DEPARTMENT OF INFORMATICS

TECHNISCHE UNIVERSITÄT MÜNCHEN

Master's Thesis in Informatics

Item Combination in Destination Recommendation Systems

Abidemi Shobayo-Eniola

DEPARTMENT OF INFORMATICS

TECHNISCHE UNIVERSITÄT MÜNCHEN

Master's Thesis in Informatics

Item Combination in Destination Recommendation Systems

Kombination von Items in Empfehlungssystemen für Reiseziele

Author:	Abidemi Shobayo-Eniola
Supervisor:	Prof. Dr.-Ing. Jörg Ott
Advisor:	Dr. Wolfgang Wörndl
Submission Date:	April 28, 2022

I confirm that this master's thesis in informatics is my own work and I have documented all sources and material used.

Munich, April 28, 2022 Abidemi Shobayo-Eniola

Abstract

Recommender systems produce individualized recommendations by aggregating users preference. The issue of item combination in travel recommender systems is a computationally hard problem. Numerous works have studied point-of-interest recommendation for travel recommender systems as a time dependent orienteering problem. However, limited work have studied the topic of item combination as a non-time dependent orienteering problem

In this thesis, we establish multi-objective evolutionary algorithms as a novel approach to solving the non-time-dependent orienteering problem. Extensive study of various state-of-the-art algorithms used to solve the orienteering problem is conducted. Then, a mathematical reformulation of the problem as a multi-objective orienteering problem is presented. Further empirical research of meta-heuristic solutions to the problem definition resulted in selecting a non-dominated genetic evolutionary algorithm for solving the problem.

The NSGA-III and five heuristic variations thereof were tested and evaluated. An evaluation of the experiments was conducted online based on computational speed and high fitness score. An offline user evaluation was also conducted. It was discovered that a beginning population consisting only of feasible individuals and penalty-based constraint handling using individuals' distances to the feasible region was rated best by users regarding accuracy, satisfaction, cohesiveness, and diversity.

Kurzfassung

Empfehlungssysteme erstellen individualisierte Empfehlungen auf der Grundlage zusammengefasster Präferenzen der Nutzer. Das Problem der Item-Kombination in Reiseempfehlungssystemen ist ein rechenintensives Problem. Zahlreiche Arbeiten haben Point-of-Interest-Empfehlungen für Reiseempfehlungssysteme als ein zeitabhängiges Orientierungsproblem untersucht. Es gibt jedoch nur wenige Arbeiten, die sich mit dem Thema der Item-Kombination als nicht-zeitabhängiges Orientierungsproblem beschäftigt haben.

In dieser Arbeit definieren wir das Problem der Item-Kombination als ein nicht-zeitabhängiges Orientierungsproblem und etablieren multi-objektive evolutionäre Algorithmen als einen neuen Ansatz zur Lösung des Problems. Es wird eine umfassende Studie verschiedener moderner Algorithmen durchgeführt, die zur Lösung des Orientierungsproblems verwendet werden. Anschließend wird eine mathematische Neuformulierung des Problems als multikriterielles Orientierungsproblem vorgestellt. Weitere empirische Untersuchungen von metaheuristischen Lösungen für die Problemstellung führten zur Auswahl eines nicht-dominanten genetischen Evolutionsalgorithmus für die Lösung des Problems.

Der NSGA-III und fünf heuristische Varianten davon wurden experimentell untersucht und bewertet. Die Bewertung der Experimente erfolgte online anhand der Rechengeschwindigkeit und der hohen Fitnesswerte. Außerdem wurde eine Offline-Evaluierung mittels Benutzerbewertung durchgeführt. Es wurde festgestellt, dass eine Startpopulation, die nur aus realisierbaren Individuen besteht, und eine penalty-basierte Beschränkungsbehandlung, die den Abstand der Individuen zur realisierbaren Region verwendet, von den Nutzern hinsichtlich Genauigkeit, Zufriedenheit, Kohäsion und Diversität am besten bewertet wurde.

Contents

1. Introduction

Recommender systems (RSs) produce individualized recommendations based on aggregating users' preferences [1]. They have the effect of guiding the user in a personalized way to interesting or valuable items in an ample space of possible options [2]. Given an input of the user's preferences, recommendation algorithms can generate a sequence of the recommended item(s). RSs are typically applied information retrieval (IR), human-computer interaction (HCI), and data mining (DM) fields [1]. Amazon [3] uses recommendation algorithms to personalize the online store for each customer, for example showing programming titles to a software engineer and baby toys to a new mother. Netflix [4] also uses sophisticated recommendation algorithms to suggest movies based on user profiles and behavior. RSs are also used to suggest articles, people, music, news, etc., in e-commerce websites such as Spotify, LinkedIn, and others.

