitors. This design provides the possibility to include our popular items for visitors and personalized recommendations for registered users.

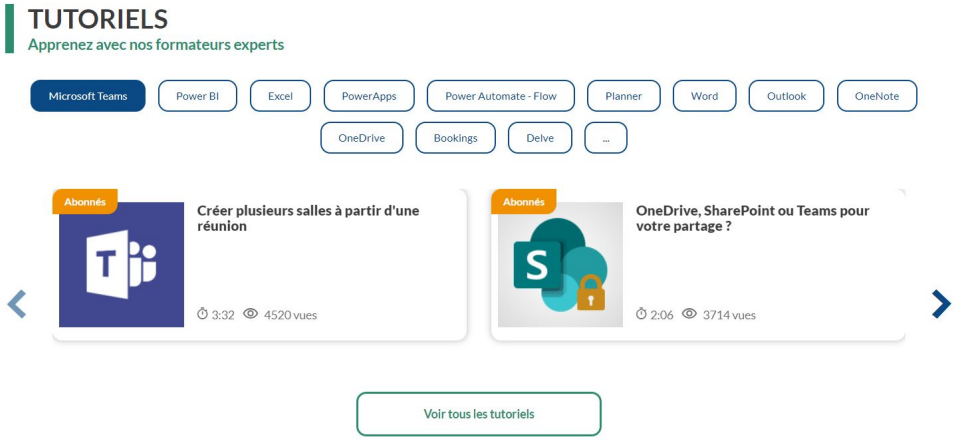

Figure 4.19 Current home page design.

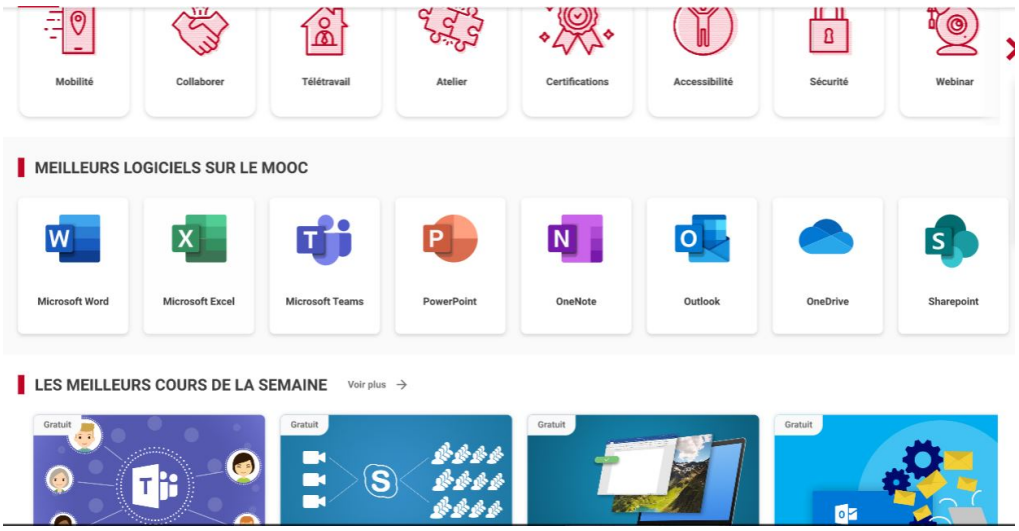

Figure 4.20 Proposed home page design.

For the content page, we improved the overall graphical interface (ratings and navigation), but most importantly we try to remove an important factor in dropout rates which is the burden of information finding. Previously users were required to go back to the content listing and search for the next video to watch which is time-consuming, this is shown in Figure.4.21. A playlist containing the next items to watch is generated using recommender systems. This playlist guides users through their learning journey that adapts to changing trends and behavior, this is shown in Figure.4.22. We discuss the integration of recommender engines in both content/home pages in detail in Section 4.5.2.

Figure 4.21 Current content page design.

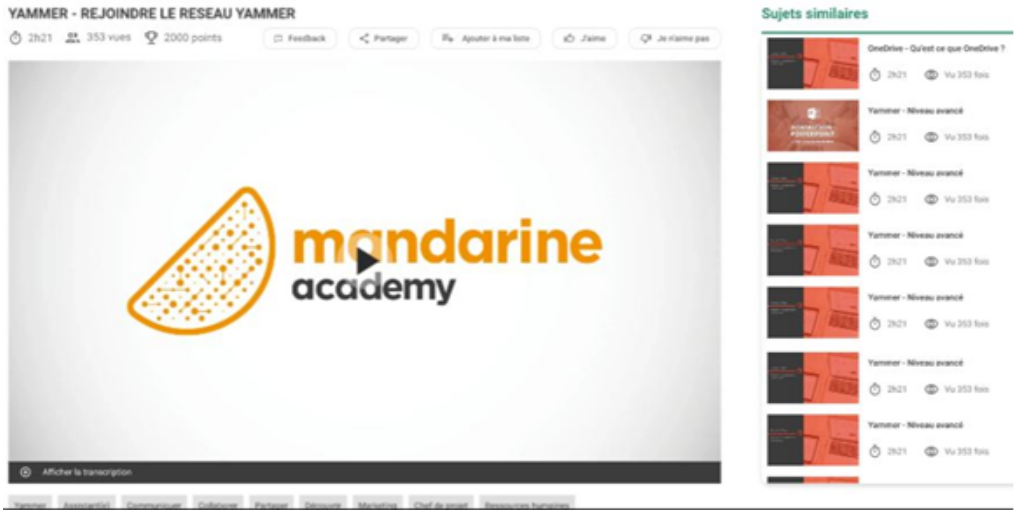

Figure 4.22 Proposed content page design.

4.5.1.3 Quality of Life

Advanced and power users that know their way well within the platform will benefit from improvements to the search experience. This is shown by giving more control over item selection through filters. We considered adding additional attributes such as "duration," "license," "quizzes," and "bonus content." To help users better tune their results. A comparison of Figure.4.23 with the old design and Figure.4.24 with the suggested design reveals that the old version of filters is less visible and does not offer as many attributes as the suggested design. More design enhancements, such as reassembling certain graphical elements (video player, item sliders, and search bar) to match common online services, were proposed. The goal was is to reduce cognitive overload caused by clumsy and unfamiliar browsing experiences, which users may encounter on occasion [130].

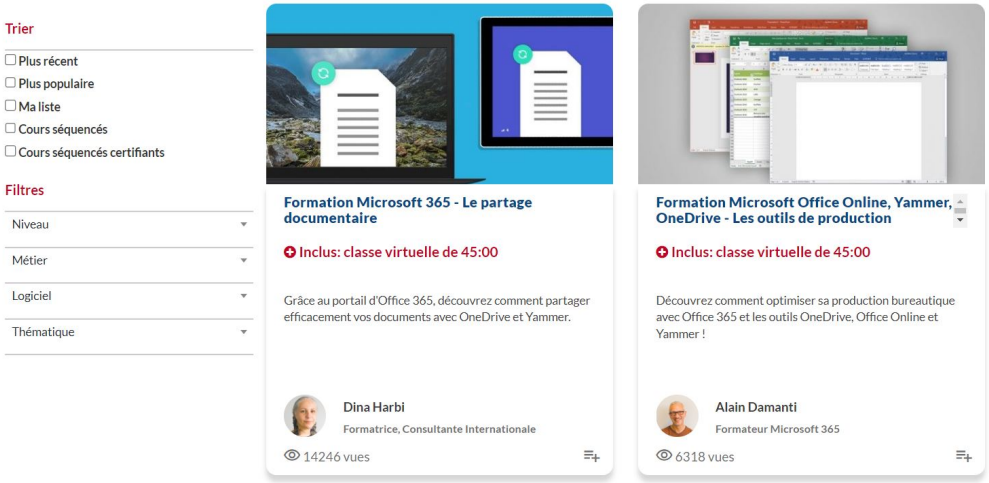

Figure 4.23 Existing filter design (courses).

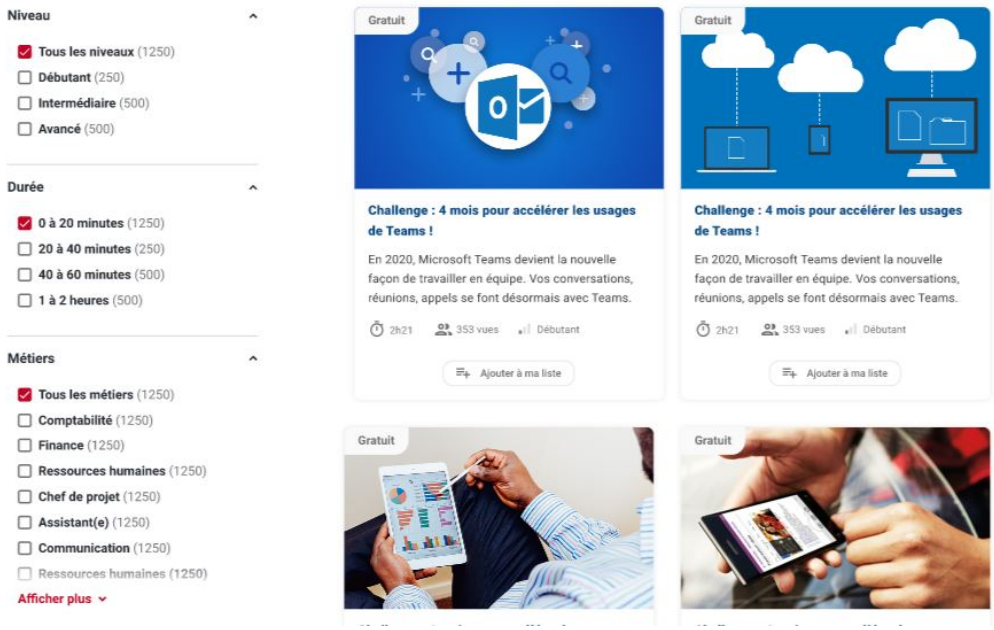

Figure 4.24 Proposed filter design (courses).