Although there are multiple domain types which RSs can be applied, a RSs typically focuses on a specific item type or domain. There are also valuable use cases of RSs in tourism. This thesis focuses solely on RSs in the tourism domain. Travel recommendation and trip planning are popularly researched areas with different proposed systems [5]–[9] and commercial destination recommendation tools (e.g. Triparoti[1], Tripzard[2], Besttripchoices[3]). Designing travel RSs is intractable because of the number of possible destinations and the complex and multi-layered nature of the preferences and needs of tourists. Computing an optimal combination of destinations is an even more complex task.

Travel recommendation can be described as a tourist trip design problem (TTDP) [10]. A TTDP model typically consists of a set of candidate points of interests (POIs), each associated with a number of attributes (e.g., activities, location), and a score for each POI, is calculated as a weighted function of the objective or subjective value of each POI. The objective of solving the TTDP is to maximize the collected score of each sequence of ordered visits to the POIs while respecting user constraints related to travel cost and POI attributes [11]. In its most basic form, the TTDP is equivalent to an orienteering problem (OP) [10]. The OP, similar to the TTDP, seeks to maximize the total collected profit by visiting selected nodes (i.e., POIs) of a given value [12]. In this problem, not all available nodes can be visited due to the limited time budget. Thus, a standard OP can be interpreted as a combination of the Knapsack Problem and the traveling salesman problem (TSP) [13].

[1]Triporati: `www.triporati.com`
[2]Tripzard: `www.tripzard.com`
[3]Besttripchoices: `www.besttripchoices.com`

A number of OP variants focus on optimizing to find the best routes between POIs (i.e., route planning) under POI attribute constraints without or within a given time frame. Hence, OPs can be viewed as the TSP with profits. The TTDP presented in this work does not include optimizing for routes between POIs (i.e., we do not recommend the order of visits to the chosen destinations). The objective function of an OP is typically modeled similarly to a knapsack problem. A standard knapsack problem is a 0–1 integer programming model and is formally defined as follows:

$$maximize \qquad \sum_{i=1}^{n} p_i x_i \tag{1}$$

$$subject\ to \qquad \sum_{i=1}^{n} w_i x_i \leq W \tag{2}$$

$$x_i \in \{0,1\}, \qquad \forall\, 1 \leq i \leq n \tag{3}$$

where each item i is associated with a profit p_i. The decision variable is, $x_i = 1$, when an item is placed inside the knapsack; otherwise it is $x_i = 0$. The objective function 1 is to maximize the total profit from collected items. Constraint 2 limits the weight w_i of items placed inside the knapsack based on the total capacity of the knapsack W. The goal of a knapsack problem is to maximize the value of the items placed in the knapsack without going over a weight limit or capacity. Intuitively, the sequence of trips to be recommended by our RS can be described as the knapsack, while the items to be placed in the knapsack are the single POIs (i.e., destinations). The weight limit enforced by the knapsack can be thought of as the budget and time constraints, while the value to be maximized by the knapsack problem is the score of each POI for given user preferences and POI attributes.

To illustrate the challenge of designing RSs for composite trips, consider the following scenario: A person wants to travel for a three weeks' holiday in March, and she has a budget of €1,500. Depending on her traveling style, her preferred activities may include hiking, biking, or visiting cultural attractions. The RS must recommend a combination of destinations from many possible destinations while respecting the traveler's time and budget limitations. The traveler will derive more satisfaction if the RS suggests a stay duration for each recommended destination. Furthermore, the RS must consider the user's preferred activities so that they do not miss out on activities during their trip. Additionally, for maximum value, the RS should consider factors like the weather during the chosen time of the year, security of the regions, and proximity of the destinations per the trip agenda. Exhaustively enumerating all possible solutions is impractical in moderate and large instances because of the combinatorial explosion of the number of possible solutions.

In this thesis, we investigate various state-of-the-art algorithms that have been used in research to solve the OP in TTDPs. Solutions to real-life optimization problems usually must be evaluated considering different points of view corresponding to multiple objectives that are often in conflict. We describe an OP as a multi-objective orienteering problem (MOOP) in which there are several pleasure categories for each POI (e.g., shopping, cultural) with each POI having a distinct profit per category. At the time of this thesis, the data we collated is not

voluminous enough for us to consider deep-learning techniques that could be applied to the TTDP. Hence, we do not research deep-learning algorithms or techniques

The main goals of this thesis are as follows:

- A formal definition of the composite trip recommendation problem as an optimization problem modeled as a MOOP;
- Empirical research into how current state-of-the-art algorithms used for solving OPs and general MOOPs can be extended;
- An efficient algorithm to obtain a sequence of candidate solutions that satisfy user constraints in a destination RS;
- Varying adaptations and clearly defined implementations of the algorithm; and
- An online comparative study and an offline user evaluation of the performance of the implemented algorithm variants using metrics such as computational speed, user satisfaction, diversity, cohesiveness, and accuracy.