4.5.2 MARS Control Center

To manage the MARS parameters we propose the design of a custom dashboard or a control center. This control center gives the platform manager the ability to create recommendations in two modes:

1. Single-Objective: In which we select a recommendation engine and create recommendations for specific user groups and specific web pages.

2. Multi-Objective: In this mode, we use metaheuristics to provide recommendations to users on specific pages. The only difference is that this approach combines multiple recommendation engines and involves a multi-criteria decision-making process.

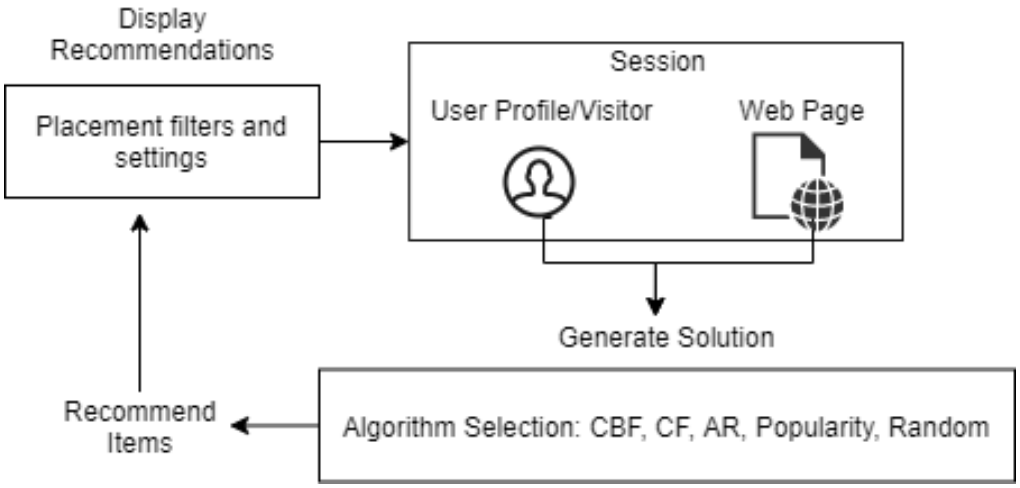

Figure 4.25 Architecture of MARS (Single-Objective).

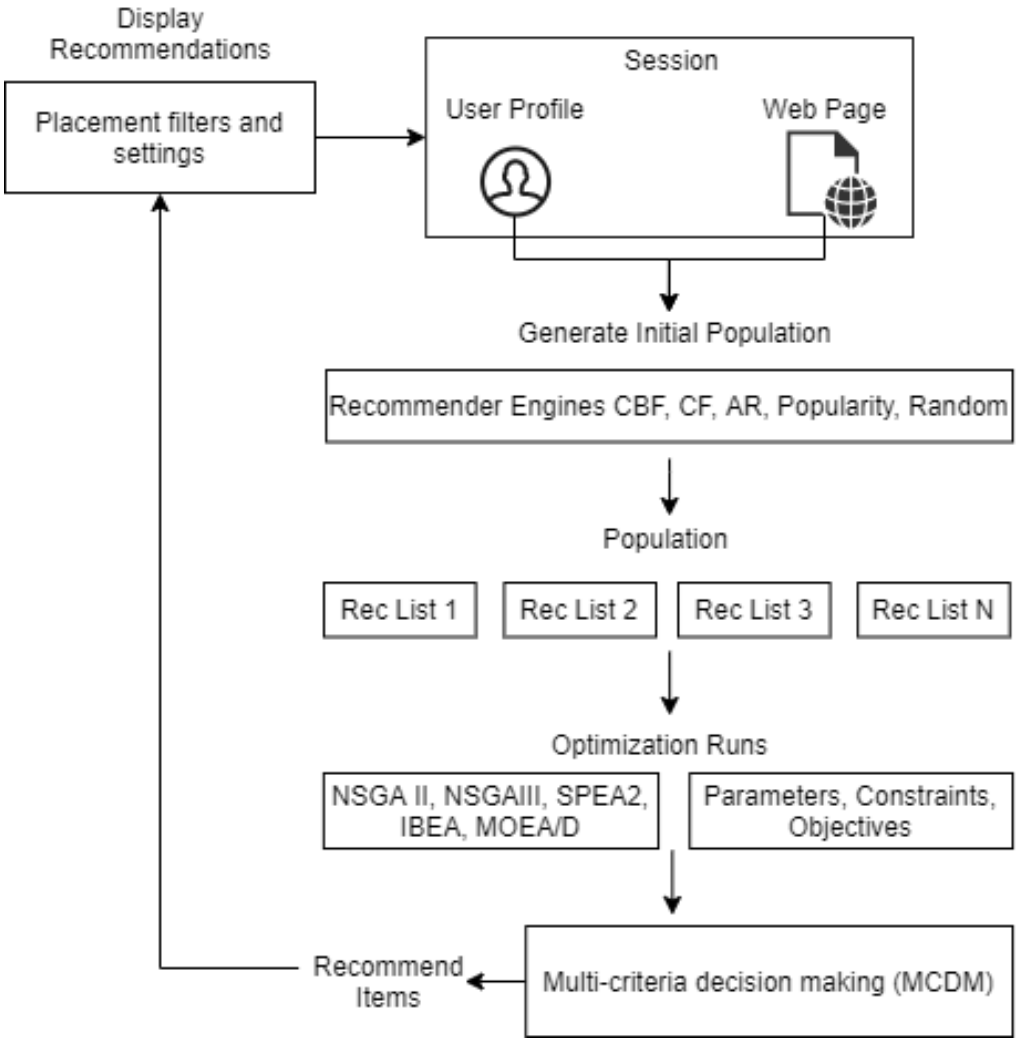

Figure 4.26 Architecture of MARS (Multi-Objective).

Both Figure.4.25 and Figure.4.26 show the architecture of both modes.

4.5.2.1 Dashboard

The MARS dashboard provides an easy way to handle multiple tasks related to using recommender systems in an E-learning platform. The following are a subset of considered features:

Model Management: This is typically the first step in creating recommendations within a service. Here managers are provided with options to select what recommendation approach to use (single or multi-objective) as well as parameters related to algorithms (objectives, constraints, user groups, evaluation, etc.).

Scenario Management: Scenarios give an extra layer of tuning results, this can be described as advanced filters such as promoting paid items over free items. Overall, here we control the behavior of displaying items.

Interaction Analysis: This interface is crucial to monitor the health of recommendations across different models, scenarios, and pages. Here we have access to interactions count (likes, dislikes, shares, etc), as well as how each model is performing (clicks, average viewing time, etc). This also includes statistics on other forms of recommendations such as mailing and push notifications.

Alerts and emails: In this section, we define the parameters of emails or push notifications, this includes what approach to apply, when to apply it, and on what user group. Using recommendations in such activities can help remind users that their actions are taken into account and build trust between their feedback and provide results.

4.5.2.2 Online Evaluation

Back in Chapter 2. Section 2.3.3 we discussed the advantages of online evaluation as it provides interactions that are less susceptible to bias because they are generated by real users using the platform. Besides, Interactions such as clicks on recommended items, sharing, or increasing the average watch time of recommended items can indicate that the model is working correctly for a specific page or specific group of users. One very known method used in the online evaluation of recommender systems is A/B testing which measures the direct impact of the recommender system on the user. AB testing campaigns are planned for both the home and content pages to assess the effects on user behavior and satisfaction levels.

4.5.2.3 Multi-Criteria Decision Making

After obtaining a set of non-dominated solutions, one may wonder how a decision-maker can choose which solution to serve. This multi-objective decision-making process is also known as Multi-Criteria Decision Making (MCDM). Pseudo-Weights is one solution among many others [33]. This is explained in detail in Chapter 2 Section 2.1.2.4.

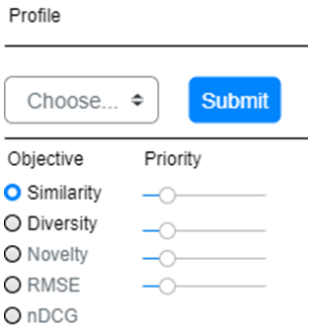

Figure 4.27 Concept design for Pseudo-Weights method applied on MARS from the decision maker perspective.

From the standpoint of the decision maker, the user will be presented with a set of already configured profiles, as well as the option to define custom profiles. The design shown in Figure.4.27 allows the decision maker to choose which objectives to focus on and to set the importance of each objective.