Subsequent parts of this thesis are structured as follows. In Chapter 2, we review the available literature on various state-of-the-art approaches to travel recommendation. In Chapter 3 we formally define the composite trip recommendation problem as a MOOP and provide empirical research into possible algorithmic approaches for solving TTDP and how they can be adapted to the MOOP. In Chapter 4, we describe our algorithms to identify candidate solutions to the MOOP as well as their implementation. We evaluate the results of our implementation in Chapter 5. Finally, in Chapter 6 we discuss the results and propose possible paths for future research.

2. Literature Review

In 2007, Vansteenwegen and Oudheusden [10] introduced a popular TTDP design. Their paper defined features that can be described as a template for the next-generation mobile tourist guide (MTG). Based on their research, the authors projected that the next generation MTG would become more personalized with components such as user profiles, attraction information, and trip information playing essential roles. They also speculated that operations research (OR) would play a pivotal role in conceiving and optimizing tourist trips. Thus, they defined the TTDP as a decision tool.

With their groundbreaking projections and proposals, technologies such as artifical intelligence (AI), internet-of-things (IOT) and cyber-physical systems (CPS), have brought about a paradigm shift in the possibilities for a dynamic tourist destination guide. Many recent studies have focused on designing and optimizing tourist trips with the aid of technological advancements. TTDP as a decision tool for next-generation MTG has been thoroughly researched. However, this research has recently become less mainstream because of options offered by newer technologies. It has become the de facto approach to personalizing new generation tourist guides and trip RSs. We explore such personalized RSs after surveying recommendation techniques commonly found in the literature.

2.1. Recommendation Techniques

Generally, RSs can be distinguished based on the issues they focus on and the techniques they use. As shown in Figure 2.1, RSs can be based on collaborative filtering (also known as social filtering), content filtering, and knowledge-based filtering. Systems designed according to collaborative filtering identify users whose preferences are similar to those of the given user by computing similarities between users and items they have liked [14]. RSs use this filtering method extensively. Collaborative filtering often involves model-based and memory-based approaches. Model-based approaches typically use machine learning or statistical methods to build models based on user records for future recommendations [15]. In contrast, memory-based approaches compare user data with previously stored data of other users [16]. This technique typically suffers from a cold-start problem. Such a problem occurs when there is not enough information about an item based on another user either rating it or specifying which other items it is similar to [14]. Also, data becomes sparse when there are not enough users available to cover the needed collection of recommendable items. The authors of [17], [18] analyzed a set of items liked by a user in the past and tried to recommend similar

items, resulting in a judgment of relevance that represents users' preferences. Thus, it is highly advantageous in its effectiveness as an information retrieval process. Systems based on content filtering suffer from over-specialization. This is a function of the similarity between the recommended items based on users' preferences at different points [19]. It also suffers from a cold-start problem whereby the system needs to have adequate user historical data to generate quality results [2].

Knowledge-based systems [20] depend on knowledge models of the object domain for effective item recommendation. They are based on the needs and preferences the user provides. Constraint-based techniques [21] allow systems to use domain knowledge regarding the features that cannot be represented by the user's record, or in the absence of the user's record, for recommending items. Case-based methods [22] typically utilize the experiences and expertise of travel agents. Knowledge-based methods do not suffer from the problems faced in collaborative and content-filtering systems; their recommendations do not depend on a database of user ratings, and the systems do not have to gather information about a particular user because its judgments are independent of individual tastes [20]. Therefore, knowledge-based systems are perfect complements to other RS types.

Hybrid approaches [23], [24] combine two or more of the approaches mentioned above to mitigate the weaknesses of each approach and complement their strengths for optimal performance.

Depending on the recommendation technique employed, destination RSs can be classified as content based, collaborative, or knowledge based. However, a good deal of recent research has heavily focused on model-based collaborative filtering.

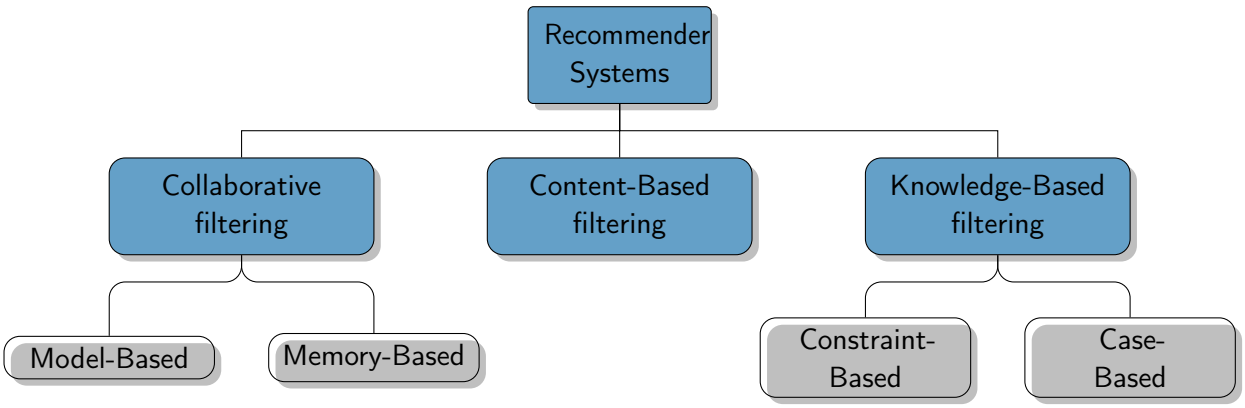


Figure 2.1.: An overview of approaches used in recommender systems.

2.2. POI Recommendation

The proliferation of social media as a societal norm has paved the way for Location-based social networks (LBSNs) like Foursquare and Facebook. Consequently, a new study area

has been established in destination recommendation research. POI recommendation, as it is commonly known, is an active research area that focuses solely on recommending destinations to users on LBSN. Such systems utilize readily available check-in records of a user on the network to recommend further possible POIs to the user. Advancements in POI recommendation need to be explored because they constitute a significant step toward extracting user preference data that can be generalized to improve destination recommendation in systems that are outside location-based networks.

The main objective of POI recommendation on LBSNs is to optimize the user experience by recommending POIs based on users' historical records and sometimes on collaborative information. LBSNs have been researched in recent years due to the spatial-temporal-social information embedded in them [25]. Typically, users check in with timestamps, location details, and their preferences that inform the POI ratings, which can be accessed via an application program interface provided by the LBSN provider. As a consequence, many recent studies have experimented with representing the data sets through statistical or machine models to mine patterns in the LBSN data for the POI prediction task. The data characteristics allow the definition of new properties that can be mined from them.

Latent dirichlet allocation (LDA) is a natural language processing statistical model that has been heavily adopted by researchers to extract POI topics from comment sections [26], [27]. Liao et al. [26] explored comment relations using LDA. In their study, their latent features were user-topic-time tensors. The tensors were constructed by connecting check-in data with a POI topic generated by a LDA model. The POI model extracts the topics based on the comments given by users, the topics, and POI-topic distributions of all POIs.

Some of the most successful realizations of statistical models for RSs are based on matrix factorization [18], [28]–[31]. Matrix factorization [32] algorithms work by mapping items to a joint latent factor space of dimensionality such that user-item interactions are modeled as inner products in that space. Consequently, high correspondence between item and user factors leads to a recommendation. A major strength of matrix factorization is that it allows the incorporation of additional information. In a LBSN, the social relations between users help in incorporating similarity information between users' interests into matrix factorization [26]–[28]. Liao et al.[26] used a higher-order singular value decomposition (HOSVD) of third-order tensors to recommend POIs. HOSVD can be considered a generalization of a matrix singular value decomposition [33]. However, factorizing the matrix often raises difficulties due to the high portion of missing values caused by sparseness in the matrix resulting from cold start. Huang et al. [27] addressed this by simultaneously mining the sequential, temporal, and spatial patterns of users' check-in behavior such that data were abundant. Others including Cheng et al.[28] employed a low-rank approximation of sparse, partially observed tensors.

In addition to the social relations that can be incorporated into matrix factorization or topics, many existing studies have used the Markov-chain property of inter check-in behavior to model the sequential check-in pattern of users [28], [34], [35]. Cheng et al. [28] explored the

geographical relation of LBSN inter check-in distance in order to recommend successive POIs to users by using a personalized Markov chain (FPMC). Liu et al. [36] employed recurrent neural networks (RNN) to detect subsequent correlations of check-in sequences.

Generally, due to the abundance of LBSN data available to these models, the results obtained in many studies of POI recommendation are mostly improvements to previous studies. Additionally, applying model-based techniques and using approximation algorithms has helped achieve a breakthrough regarding the cold-start problem that sometimes occurs in collaborative filtering. These steady improvements in POI recommendation precision and accuracy are indeed groundbreaking and constitute a significant step toward improving destination recommendation as a whole.