4.6 Conclusion

We addressed the issue of the recommender system for e-learning content in this chapter. We took into account the Mooc-Office365-Training a real-world e-learning platform operated by Mandarine Academy. We identify the main problems obstructing the platform's potential growth and adding to drop-out rates through in-depth research of historical data as well as graphical interfaces.

Challenges such as data sparsity and dropout rates are addressed throughout this chapter. Our first contribution is formulating the Mandarine Academy Recommender System (MARS) problem as a Multi-Objective Optimization Problem (MOP). To expand, 3 different stages are considered for formulation. The first stage is entity definition. We explain how each entity functions and its relationship to other entities. Understanding the domain of our problem is crucial in the functioning of a recommender system.

The second stage involves mathematically describing our business constraints and goals (objectives). This process entails translating key metrics and business requirements into mathematical forms. In total, 5 objectives are considered in this problem (Similarity, Diversity, Novelty, RMSE, and nDCG@5). By modeling how we evaluate and check the validity of a potential solution we proceed to stage 3. In this last stage, we look over the circumstances that led to selecting genetic algorithms as a candidate algorithm to handle the found problem at Mandarine Academy.

Our second contribution in this chapter is the implementation of Multi-Objective Genetic Algorithms to solve the MARS problem. This phase included selecting an appropriate encoding structure from different candidate schemes found in the literature, as well as the reasons behind the choice.

Our third contribution is the adaptation of several recommendation strategies from the literature (Content-Based, Collaborative-Based, Association Rules, and Random) to the MARS problem. This is part of the initial population generators section, where we tested several methods to generate solutions (user recommendations) that will later be improved by genetic algorithms.

Our fourth contribution, which is also the final phase of genetic modeling, is the selection of genetic operators (mutation and crossover). For crossover, we examine several operators used in similar works and reveal a potential problem that might violate the company's constraints. We ensure that a repairing function will fix invalid solutions when using genetic operators. We detail the logic behind a proposed custom mutation operator named $MARS_{mo}$, which will be compared to classic operators under different settings.

We give a complete experimental methodology with numerous steps before doing our final experiments. The initial stage in our proposed approach was to collect, clean, and transform input data (user ratings). We compared various captured user logs and conducted exploratory data analysis to determine the significance of each event type. This involves feature engineering to better capture user preferences. We discuss the selected subset of data that will be used throughout the studies after analyzing user events. Our fifth contribution was making this dataset available for future scientific studies and engineering projects. The second stage was to develop initial solutions for users, which were then used as input to the parameter tuning phase. To choose the best-performing settings for use in population generation, we compare the advantages of multiple recommender systems approaches. 7 of 10 different recommendation approaches were chosen based on their performance or capabilities (e.g., diversity of recommendations).

The application of irace to identify optimal settings for Multi-Objective Genetic Algorithms (NSGAII, NSGAIII, SPEA2, IBEA, and MOEA/D) is our sixth contribution. To give a fair performance comparison, we gave each algorithm an equal opportunity to apply its optimal parameters after training on the data.

We began our tests after putting everything together, from genetic modeling to population generation to parameter tuning. Starting with parameter tuning tests that revealed various modifications under two separate settings (3 and 5 objectives). We discovered that higher and more complex

settings (more objectives) cause significant changes in genetic operators but had no influence on other parameters (e.g., population size). The custom mutation operator $MARS_{mo}$ was chosen because it was best suited for complex situations (5 objectives). Running tests using the optimal parameters and multiple evaluation metrics (Hypervolume HV, Generational Distance GD, Inverse Generational Distance IGD, Epsilon-Indicator ($epsilon$) is the final phase in our experimental methodology.

The proposed mutation operator was selected as best suited for complex settings (5 objectives). The final step in our experimental protocol is running experiments using the best settings and different evaluation metrics (Hypervolume, Generational Distance, Inverse Generational Distance, Epsilon-Indicator (ϵ)).

Similar to parameter tuning experiments, the selection of best approaches changes when the problem settings change from 3 to 5 objectives. For 3 objective settings, both NSGA III and IBEA performed best in most of the evaluation metrics but more importantly in the hypervolume score.

When the problem settings change from 3 to 5 objectives, the selection of optimal techniques changes similarly to parameter tuning studies. Both NSGA III and IBEA performed best in most evaluation measures, but especially in the hypervolume score HV, for three objective settings. This behavior continues in complex settings (5-Objectives) with both NSGAIII and SPEA2 achieving the highest hypervolume score HV.

Further examination of the obtained recommendations from randomly selected users reveals the variety of each algorithm's solution. When looking at performance graphs, we discovered that in both studies, satisfactory results are reached in roughly 5 minutes of computation time. This greatly simplifies the process of updating models when considering production environment scenarios.

For future experiments, we consider the addition of custom genetic operators (mutation and crossover) to be compared against different objective combinations.

Furthermore, the need to improve recommendation response times necessitates testing the suggested approach in near real-time contexts, which will allow for the observation of changes in user taste as they occur and the adaptation of future recommendations to those changes. Finally, business-related objectives are being developed, with metrics such as click-through rate (CTR) already in the works.

We want to extend the proposed graphical improvements to additional pages of the E-learning platform for the corporate context. The planned User Interface (UI) enhancements include updated explicit ratings and a renewal of the browsing experience by making the user familiar with the service's operations and also by emphasizing readability, clarity, and feedback. Other considered improvements are the access to a "command cen-

ter" or an "administrative dashboard" which makes it simpler to place recommenders in the best-suited areas of each page. This dashboard will empower platform owners and users. Through the adaptability of providing recommendations, the development of multiple models per user group or web page, and finally the analysis of online performance data.

This chapter has been the subject of a publication in the Workshop of Multi-Objective Recommender Systems (MORS'22), in conjunction with the 16th ACM Conference on Recommender Systems, RecSys (2022) [65] and in the Genetic and Evolutionary Computation Conference (GECCO 2022) [56], the French Association for Operational Research and Decision Support (ROADEF) 2022 Congress [57]. The dataset used in this research was made public for benchmarking recommender systems at Harvard Dataverse [53].

Chapter 5

Conclusion

Contents

Solving real-world problems at Mandarine Academy was a real challenge as considerations for real-world conditions were taken into account. Both found problems, Professional Timetabling (MAPT) and E-learning Recommender Systems (MARS) are investigated through literature reviews and company requirements. While further improvements are still underway, both problems were solved using Multi-Objective Genetic Algorithms, and initial solutions returned by the proposed approaches have been approved by corporate experts. In this chapter, we first describe the contributions relevant to the MAPT and MARS challenges, and then we discuss perspectives for future works.

5.1 Summary of the Main Contributions

After introducing the company where we did our research, we offer a complete overview of the products operated by Mandarine Academy in this study. The context, motivation, and objectives of the thesis are discussed in Chapter 1. This chapter introduces us to the DiLeap Logistic timetabling problem and the Mooc-Office365-Training recommendation challenges. Both platforms are intended for commercial use and are already used daily by different companies and users.

Our initial contribution is to identify the different aspects relevant to both problems. This covers the initial observations directed at both platforms to comprehend the current process and how it fails to give the necessary outcomes desired by the company. We were able to identify the advantages and drawbacks of current methods as well as gain a general understanding of the problem domain. This stage demonstrated that we were dealing with a professional timetable problem rather than a standard academic one. Furthermore, we are working with educational content recommendations rather than entertainment-based suggestions (e.g., songs, movies, etc.).

Our second contribution is the establishment of a state-of-the-art review for both problems. We devoted Chapter 2 to giving further information on the specifics of each topic. Before addressing the two problems, we present an overview of the many resolution approaches employed in the literature to deal with similar situations. Because standard algorithms cannot give an optimal solution to either problem (NP-Complete), we investigate customized algorithms that quickly rule out major sections of the search process to deliver adequate results.

In Section 2.1 of Chapter 2, we provide an overview of Combinatorial Optimization (CO) techniques and the various strategies utilized to solve computationally complicated problems. Sections 2.1.2 and 2.1.3 give insight into the many algorithms considered in our approach, including how they function, their advantages, applications, and evaluation criteria.

In Section 2.2, we go through timetabling problems in greater depth, beginning by outlining each type of timetabling problem and how it differs from others. We also compare educational and professional timetabling cases, which are rarely discussed in the literature. This is accomplished by examining the shared constraints and objectives and highlighting their distinctions. Mandarine Academy's case is characterized as professional timetabling. Although this is a rare form of timetabling problems, we were able to locate a few works[52, 36] that fall into the same category. A comparison of several works that deal with educational and professional timetabling issues is offered to highlight the contribution of comparable works as well as their advantages and downsides.

We introduce recommender systems in Section 2.3. This covers a technical explanation of both goals as well as the many types of evaluation metrics utilized in the literature. We also explain how each recommendation method operates along with the advantages and disadvantages. Finally, a detailed comparison of diverse academic works [110, 132] dealing with educational and non-educational recommender systems is explored. We describe the advantages and drawbacks of each work, as well as analyze their contributions and what our suggested strategy may offer in comparison to existing efforts.

The goal of Chapter 2 is to build domain knowledge about both challenges. This gives insights into how to tackle the technical aspects of solving both the timetabling and recommendation problems at Mandarine Academy.

Beginning with Chapter 3, we will delve more into the timetabling case at Mandarine Academy. We dubbed the problem MAPT, which stands for Mandarine Academy Professional Timetabling.

We provide a summary of the problem formulation in the first part of this chapter, which includes comprehending the entities involved, corporate guidelines and objectives, and defined constraints. With over 14 different hard constraints, 5 soft constraints, and 5 objectives. The objectives under consideration are to maximize the number of scheduled trainings, minimize soft constraint violations, reduce the number of external trainers, achieve workload balance, and reduce the number of assigned classrooms. In order to tackle this Multi-Objective Combinatorial Problem, we investigated the use of Genetic Algorithms.

Section 3.2 discusses the technicalities of solving the MAPT problem using Genetic Algorithms. Overall, we suggest a direct encoding for timetable representation, this was based on similar literature works[22]. Furthermore, we suggest the application of a construction heuristic that has been shown to be superior to the present greedy technique in terms of creating initial viable schedules. A final step in this section is the definition of custom genetic operators namely $MAPT_{mo}$ for mutation, and $MAPT_{co}$ for crossover.

Section 3.3 goes through the historical dataset that was utilized in our investigations. Instead of utilizing synthetic data, we used historical planning data that the organization had previously employed. The significance of test data is that it allows us to react to unforeseen real-world events, longer planning windows, and stricter criteria. Mandarine Academy Professional Timetabling Dataset (MAPTD)[54] is the title given to this real-world data. The data is intended to assess timetabling solvers in education and other professional areas.

MAPTD was used as input for a parameter tuning phase, which is covered in depth in Section 3.3.2. This stage uses the i-race package to discover the best-performing genetic algorithm hyperparameters. Using the elite parameters in the experiment phase will offer each picked metaheuristic a fair shot against the MAPT problem. Furthermore, in this stage, we compare the proposed genetic operators ($MAPT_{mo}$ and $MAPT_{co}$) to conventional operators to see which might be advantageous in efficiently dealing with this problem. This step was rarely found in literary works. Most discussed works chose to work with either predefined configurations or copy the same parameters used in other similar works.

We addressed the experimental findings of parameter-tuning and 30-independent runs on selected algorithms NSGAII, NSGAIII, SPEA2, MOEA/D,

and IBEA in Section 3.4. Beginning with the parameter tuning phase, our suggested mutation operator $MAPT_{mo}$ outperformed the classic Swap operator. PMX was chosen as the best crossover operator since it outperformed our custom operator $MAPT_{co}$.

In terms of experimental runs, both IBEA and NSGA III produced the best results under varied conditions (3 and 5 objectives). The following metrics were used in the evaluations: Hypervolume, Generational Distance, Inverse Generational Distance, and Epsilon Indicator. Even though our technique did not perform well as the complexity rose (larger test sets, number of specified objectives), it beats the current DiLeap Logistic strategy. The construction heuristic was able to schedule on average 35%-48% of total sessions with 0 broken constraints in complex cases (a high number of sessions), while the greedy technique only managed to plan 13%-20% of all sessions, with multiple broken hard constraints.

Finally, in Section 3.5, we present a roadmap of several features that will be introduced to DiLeap Logistic in connection with the integration of our proposed approach. Future additions will include graphical interfaces that help platform administrators understand the planning process. This also helps us to analyze and comprehend the scheduling algorithm's performance through metrics number of valid sessions, used resources, and possible time conflicts. Chapter 3 has been the subject of a publication in the IEEE Congress on Evolutionary Computation (CEC 2021) [55] and in the French Association for Operational Research and Decision Support (ROADEF-2021) [100].

In Chapter 4, we address the recommender system challenges found in the company's e-learning platform. Mandarine Academy gave us access to Mooc-Office365-Training, a real-world public e-learning platform that is available in both French and English. We said in Chapter 1 that consumers are dropping out after learning a limited amount of content (on average 4 videos). This was concerning for the organization, which had made investments to improve the learning experience. In Section 4.1, we introduce the MARS issue, which stands for Mandarine Academy Recommender System. This section contains information on the many entities involved in the recommendation process. Users, educational content, and ratings are examples of these entities. We describe the link between these elements and how they are used to feed recommender systems. In comparison to MAPT, we are simply dealing with two hard constraints this time. However, we are still in a many-objective context with several conflicting optimization goals: similarity, diversity, novelty, RMSE, and nDCG@5.

Because we have multiple conflicting objectives to optimize, this issue is classified as a Multi-Objective Combinatorial Optimization Problem (MOCOP). We chose Dominance and Indicator-based techniques, which have proved effective in dealing with multi-objective situations. NSGAII,

NSGAIII, SPEA2, MOEA/D, and IBEA are the selected algorithms. After deciding which metaheuristic to use, we design an encoding structure to represent the recommendations in a form that genetic algorithms can comprehend.

Following that, we define our first population generators, which are in charge of generating user personalized recommendations. In this stage, we examine many approaches to see which one performs the best. Out of 10 algorithms, 7 are picked. These methods can be classified as collaborative filtering (item/user/model), content-based filtering, random, and association rules models. This approach is rarely found in the discussed literature works. The benefit of such a stage is the ability to give a diverse initial population of suggestions per user, which can aid in the optimization process that will be conducted later utilizing genetic algorithms. Finally, genetic operators for the MARS challenge are defined. Starting with mutation, we suggest using a custom-made mutation operator called $MARS_{mo}$ that contains business logic. $MARS_{mo}$ will be compared to a classical random mutation to determine whether it is advantageous to our problem. We chose the simpler 1 and 2-point operators for the crossover process.

The next stage in our work was to choose the appropriate dataset to work with. We examined the reliability of each user data type because the company utilizes multiple logging techniques to track user behavior on its platforms. According to our findings, employing implicit interactions, especially video view duration, was the most reliable in capturing users' interest rates. Experimental data analysis revealed that explicit evaluations are insufficient for creating effective recommendations for the majority of users. To better portray degrees of importance, the selected user ratings (implicit) are filtered using outlier detection techniques and translated into a different scale. This dataset, MARSD (Mandarine Academy Recommender System Dataset), was made public for benchmarking recommender systems at Harvard Dataverse [53].

MARSD was employed as an input to a parameter tuning phase. Details in Section 4.4 demonstrate that, rather than executing predetermined configurations, we chose to offer fair testing grounds for each Genetic Algorithm. The parameter tuning results were divided into two test settings, 3 objectives, and 5 objectives. For the 3 objectives experiments, we used a set of users who had seen at least 5 videos from the platform. The second category includes users who saw more than 5 items. The reasoning behind dividing users is that some objectives (nDCG@5 and RMSE) require a portion of viewing history to be estimated. We also discovered that adding objectives caused significant changes in genetic operators' choices but did not influence other parameters (e.g., population size). This explains why the custom mutation operator $MARS_{mo}$ was considered for 5 objective settings instead of only 3 objectives.

We examined experimental results from 30 independent runs on selected algorithms NSGAII, NSGAIII, SPEA2, MOEA/D, and IBEA in Section 4.4.2. Similarly to parameter tuning, we launched experiments with 3, 5 objectives, and two user groups. We noticed that best-method selection changes when the problem settings switch from 3 to 5 objectives. In terms of the 3 objective settings, both NSGA III and IBEA performed best in most evaluation metrics, but especially in the hypervolume and the epsilon indicator scores, which indicates how quickly the algorithms converge to finding the best feasible solutions. SPEA 2 was able to get superior results for the 5-objective experiments, followed by NSGA III, and NSGA II converging faster.

When we examine the convergence graphs, we observe that for both experiments, satisfactory results are attained in roughly 5 minutes of processing time. While most techniques improved after that point, this time period appeared to be adequate for production mode settings. Allowing the platform managers to update and deliver recommendations faster.

Evaluations of running optimization algorithms may not accurately represent the quality of the proposed items. To validate the final solutions, we chose two random users from different user groups, listed their most recent videos, and compared results from each genetic algorithm we employed in our experiment. The results were compared based on the category of seen items, their titles, and duration. Overall, the best-chosen techniques for both 3 (NSGA III and IBEA) and 5 (SPEA2 and NSGA III) objectives were able to balance suggestions between similarity to the user profile and offering diverse content. Other algorithms, either focused on a particular goal (e.g., diverse content exclusively) or did not fairly balanced recommended content.

Finally, we explore production mode considerations in Section 4.5. We were able to successfully develop, experiment, and test recommendation techniques using real-world data. The next step is enabling the models to serve recommendations in real-world environments. However, non-expert users may find it difficult to configure the suggested approach. Platform administrators must handle tasks like setting objectives, targeting specific user groups, or algorithm parameters in a user-friendly way.

For this reason, we discuss the considered features to be implemented in future updates for the MARS system. Proposals include a dashboard that allows users to easily create individual recommendation models, tweak their parameters and objectives, and choose a target web page or user category. Finally, Multi-Criteria Decision Making (MCDM) methodologies with graphical features for describing preferences are used to choose which final solutions to display for users. Making graphical pages that can be used to administer such a system would allow for easy monitoring of the performance of suggestions in real-time. Platform administrators will observe the

benefits of adopting particular approaches on certain placements across the platform by tracking rating interactions (e.g., recommended items clicks, shares, and view times).