2.3. OP solving methodologies

Though POI recommendation is an attractive research area, much research has also been done into destination recommendation outside LBSNs. The OP that originates from operations research is still regarded as a tool for modeling the TTDP and, by extension, solving the problem of destination recommendation. According to [37], given a set of nodes $N = \{1, ..., |N|\}$ where each node $i \in N$ is associated with a non-negative score S_i, and the nodes 1 and N are the start and end nodes, respectively, the goal of the OP is to determine a path, limited by a given time budget T_{max}, that visits a subset of N and maximizes the total collected score. Each collected score can be added, and each node can be visited at most once. Many researchers have solved different problems by formulating them as variants of the OP; these variants include the OP. Such variants include team orienteering problem (TOP), time dependent orienteering problem (TDOP), orienteering problem with time windows (OPTW), team orienteering problem with time windows (TOPTW), OP with stochastic profits (OPSP), MOOP, and more recently the thief orienteering problem (ThOP).

Gunawan et al. (2016) [13] published a paper surveying all recent variations, solution approaches, and applications of OP variants. Although numerous studies have thoroughly researched some OP variants, it is nearly impossible to find studies for some other variants in the literature. MOOP is the variant closest to our objective; it derives its name from multi-objective optimization in OR in which there is more than one objective function. A MOOP assigns a set of scores S_{ik} to each POI to model the different scores a POI might have for each attraction category $k \in \{1, ..., m\}$. Exact algorithms for the OP are complex and computationally time-consuming because it is NP-hard [38]. Therefore, heuristic approaches such as those found in [12], [38]–[40] are the focus of most research.

A good deal of literature uses several meta-heuristics to solve different OP variants. Most of the meta-heuristics mimic natural metaphors to solve optimization problems (e.g., evolution of species, annealing process, ant colony, particle swarm, immune system, bee colony, and wasp swarm). Swarm algorithms such as Ant colony optimization (ACO) [41]–[44] and particle swarm optimization (PSO) [42], [45], [46] are meta-heuristics commonly found in

literature for solving two-objective and multi-objective OPs. In their survey Vansteenwegen and Oudheusden [37] found ACO-based algorithms to be the best-performing algorithms for the TOP in their experiments. Memetic algorithms as used in [47]–[49] have also been used to solve different variants of OP problems. Evolutionary algorithms (EAs) that mimic genetics constitute an algorithmic trend in the literature [50]–[52]. Swarm algorithms, EA, memetic algorithms, and simulated annealing as used in [53], are commonly combined with other heuristics such as tabu search [47] or pure local search [48], [49] to achieve the best results. For example, Labadie et al. [54] developed a hybridized evolutionary local search algorithm for TOPTW that showed an improvement over the 150 best-known solutions, as shown in [13]. Recent studies have focused on exploring the strengths of many of these state-of-the-art meta-heuristics. However, standard heuristics such as the branch-and-cut algorithm used in [55] and tabu search used in [56] are known to show positive results for the TOP.

2.4. Related Work

A good deal of literature focuses either on travel recommendation or bundling items in a recommendation package. However, there is little research into bundling items for destination recommendation. A non-time-dependent MOOP best describes our OP variant. To the best of our knowledge, no researchers have carried out travel recommendations using this OP variant. Martín-Moreno and Vega-Rodríguez [44] proposed a swarm intelligence-based algorithm to solve a time-dependent MOOP. However, due to little work having been done in OP for multi-objectives, benchmark instances to compare the performance of their algorithm are two-objective OP based. Moreover, the MOOP presented seeks to find optimal tours and not paths (i.e., start at node A and end at node A).

At the core of this thesis are aspects relating to other works found in the literature. The following sections highlight the two most closely related works.

2.4.1. The Oregon Trail Knapsack

Burg et al. [57] defined an Oregon Trail knapsack problem that was inspired by Oregon Trail game. The game asks players to imagine preparing for a trek along the Oregon Trail. With a given amount of money to spend and a weight limit on supplies, the travelers need to get good value for the supplies they purchase to make it across the country. This extension of the knapsack problem can be considered similar to our knapsack problem. However, the Oregon Trail knapsack problem does not add a time constraint to the travel node as found in

a standard OP. The authors' formal definition of the problem is as follows:

$$\textit{maximize} \qquad \sum_{j=1}^{n} f_j(x_j, x_{d_j}) \tag{1}$$

$$\textit{subject to} \qquad \sum_{j=1}^{n} w_j x_j \leq W \tag{2}$$

$$\sum_{j=1}^{n} c_j x_j \leq C \tag{3}$$

$$0 \leq x_j \leq b_j, \qquad \forall\, 1 \leq j \leq n \tag{4}$$

where W in Equation 2 is the weight limit, and C in Equation 3 is the cost limit. The function in Equation 1 is the value function for Type j, in which d_j gives the index of the type upon which the value of x_j depends. $d_j \in \{d_1, d_2, ..., d_n\}$ provides the dependency information among the item types. The value function in this model is of particular interest because it models three possible value types of the items. Burg and Lang[57] define the value functions as follows:

$$f_j(x_j, x_{d_j}) = x_j \cdot v_j \cdot [x_{d_j} > 0] \tag{type 1}$$

$$f_j(x_j, x_{d_j}) = [x_{d_j} > 0] \cdot \sum_{i=0}^{x_j - 1} r^i \cdot v_j \tag{type 2}$$

$$= [x_{d_j} > 0] \cdot v_j \cdot \frac{1 - r^{x_j}}{1 - r}, \qquad \forall\, 0 < r < 1$$

$$f_j(x_j, x_{d_j}) = x_j \cdot v_j - [x_{d_j} > 0] \cdot t \cdot x_j \cdot v_j, \qquad \forall\, 0 < t \leq 1 \wedge d_j \neq j \tag{type 3}$$

Each function has a value constant given by v_j. The Iverson notation $[x_{d_j} > 0]$ denotes the boolean value $x_{d_j} \in \{0, 1\}$, which is when $x_{d_j} > 0$. Value function Type type 1 represents the case, in which items of Type j only have a value if at least one item of Type d_j is present in the solution. The value function Type type 2 covers the value type, in which with each added item of Type j, the item value v_j diminish at a rate of r. Once more, the function adds the value of items of Type j if at least one item of type d_j is in the knapsack. Lastly, the Type-type 3 function represents the case where a factor of t reduces the value associated with x_j of Type-j items, when Type-x_{d_j} items are in the solution.

2.4.2. Composite Trip Recommendation

Wörndl and Herzog [6] researched combining multiple travel regions into a composite trip. Their algorithmic approach comprises three phases:

1. Reduce number of regions
2. Rate regions

3. Calculate the best combination of regions

Phases 1 and 2 are preliminary steps taken before the actual recommendation process. This approach allows users to exempt regions from their query. The hierarchical tree data model used enables the procedure to exempt sub-regions of exempted regions. Thus, the number of possible regions to explore is significantly reduced before starting the next phases. In Phase 2, an asymmetric similarity metric is used to assign preference ranks to regions. The asymmetrical similarity metric allows region features to be assigned weights. For example, the traveling period is weighted higher than features such as crime level. Regions with low scores are then removed from consideration before the next step. Finally, the best combinations of regions are calculated using dynamic programming. Intrinsic details regarding the dynamic programming approach used in the developing their travel package algorithm are unclear. The algorithm was shown to satisfy expert users more than a baseline knapsack algorithm and a top-k algorithm.

2.5. Conclusion

In this chapter, we reviewed different techniques found in literature for item recommendation. We investigated state-of-art methodologies found in the literature for general POI recommendation and then went on to review how OPs have been solved in the literature. Wörndl and Herzog [6] did substantial work on item combination for destination RSs. However, their algorithm fails to achieve high diversity in regions [6] and their approach fails to scale. Penalty-based approaches as used in their work can either improve the heuristic's efficiency, or it can restrict the effectiveness of the heuristic. This depends on the penalty function definition and the chosen penalty value. Our work seeks to improve the work of Wörndl and Herzog by exploring methodologies and techniques to efficiently optimize the item-combination procedure. In addition, our thesis contributes to existing research on OPs by characterizing a multi-objective, non-time-dependent OP. Thereby, more efficient heuristics for computing combinations of regions can be explored. Overall, higher diversity in recommended regions is ensured. A robust framework of problem-model and scalable algorithmic approaches to building our RS for recommending diverse sequences of travel regions based on given user preferences is developed in subsequent chapters.

3. Solution Development

In a previous work on this topic, Wörndl and Herzog [6] drew inspiration for their formal model based on the Oregon Trail knapsack problem (see Section 2.4.1); they formulated a model that extends the value function of the problem using a penalty function.

For clarity, we adjusted the notation used in their paper to fit the notations used in previous function values. Wörndl and Herzog formally defined their value function as follows:

$$f_j(x_j, x_{d_j}) = x_j \cdot v_j - \sum_{e \in x_{d_j}} (t(j,e) \cdot x_j \cdot v_j \cdot [x_e > 0]) \tag{1}$$

where v_j is the value of region j (i.e, item) for the specified user query, and $t(j,e)$ is the penalty function for the two regions j and e. Their value function is similar to the function Type type 3 specified by the original Oregon Trail knapsack model. However, the previously defined value-reducing constant factor t, is a function of the two regions j and e.