Finally, whether or not recommendations are relevant is up to the end user, but presenting recommendations at the correct time and place may build trust and engagement. Making recommendations is one thing; displaying them is quite another. The present graphical condition of the Mooc-Office365-Training falls short in several areas, including ambiguous explicit ratings, restricted resource exploration, and limited search mechanisms. We presented many web page designs to improve and correct the existing user interface. Propositions include better visibility for explicit ratings, recommended items, and additional filtering features for selecting relevant items.

In addition, feedback and save-for-later buttons are currently available on the platform. Improving the visibility of rating collectors and creating a safe atmosphere for learners to openly communicate their opinions can help reduce the scarcity of explicit interactions. Different recommendation tactics, such as "popularity" models are used to alleviate the cold start problem and draw users to initiate learning on the platform. Personalized recommendation algorithms are also considered for implementation in various parts of the platform (Home page and content pages).

5.2 Perspectives

The thesis contributions pave the way for several future studies on professional timetabling and e-learning recommendations. This section summarizes the ongoing work addressed in Chapters 3 and 4, as well as the future considerations we intend to investigate for future research.

5.2.1 Considered Perspectives for the Timetabling Problem

Our first perspective concerns the timetabling problem in DiLeap Logistic. We have seen in Chapter 2, Section 2.2.2, multiple works [98, 8] considered using a random approach instead of a construction heuristic. Even though a random approach may provide invalid solutions, however in terms of performance, the random approach would be quicker than the construction heuristic. Given that we already have a repairing function that is responsible for fixing invalid solutions after a crossover or mutation process, testing the metaheuristic with a random approach for generating initial solutions may be a viable option given the proposed approach's limitations, which include a longer computing time than the greedy approach. The entire pro-

cessing time might be greatly lowered by supplying initial solutions using a random method and doing both repair and optimization using metaheuristics.

Our second perspective concerns the experimental protocol used to solve the timetabling problem. The preliminary experiments revealed that we are still constrained in terms of performance. This was especially evident when working with larger test sets. We believe that the chosen hyperparameters have an impact on the performance of algorithms in complex scenarios. A significant upgrade to the experimental protocol is being studied, which includes executing an extended parameter tuning phase that comprises a more diversified combination of objectives, test data, and execution timeframes. Essentially, we should not limit ourselves to a single test size or two sets of objectives. By exploring different use cases, we could see changes concerning the choice of parameters such as genetic operators. We anticipate that newly discovered hyperparameters will perform better, particularly in difficult scenarios. Other minor enhancements for future experimentation include the addition of new custom or classical genetic operators (e.g., Cycle Crossover, Order Crossover, Scramble Mutation), as well as additional metaheuristics such as Ant Colony Optimization (ACO) or Particle Swarm Optimization (PSO) (PSO).

Our third perspective concerns the dataset used in the timetabling problem. The historical real-world test data is published online under the title Mandarine Academy Professional Timetabling Dataset (MAPTD)[54]. We consider updating this dataset with newer instances to provide a newer reliable dataset for benchmarking timetabling problems, whether they are educational or professional. Both test instances and training information are provided to serve as input data for different timetabling scenarios. MAPTD is much more adapted to real-life use cases, for example, the timetabling of remote trainings due to the Covid-19 pandemic is a special use case of timetabling under urgent requirements. Keeping this dataset up to date can give more rich use cases for future studies on timetabling issues.

Our fourth perspective focuses on the graphical interfaces designed for DiLeap Logistic. In Chapter 3, Section 3.5.1, we discussed the different pages examined for timetable management. While the problem input page and analytical dashboards are already in place, other pages such as problem settings that encompass several stages are now a top priority. This page will allow controlling the behavior of the approaches and provide planners the freedom to optimize the schedule depending on their requirements. Because this is a missing feature in the present greedy approach, it seems reasonable to prioritize its implementation in order to boost user acceptance of the new approach and reduce wasted time spent manually adjusting invalid solutions offered by the greedy approach in DiLeap Logistic.

5.2.2 Considered Perspectives for the Recommendation Problem

Our fifth perspective concentrates on the recommendation problem in MoocOffice365-Training. Even though existing solutions are acknowledged by domain experts. We are thinking about ways to improve our experimental methodology. This covers the initial population generators at the first stage. More model-based collaborative filtering strategies, such as Non-negative Matrix Factorization (NMF) and SlopeOne, are being considered. In addition to popularity-based methods that are already in production mode to supply visitors with potentially interesting items. For the content-based approach, we consider testing another text vectorization technique and comparing performance to TF-IDF. Possible techniques include topic discovery models such as Latent Dirichlet allocation (LDA) [14], word embeddings such as Word2Vec [111], and deep learning models such as BERT [124]. One last consideration for a content-based method is parameter tweaking of the previously outlined possible models, which might include assessing different distance measures for text similarity (e.g., Jaccard, Euclidean, Cosine). Finally, testing with different feature weights is also considered.

One significant adjustment considered in the initial population generation is assessing the impact of every single technique. The purpose of this analysis is to see how different techniques affect the time it takes for the metaheuristic to provide good solutions. Our hypothesis is based on the fact that certain strategies can have a greater impact on prediction quality than others. To validate these assumptions, we consider comparing the effects of selecting different recommendation approaches on metaheuristic performance and recommendation quality. Similar to experiments considered for the timetabling problem discussed above, we are interested in including more genetic operators, custom or classical (e.g., Cycle Crossover, Order Crossover, Scramble Mutation), with additional metaheuristics such as Ant Colony Optimization (ACO) or Particle Swarm Optimization (PSO) into future experiments.

Our sixth perspective concerns business metrics implementation in the MARS system. By including business objectives such as Click-Through Rates (CTR), Conversion/Adoption Rates, and User Engagement (customer retention). Company managers will be able to have an overview of the performance of recommendations using metrics that are easy for marketing and management teams to understand. However, such measurements need real-time capabilities to receive data and offer adaptive outcomes. Because the platform has a high volume of learners on a regular basis, recommender approaches are currently trained offline and predictions are served to users via Application Programming Interface (API) requests.

This strategy has certain limitations, such as not accounting for changes in user actions that occur only after the model has been changed. While the existing technique efficiently guides users to appropriate information, it does not account for rapid changes in the learning experience and cannot respond to that scenario only after updates.

Collecting real-time user behaviors and delivering customized recommendations based on the present learning context is considered in future major updates. This includes suggested interactions such as bookmark and feedback buttons, as well as current refurbished explicit ratings such as like and share buttons. With the present strategy in place, explicit rating adoption is likely to develop faster than previously, taking into account the new planned User Experience (UX). Future work will include testing our technique under other interaction types. Observing the impacts of specific interactions on overall recommendation quality is taken into account, especially as the collaborative filtering approaches utilized in our study were designed to perform on explicit data rather than implicit data. We want to evaluate user adoption rates for explicit rates before and after adopting graphical changes in future research.

Our seventh perspective concerns the dataset used to benchmark MARS. As indicated in Chapter 4, Section 4.3.1, the dataset[53] was uploaded on the Harvard Dataverse platform. The MARS dataset is useful for testing recommender systems since it includes both interaction types for the same problem. Other considerations include the consequences of the COVID-19 pandemic on e-learning systems, where we may see differences in user behavior before and after the lockdown periods. Such data can provide insights into user learning behavior, observable learning patterns, and, most significantly, knowledge of overall changes in the workplace-adopted tools. For future work, we intend to keep updating the data set with additional explicit and implicit rating data and also provide support for those who are interested in using the data. We aspire that this dataset will benefit the recommender systems research community in particular, as well as the machine learning research community in general.

Bibliography

[1] Emile Aarts, Emile HL Aarts, and Jan Karel Lenstra. *Local search in combinatorial optimization*. Princeton University Press, 2003.

[2] B Adenso-Dıaz, M Oliva González, and P González-Torre. On-line timetable re-scheduling in regional train services. *Transportation Research Part B: Methodological*, 33(6):387–398, 1999.

[3] Charu C Aggarwal et al. *Recommender systems*, volume 1. Springer, 2016.

[4] Bushra Alhijawi and Yousef Kilani. A collaborative filtering recommender system using genetic algorithm. *Information Processing & Management*, 57(6):102310, 2020.

[5] Bushra Jawad Mohammad Alhijawi. *The use of the genetic algorithms in the recommender systems*. PhD thesis, Ph. D. Dissertation, Hashemite University, 2017.

[6] Oras Baker and Qing Yuan. Machine learning: Factorization machines and normalized discounted cumulative gain for tourism recommender system optimisation. In *2021 IEEE International Conference on Computing (ICOCO)*, pages 31–36. IEEE, 2021.

[7] Seyed Saber Banihashemian and Fazlollah Adibnia. A novel robust soft-computed range-free localization algorithm against malicious anchor nodes. *Cognitive Computation*, 13(4):992–1007, 2021.

[8] Ruggero Bellio, Sara Ceschia, Luca Di Gaspero, Andrea Schaerf, and Tommaso Urli. Feature-based tuning of simulated annealing applied to the curriculum-based course timetabling problem. *Computers & Operations Research*, 65:83–92, 2016.

[9] Richard E Bellman and Stuart E Dreyfus. *Applied dynamic programming*. Princeton university press, 2015.

[10] Antonio Benitez-Hidalgo, Antonio J Nebro, Jose Garcia-Nieto, Izaskun Oregi, and Javier Del Ser. jmetalpy: A python framework for

multi-objective optimization with metaheuristics. *Swarm and Evolutionary Computation*, 51:100598, 2019.

[11] Mauro Birattari, Zhi Yuan, Prasanna Balaprakash, and Thomas Stützle. F-race and iterated f-race: An overview. *Experimental methods for the analysis of optimization algorithms*, pages 311–336, 2010.

[12] J. Blank and K. Deb. pymoo: Multi-objective optimization in python. *IEEE Access*, 8:89497–89509, 2020.

[13] Julian Blank. pymoo - multi-criteria decision making (mcdm).

[14] David M. Blei, Andrew Y. Ng, and Michael I. Jordan. Latent dirichlet allocation. *Journal of Machine Learning Research*, 3(Jan):993–1022, 2003.

[15] Eric Bonabeau, Guy Theraulaz, Marco Dorigo, Guy Theraulaz, Directeur de Recherches Du Fnrs Marco, et al. *Swarm intelligence: from natural to artificial systems*, volume 1. Oxford university press, 1999.

[16] Zlatko Bratković, Tomislav Herman, Vjera Omrčen, Marko Čupić, and Domagoj Jakobović. University course timetabling with genetic algorithm: A laboratory excercises case study. In *European Conference on Evolutionary Computation in Combinatorial Optimization*, pages 240–251. Springer, 2009.

[17] Edmund K Burke, Graham Kendall, and Eric Soubeiga. A tabu-search hyperheuristic for timetabling and rostering. *Journal of heuristics*, 9(6):451–470, 2003.

[18] Edmund K Burke, Barry McCollum, Amnon Meisels, Sanja Petrovic, and Rong Qu. A graph-based hyper-heuristic for educational timetabling problems. *European Journal of Operational Research*, 176(1):177–192, 2007.

[19] Robin Burke. Hybrid recommender systems: Survey and experiments. *User modeling and user-adapted interaction*, 12(4):331–370, 2002.

[20] Zheng-Yi Chai, Ya-Lun Li, Ya-Min Han, and Si-Feng Zhu. Recommendation system based on singular value decomposition and multi-objective immune optimization. *IEEE Access*, 7:6060–6071, 2018.

[21] Li Chen and Ho Keung Tsoi. Users' decision behavior in recommender interfaces: Impact of layout design. In *RecSys' 11 Workshop on Human Decision Making in Recommender Systems*, 2011.

[22] Runwei Cheng, Mitsuo Gen, and Yasuhiro Tsujimura. A tutorial survey of job-shop scheduling problems using genetic algorithms—i. representation. *Computers & industrial engineering*, 30(4):983–997, 1996.

[23] Wutthipong Chinnasri, Soradech Krootjohn, and Nidapan Sureerattanan. Performance comparison of genetic algorithm's crossover operators on university course timetabling problem. In *2012 8th International Conference on Computing Technology and Information Management (NCM and ICNIT)*, volume 2, pages 781–786. IEEE, 2012.

[24] Carlos A Coello Coello, Clarisse Dhaenens, and Laetitia Jourdan. Multi-objective combinatorial optimization: Problematic and context. In *Advances in multi-objective nature inspired computing*, pages 1–21. Springer, 2010.

[25] Alberto Colorni, Marco Dorigo, Vittorio Maniezzo, et al. An investigation of some properties of an" ant algorithm". In *Ppsn*, volume 92, 1992.

[26] William Cook. Computing in combinatorial optimization. In *Computing and Software Science*, pages 27–47. Springer, 2019.

[27] Oliver Czibula, Hanyu Gu, Aaron Russell, and Yakov Zinder. A multi-stage ip-based heuristic for class timetabling and trainer rostering. *Annals of Operations Research*, 252(2):305–333, 2017.

[28] Edjalma Queiroz Da Silva, Celso G Camilo-Junior, Luiz Mario L Pascoal, and Thierson C Rosa. An evolutionary approach for combining results of recommender systems techniques based on collaborative filtering. *Expert Systems with Applications*, 53:204–218, 2016.

[29] Charles Darwin. On the origin of species, 1859, 2016.

[30] Gabriele de Luca. Roulette selection in genetic algorithms, 2022.

[31] Olivier L De Weck. Multiobjective optimization: History and promise. In *Invited Keynote Paper, GL2-2, The Third China-Japan-Korea Joint Symposium on Optimization of Structural and Mechanical Systems, Kanazawa, Japan*, volume 2, page 34, 2004.

[32] Kalyanmoy Deb. A population-based algorithm-generator for real-parameter optimization. *Soft Computing*, 9(4):236–253, 2005.

[33] Kalyanmoy Deb. Multi-objective optimisation using evolutionary algorithms: an introduction. In *Multi-objective evolutionary optimisation for product design and manufacturing*, pages 3–34. Springer, 2011.

[34] Kalyanmoy Deb, Samir Agrawal, Amrit Pratap, and Tanaka Meyarivan. A fast elitist non-dominated sorting genetic algorithm for multi-objective optimization: Nsga-ii. In *International conference on parallel problem solving from nature*, pages 849–858. Springer, 2000.

[35] Kalyanmoy Deb and Himanshu Jain. An evolutionary many-objective optimization algorithm using reference-point-based nondominated sorting approach, part i: solving problems with box constraints. *IEEE transactions on evolutionary computation*, 18(4):577–601, 2013.

[36] Ulrich Derigs and Olaf Jenal. A ga-based decision support system for professional course scheduling at ford service organisation. *OR Spectrum*, 27(1):147–162, 2005.

[37] Pedro A Diaz-Gomez and Dean F Hougen. Initial population for genetic algorithms: A metric approach. In *Gem*, pages 43–49. Citeseer, 2007.

[38] Russell Eberhart and James Kennedy. A new optimizer using particle swarm theory. In *MHS'95. Proceedings of the Sixth International Symposium on Micro Machine and Human Science*, pages 39–43. Ieee, 1995.

[39] Omayma El Majdoubi, Farah Abdoun, Najat Rafalia, and Otman Abdoun. Artificial intelligence approach for multi-objective design optimization of composite structures: parallel genetic immigration. *International Journal*, 9(3), 2020.

[40] Andries P Engelbrecht. Fitness function evaluations: A fair stopping condition? In *2014 IEEE Symposium on Swarm Intelligence*, pages 1–8. IEEE, 2014.

[41] Aurora Esteban, Amelia Zafra, and Cristóbal Romero. A hybrid multi-criteria approach using a genetic algorithm for recommending courses to university students. *International Educational Data Mining Society*, 2018.

[42] Aurora Esteban, Amelia Zafra, and Cristóbal Romero. Helping university students to choose elective courses by using a hybrid multi-criteria recommendation system with genetic optimization. *Knowledge-Based Systems*, 194:105385, 2020.

[43] Richard David Evans, Eduardo Ahumada-Tello, and Joseph Zammit. Yammer: Investigating its impact on employee knowledge sharing during product development. In *2017 IEEE Technology & Engineering Management Conference (TEMSCON)*, pages 409–414. IEEE, 2017.

[44] Kim Falk. *Practical recommender systems*. Simon and Schuster, 2019.

[45] Reinaldo Silva Fortes, Daniel Xavier de Sousa, Dayanne G Coelho, Anisio M Lacerda, and Marcos A Gonçalves. Individualized extreme dominance (inded): A new preference-based method for multi-objective recommender systems. *Information Sciences*, 572:558–573, 2021.

[46] Felix-Antoine Fortin, Francois-Michel De Rainville, Marc-Andre Gardner Gardner, Marc Parizeau, and Christian Gagne. Deap: Evolutionary algorithms made easy. *The Journal of Machine Learning Research*, 13(1):2171–2175, jul 2012.

[47] Ibtissem Gasmi, Fouzia Anguel, Hassina Seridi-Bouchelaghem, and Nabiha Azizi. Context-aware based evolutionary collaborative filtering algorithm. In *International Symposium on Modelling and Implementation of Complex Systems*, pages 217–232. Springer, 2020.

[48] Fred Glover. *Tabu search fundamentals and uses*. Graduate School of Business, University of Colorado Boulder, 1995.

[49] Say Leng Goh, Graham Kendall, and Nasser R Sabar. Improved local search approaches to solve the post enrolment course timetabling problem. *European Journal of Operational Research*, 261(1):17–29, 2017.

[50] Ken Goldberg, Theresa Roeder, Dhruv Gupta, and Chris Perkins. Eigentaste: A constant time collaborative filtering algorithm. *information retrieval*, 4(2):133–151, 2001.

[51] Maoguo Gong, Fang Liu, Wei Zhang, Licheng Jiao, and Qingfu Zhang. Interactive moea/d for multi-objective decision making. In *Proceedings of the 13th annual conference on Genetic and evolutionary computation*, pages 721–728, 2011.

[52] Knut Haase, Jörg Latteier, and Andreas Schirmer. The course scheduling problem at lufthansa technical training. *European Journal of Operational Research*, 110(3):441–456, 1998.

[53] Mounir Hafsa. E-learning Recommender System Dataset, 2022.

[54] Mounir Hafsa. Mandarine Academy Professional Timetabling Dataset, 2022.