Wörndl and Herzog's formal model is defined as follows:

$$\begin{aligned}
&\textit{maximize} && \sum_{i=1}^{n} f_j(x_j, x_{d_j}) && (1)\\
&\textit{subject to} && \sum_{i=1}^{n} x_i d_i \leq D && (2)\\
& && \sum_{i=1}^{n} x_i \; b_i \leq B && (3)\\
& && x_i \in \{0,1\} && (4)
\end{aligned}$$

where $f_j(x_j, x_{d_j})$ is the penalty function defined in 1; d_i is item i's recommended duration of stay, and b_i is item i's recommended daily budget. D is the maximum possible duration of stay and B is the maximum budget the user can spend on their trip.

Wörndl and Herzog concluded that the penalty function defined in their work could be improved to produce better results. Building penalty-based functions requires careful consideration of the function parameters and the function itself. Thus, to gain freedom from the influence of the penalty function, we reformulate and extend the problem definition to a non-penalty-based approach. If through further analysis, a penalty-based approach proves to be a better fit for our selected algorithm, we will redefine a penalty function as needed.

The framework for our RS comprises a knapsack model for the multi-objective OP, a data model, and a meta-heuristic solution. In the following sections, we analyze these three parts in detail.

3.1. Problem Reformulation

Let us consider n regions $(i = 1, ..., n)$. Each region i will have k profits s_{ik} $(p = 1, ..., k)$. Given a set N of m traveling preferences $N = \{N_1, N_2, ..., N_m\}$, s_{ik} is dependent on i's score on N. For example, a region can have a different score for shopping and another for hiking. The recommendation problem can be modeled as a multi-objective OP. Given a user query, some regions must be selected to maximize the p total profits while not exceeding the budget limit B and the total stay duration D. Each user preference must be fulfilled by at least one of the selected regions. Assuming the system recommends Trip T comprising n regions $(T = x_1, x_2, ..., x_n)$, the physical distance between a region x_i and its neighbor x_2 should not exceed a defined constant σ. In a case in which no location coordinates for the regions are available, the regions in the recommended trip should be within reasonable distance from one other. Additionally, the duration of stay in a particular region x_i should ensure a minimum utility value, which is defined in the algorithm.

The mathematical model of the MOOP is defined as follows:

$$\textit{maximize} \qquad z_n(x) = \sum_{k=1}^{p} x_i \, s_{ik}, \qquad i = 1, ..., n \tag{1}$$

$$\textit{subject to} \qquad \sum_{i=1}^{n} x_i \, d_i \leq D \tag{2}$$

$$\sum_{i=1}^{n} x_i \, b_i \leq B \tag{3}$$

$$\sum_{j \in N} x_{ij} \geq 1 \tag{4}$$

$$\sum_{i=1}^{n} x_i \, g(p_i) \leq \sigma, \tag{5}$$

$$x_i \in \{0, 1\}, \qquad \forall \, 1 \leq i \leq n \tag{7}$$

where $x_i = 1$ when item i is chosen; otherwise $x_i = 0$. Constraints 2 and 3 represent the constraints on total stay duration and budget, respectively. Equation 4 ensures that each user preference is satisfied at least once. Constraint 5 represents the physical distance between the regions, in which g is a function of the physical location. It is assumed that all coefficients s_{ik}, b_i, B, d_i, and D are positive.

Pareto Optimum

There are multiple scores to be maximized in the objective function. Therefore, choosing an objective maximizing optimal solution without trade-offs is infeasible. In multi-objective optimization, Solution s is said to dominate s' if s is at least as good as s' in every criterion, and s is better in at least one criterion; $s \succ s'$ is used to denote such a case. If no solution dominates Solution s^*, then s^* is said to be *Pareto optimal* (i.e., non-dominated). The set of all non-dominated vectors is known as the *Pareto front*.

Example: Figure 3.1 a two-objective instance of the MOOP problem. For simplicity, we assume two given preferences. The gray box displays the user input. When mapped to our problem definition, there are two objectives to be maximized, where an objective represents a single item. An item can be any POI, for example attraction sites, region, or route. $f = (\vec{f}_1, \vec{f}_2, ..., \vec{f}_k)$ represents all feasible solutions. We compute a solution in the objective space $\vec{f}_i = (z_1, z_2)$. The objective values are computed on preferences M and S, while respecting the necessary constraints. Parameter x in the illustration is a boolean array that encodes the selection of an item combination. The selected item combinations are subject to the notion of Pareto optimality defined above. For example $f_7 \succ f_2$ because it is better in at least one of its objective values as f_2 and it is strictly better in another value. The Pareto front consists of the selections $\{f_1, f_5, f_7, f_9, f_{10}\}$, and we achieve a total score of (24, 23).