[55] Mounir Hafsa, Pamela Wattebled, Julie Jacques, and Laetitia Jourdan. A multi-objective evolutionary approach to professional course timetabling: A real-world case study. In *2021 IEEE Congress on Evolutionary Computation (CEC)*, pages 997–1004. IEEE, 2021.

[56] Mounir Hafsa, Pamela Wattebled, Julie Jacques, and Laetitia Jourdan. Multi-objective recommender system for corporate mooc. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, GECCO '22, page 2314–2317, New York, NY, USA, 2022. Association for Computing Machinery.

[57] Mounir Hafsa, Pamela Wattebled, Julie Jacques, and Laetitia Jourdan. Optimisation Multi-Objectif pour la Recommandation : application a une MOOC d'entreprise. In *23eme congres annuel de la Societe Francaise de Recherche Operationnelle et d'Aide a la Decision*, Villeurbanne - Lyon, France, February 2022. INSA Lyon.

[58] Pierre Hansen and Nenad Mladenović. Variable neighborhood search. In *Handbook of metaheuristics*, pages 145–184. Springer, 2003.

[59] Pierre Hansen, Nenad Mladenović, and José A Moreno Pérez. Variable neighbourhood search: methods and applications. *Annals of Operations Research*, 175(1):367–407, 2010.

[60] F Maxwell Harper and Joseph A Konstan. The movielens datasets: History and context. *Acm transactions on interactive intelligent systems (tiis)*, 5(4):1–19, 2015.

[61] Jonathan L Herlocker, Joseph A Konstan, Loren G Terveen, and John T Riedl. Evaluating collaborative filtering recommender systems. *ACM Transactions on Information Systems (TOIS)*, 22(1):5–53, 2004.

[62] Alain Hertz and Marino Widmer. Guidelines for the use of metaheuristics in combinatorial optimization. *European Journal of Operational Research*, 151(2):247–252, 2003.

[63] John H Holland. Genetic algorithms and the optimal allocation of trials. *SIAM Journal on Computing*, 2(2):88–105, 1973.

[64] Matthew Honnibal, Ines Montani, Sofie Van Landeghem, and Boyd Adriane. spaCy: Industrial-strength Natural Language Processing in Python. 2020.

[65] Holger Hoos, Laetitia Jourdan, Marie-eléonore Kessaci, Thomas Stützle, and Nadarajen Veerapen. A Multi-Objective E-learning Recommender System at Mandarine Academy *. *International Transactions in Operational Research*, 29(5):2735–2736, September 2022.

[66] Yifan Hu, Yehuda Koren, and Chris Volinsky. Collaborative filtering for implicit feedback datasets. In *2008 Eighth IEEE International Conference on Data Mining*, pages 263–272, 2008.

[67] Li Huang, Yi-feng Yang, and Lei Wang. Recommender engine for continuous-time quantum monte carlo methods. *Physical Review E*, 95(3):031301, 2017.

[68] Nicolas Hug. Surprise: A python library for recommender systems. *Journal of Open Source Software*, 5(52):2174, 2020.

[69] Akihiro Inokuchi, Takashi Washio, and Hiroshi Motoda. An apriori-based algorithm for mining frequent substructures from graph data. In *European conference on principles of data mining and knowledge discovery*, pages 13–23. Springer, 2000.

[70] Rizwana Irfan, Osman Khalid, Muhammad Usman Shahid Khan, Camelia Chira, Rajiv Ranjan, Fan Zhang, Samee U Khan, Bharadwaj Veeravalli, Keqin Li, and Albert Y Zomaya. Mobicontext: A context-aware cloud-based venue recommendation framework. *IEEE transactions on cloud computing*, 5(4):712–724, 2015.

[71] Hisao Ishibuchi, Ryo Imada, Yu Setoguchi, and Yusuke Nojima. Performance comparison of nsga-ii and nsga-iii on various many-objective test problems. In *2016 IEEE Congress on Evolutionary Computation (CEC)*, pages 3045–3052. IEEE, 2016.

[72] Kamal Jain and Vijay V Vazirani. An approximation algorithm for the fault tolerant metric facility location problem. *Algorithmica*, 38(3):433–439, 2004.

[73] Mouna Jamom, Masri Ayob, and Mohammed Hadwan. A greedy constructive approach for nurse rostering problem. In *2011 3rd Conference on Data Mining and Optimization (DMO)*, pages 227–231. IEEE, 2011.

[74] Dietmar Jannach, Markus Zanker, Alexander Felfernig, and Gerhard Friedrich. *Recommender systems: an introduction*. Cambridge University Press, 2010.

[75] Arush Jasuja. Feature selection using diploid genetic algorithm. *Annals of Data Science*, 7(1):33–43, 2020.

[76] Jyun-Yu Jiang, Patrick H Chen, Cho-Jui Hsieh, and Wei Wang. Clustering and constructing user coresets to accelerate large-scale top-k recommender systems. In *Proceedings of The Web Conference 2020*, pages 2177–2187, 2020.

[77] Michael Jugovac, Dietmar Jannach, and Lukas Lerche. Efficient optimization of multiple recommendation quality factors according to individual user tendencies. *Expert Systems with Applications*, 81:321–331, 2017.

[78] Marcela Juliani and Wellison Gomes. Influence of limit states on the optimization of reinforced concrete plane frames. 01 2017.

[79] Frank Kane. *Building Recommender Systems with Machine Learning and AI: Help people discover new products and content with deep learning, neural networks, and machine learning recommendations.* Independently published, 2018.

[80] Joseph Kasprzyk. Moea performance metrics, 2013.

[81] Rahul Katarya. Movie recommender system with metaheuristic artificial bee. *Neural Computing and Applications*, 30(6):1983–1990, 2018.

[82] Massimiliano Kaucic, Mojtaba Moradi, and Mohmmad Mirzazadeh. Portfolio optimization by improved nsga-ii and spea 2 based on different risk measures. *Financial Innovation*, 5(1):1–28, 2019.

[83] Scott Kirkpatrick, C Daniel Gelatt, and Mario P Vecchi. Optimization by simulated annealing. *science*, 220(4598):671–680, 1983.

[84] Martin Klammer, J Nikolaj Dybowski, Daniel Hoffmann, and Christoph Schaab. Pareto optimization identifies diverse set of phosphorylation signatures predicting response to treatment with dasatinib. *PLoS one*, 10(6):e0128542, 2015.

[85] Yehuda Koren. Factorization meets the neighborhood: a multifaceted collaborative filtering model. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 426–434, 2008.

[86] Yehuda Koren, Robert Bell, and Chris Volinsky. Matrix factorization techniques for recommender systems. *Computer*, 42(8):30–37, 2009.

[87] Unni Koroth. The np complete problems-why we have self driving cars, colonization plans for mars, but no true automatic timetable generators in our schools.

[88] Daniel Lemire and Anna Maclachlan. Slope one predictors for online rating-based collaborative filtering. In *Proceedings of the 2005 SIAM International Conference on Data Mining*, pages 471–475. SIAM, 2005.

[89] Rhydian Lewis. A survey of metaheuristic-based techniques for university timetabling problems. *OR spectrum*, 30(1):167–190, 2008.

[90] Hao-ran Li, Fa-zhi He, and Xiao-hu Yan. Ibea-svm: an indicator-based evolutionary algorithm based on pre-selection with classification guided by svm. *Applied Mathematics-A Journal of Chinese Universities*, 34(1):1–26, 2019.

[91] Haoyuan Li, Yi Wang, Dong Zhang, Ming Zhang, and Edward Y Chang. Pfp: parallel fp-growth for query recommendation. In *Proceedings of the 2008 ACM conference on Recommender systems*, pages 107–114, 2008.

[92] Michael Lindahl. *Strategic, Tactical and Operational University Timetabling*. PhD thesis, University of Denmark, 01 2017.

[93] Greg Linden, Brent Smith, and Jeremy York. Amazon. com recommendations: Item-to-item collaborative filtering. *IEEE Internet computing*, 7(1):76–80, 2003.

[94] Manuel López-Ibánez, Jérémie Dubois-Lacoste, Leslie Pérez Cáceres, Mauro Birattari, and Thomas Stützle. The irace package: Iterated racing for automatic algorithm configuration. *Operations Research Perspectives*, 3:43–58, 2016.

[95] Helena R Lourenço, Olivier C Martin, and Thomas Stützle. Iterated local search. In *Handbook of metaheuristics*, pages 320–353. Springer, 2003.

[96] Zhipeng Lü and Jin-Kao Hao. Adaptive tabu search for course timetabling. *European journal of operational research*, 200(1):235–244, 2010.

[97] Chris Meyer MacKenzie and Steve Noble. How retailers can keep up with consumers, 2013.

[98] Junrie B Matias, Arnel C Fajardo, and Ruji M Medina. A fair course timetabling using genetic algorithm with guided search technique. In *2018 5th International Conference on Business and Industrial Research (ICBIR)*, pages 77–82. IEEE, 2018.

[99] S Mohanty and RN Behera. Iterative deepening branch and bound. *arXiv preprint cs/9909003*, 1999.

[100] Hasfa Mounir, Pamela Wattebled, Julie Jacques, and Laetitia Jourdan. Une approche évolutionnaire multiobjectif pour la planification des cours professionnels. In *ROADEF 2021*, 2021.

[101] AR Mushi. Tabu search heuristic for university course timetabling problem. *African Journal of Science and Technology*, 7(1), 2006.

[102] Kawtar Najmani, El Habib Benlahmar, Nawal Sael, and Ahmed Zellou. Collaborative filtering approach: A review of recent research. In *International Conference on Advanced Intelligent Systems for Sustainable Development*, pages 151–163. Springer, 2020.

[103] Sorush Niknamian. Proposing a novel mathematical model and metaheuristic algorithm for university course timetabling with an educational quality approach. *Available at SSRN 3789765*, 2021.

[104] Anil Palaparthi, Tobias Riede, and Ingo R Titze. Combining multiobjective optimization and cluster analysis to study vocal fold functional morphology. *IEEE Transactions on Biomedical Engineering*, 61(7):2199–2208, 2014.

[105] Vilfredo Pareto. *Cours d'economie Politique*, volume 1. F. Rouge, 1896.

[106] Kenekayoro Patrick and Zipamone Godswill. Greedy ants colony optimization strategy for solving the curriculum based university course timetabling problem. *Journal of Advances in Mathematics and Computer Science*, pages 1–10, 2016.

[107] Michael J Pazzani and Daniel Billsus. Content-based recommendation systems. In *The adaptive web*, pages 325–341. Springer, 2007.

[108] Leif E Peterson. K-nearest neighbor. *Scholarpedia*, 4(2):1883, 2009.

[109] Juan Ramos et al. Using tf-idf to determine word relevance in document queries. In *Proceedings of the first instructional conference on machine learning*, volume 242, pages 29–48. New Jersey, USA, 2003.

[110] Marco Tulio Ribeiro, Nivio Ziviani, Edleno Silva De Moura, Itamar Hata, Anisio Lacerda, and Adriano Veloso. Multiobjective pareto-efficient approaches for recommender systems. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 5(4):1–20, 2014.

[111] Xin Rong. word2vec parameter learning explained. *arXiv preprint arXiv:1411.2738*, 2014.

[112] Peter Ross and Dave Corne. Comparing genetic algorithms, simulated annealing, and stochastic hillclimbing on timetabling problems. In *AISB workshop on evolutionary computing*, pages 94–102. Springer, 1995.

[113] Scott E Sampson and Elliott N Weiss. Increasing service levels in conference and educational scheduling: A heuristic approach. *Management Science*, 41(11):1816–1825, 1995.

[114] Andrea Schaerf. *Tabu search techniques for large high-school timetabling problems*. Centrum voor Wiskunde en Informatica, 1996.

[115] D Seung and L Lee. Algorithms for non-negative matrix factorization. *Advances in neural information processing systems*, 13:556–562, 2001.

[116] John Silberholz and Bruce Golden. Comparison of metaheuristics. In *Handbook of metaheuristics*, pages 625–640. Springer, 2010.

[117] Slurm. Slurm workload manager.

[118] Behzad Soleimani Neysiani, Nasim Soltani, Reza Mofidi, and Mohammad H. Nadimi-Shahraki. Improve performance of association rule-based collaborative filtering recommendation systems using genetic algorithm. *International Journal of Information Technology and Computer Science*, 11:48–55, 02 2019.

[119] Kenneth Sörensen. Distance measures based on the edit distance for permutation-type representations. *Journal of Heuristics*, 13(1):35–47, 2007.

[120] Tiago Sousa, Hugo Morais, Rui Castro, and Zita Vale. Evaluation of different initial solution algorithms to be used in the heuristics optimization to solve the energy resource scheduling in smart grids. *Applied Soft Computing*, 48:491 – 506, 2016.

[121] Nidamarthi Srinivas and Kalyanmoy Deb. Muiltiobjective optimization using nondominated sorting in genetic algorithms. *Evolutionary computation*, 2(3):221–248, 1994.

[122] Witold Stankiewicz, Robert Roszak, and Marek Morzynski. Genetic algorithm-based calibration of reduced order galerkin models. *Mathematical Modelling and Analysis*, 16(2):233–247, 2011.

[123] El-Ghazali Talbi. *Metaheuristics: from design to implementation*, volume 74. John Wiley & Sons, 2009.

[124] Ian Tenney, Dipanjan Das, and Ellie Pavlick. Bert rediscovers the classical nlp pipeline. *arXiv preprint arXiv:1905.05950*, 2019.

[125] Kevin Tierney, Dario Pacino, and Stefan Voß. Solving the pre-marshalling problem to optimality with a* and ida. *Flexible Services and Manufacturing Journal*, 29(2):223–259, 2017.

[126] Nguyen Huy Truong and Dinh-Nam Dao. New hybrid between nsga-iii with multi-objective particle swarm optimization to multi-objective robust optimization design for powertrain mount system of electric vehicles. *Advances in Mechanical Engineering*, 12(2):1687814020904253, 2020.

[127] David A Van Veldhuizen. Multiobjective evolutionary algorithms: classifications, analyses, and new innovations. Technical report, AFIT, 1999.

[128] Saúl Vargas. Novelty and diversity enhancement and evaluation in recommender systems and information retrieval. In *Proceedings of the 37th international ACM SIGIR conference on Research and development in information retrieval*, pages 1281–1281, 2014.

[129] Christos Voudouris and Edward PK Tsang. Guided local search. In *Handbook of metaheuristics*, pages 185–218. Springer, 2003.

[130] Wesley Waldner and Julita Vassileva. Emphasize, don't filter! displaying recommendations in twitter timelines. In *Proceedings of the 8th ACM Conference on Recommender systems*, pages 313–316, 2014.

[131] Kaixiang Wang, Wenqian Shang, Mengyu Liu, Weiguo Lin, and Hao Fu. A greedy and genetic fusion algorithm for solving course timetabling problem. In *2018 IEEE/ACIS 17th International Conference on Computer and Information Science (ICIS)*, pages 344–349. IEEE, 2018.

[132] Pan Wang, Xingquan Zuo, Congcong Guo, Ruihong Li, Xinchao Zhao, and Chaomin Luo. A multiobjective genetic algorithm based hybrid recommendation approach. In *2017 IEEE Symposium Series on Computational Intelligence (SSCI)*, pages 1–6. IEEE, 2017.

[133] Shanfeng Wang, Maoguo Gong, Haoliang Li, and Junwei Yang. Multi-objective optimization for long tail recommendation. *Knowledge-Based Systems*, 104:145–155, 2016.

[134] Shanfeng Wang, Maoguo Gong, Lijia Ma, Qing Cai, and Licheng Jiao. Decomposition based multiobjective evolutionary algorithm for collaborative filtering recommender systems. In *2014 IEEE Congress on Evolutionary Computation (CEC)*, pages 672–679. IEEE, 2014.

[135] Damien Woods and Adrian Trenaman. Simultaneous satisfaction of hard and soft timetable constraints for a university department using evolutionary timetabling. 04 2001.

[136] Ruobing Xie, Yanlei Liu, Shaoliang Zhang, Rui Wang, Feng Xia, and Leyu Lin. Personalized approximate pareto-efficient recommendation. In *Proceedings of the Web Conference 2021*, pages 3839–3849, 2021.

[137] Shangyao Yan and Chich-Hwang Tseng. A passenger demand model for airline flight scheduling and fleet routing. *Computers & Operations Research*, 29(11):1559–1581, 2002.

[138] Qingfu Zhang and Hui Li. Moea/d: A multiobjective evolutionary algorithm based on decomposition. *IEEE Transactions on evolutionary computation*, 11(6):712–731, 2007.

[139] Yi Zhou, Fazhi He, and Yimin Qiu. Optimization of parallel iterated local search algorithms on graphics processing unit. *The Journal of Supercomputing*, 72(6):2394–2416, 2016.

[140] Eckart Zitzler. *Evolutionary algorithms for multiobjective optimization: Methods and applications*, volume 63. Citeseer, 1999.

[141] Eckart Zitzler, Kalyanmoy Deb, and Lothar Thiele. Comparison of multiobjective evolutionary algorithms: Empirical results. *Evolutionary computation*, 8(2):173–195, 2000.

[142] Eckart Zitzler and Simon Künzli. Indicator-based selection in multiobjective search. In *International conference on parallel problem solving from nature*, pages 832–842. Springer, 2004.

[143] Eckart Zitzler, Marco Laumanns, and Lothar Thiele. Spea2: Improving the strength pareto evolutionary algorithm. *TIK-report*, 103, 2001.

[144] Eckart Zitzler and Lothar Thiele. Multiobjective evolutionary algorithms: a comparative case study and the strength pareto approach. *IEEE transactions on Evolutionary Computation*, 3(4):257–271, 1999.

[145] Eckart Zitzler, Lothar Thiele, Marco Laumanns, Carlos M Fonseca, and Viviane Grunert Da Fonseca. Performance assessment of multiobjective optimizers: An analysis and review. *IEEE TEC*, 7(2):117–132, 2003.

[146] Yi Zuo, Maoguo Gong, Jiulin Zeng, Lijia Ma, and Licheng Jiao. Personalized recommendation based on evolutionary multi-objective optimization [research frontier]. *IEEE Computational Intelligence Magazine*, 10(1):52–62, 2015.