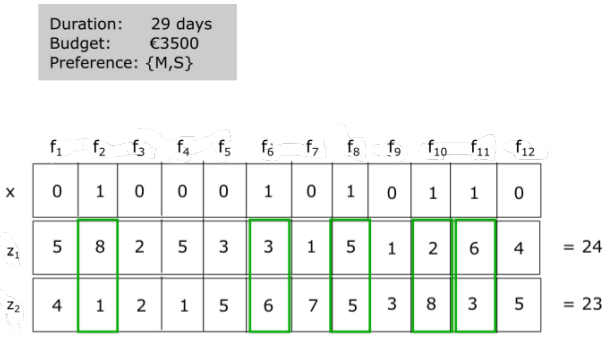


Figure 3.1.: Example solution to the multi-objective orienteering problem

Pareto optimality here implies that there is no item combination f_k that will improve the achievable score of at least one item without diminishing the achievable score of another item.

We can also extend the concept of Pareto optimality from the solution space to the objective space. Given n objective functions $z = (z_1, ..., z_n)$, with each objective having a score $\vec{v} = (v_1, ..., v_j)$ on j criteria, we define the concept of Pareto optimality in the objective space as follows. An objective z dominates an objective z', denoted by $z \succ z'$, if

- $z_i(v) \geq z'_i(v)$ on each score of $\vec{v}_j$; and
- there exists one score v^* such that $z'_i(v^*) > z'_i(v)$.

Again, Pareto optimality implies that there is no feasible objective z_i such that v' does not

make it less effective in at least one other criterion.
Having gained a better understanding of what is deemed optimal, an algorithm that can efficiently compute the Pareto front from a set of given items is what we aim to develop in subsequent sections.

3.2. Data Model

The underlying data model for this thesis is a travel database curated by Wörndl and Herzog [6] in their previous work. The travel database contains realistic data on a region. Information about each region includes:

1. Minimum weekly budget for a region
2. Minimum duration to achieve a 25% and 75% utility respectively. An algorithm can either implement a minimum 25% threshold or a 75% minimum utility from stay threshold;
3. The score of a region for particular travel activity preferences on a five-point Likert scale;
4. The security score of a region on a five-point Likert scale; and
5. The monthly weather score of a region on a five-point Likert scale (i.e., recommended months).

The data model is hierarchically structured such that a region is always a sub-region of another region. The world is the root region, followed by the continents at the second level. Regions are geographical areas. A region could be a country, a section of a country, or a continent. For example Figure 3.2 illustrates the hierarchical structure of the data model. The US and Canada are sub-regions of North America, while Alaska and Texas are sub-regions of the US. Ontario is a sub-region of Canada. The database is composed of 196 regions. All regions (except the root region) have a recommendation for a minimum stay at 25%. Unlike previous models that calculate the connection between regions by specifying the necessary effort (time and cost) to travel from one region to another, we model connections between regions based solely on positions in the hierarchical tree structure. We extend the underlying data model to contain all sub-regions of a region. If two regions are sub-regions of the same parent, we consider them connected.

3.3. Analysis of Algorithmic Methodologies

According to Golden et al. [38], OP is an NP-hard problem. Hence, no known or expected algorithm can solve the problem optimally in polynomial time (i.e., an exact solution cannot be found within a reasonable amount of time). Computing an optimal solution with multiple objectives is even more intractable. Hence, there is usually a trade-off between time and

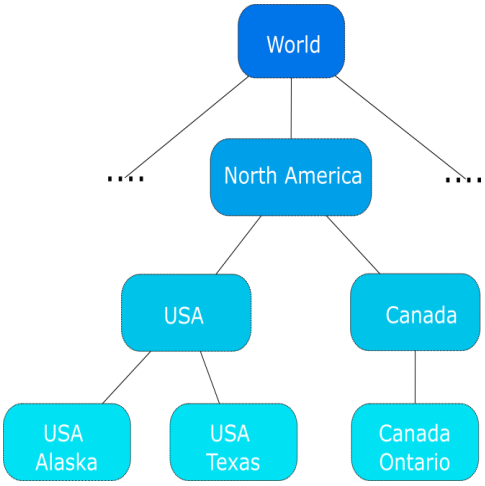


Figure 3.2.: Hierarchical structure of the underlying data model

accuracy. In multi-objective optimization, one is often interested in the Pareto front. However, it is usually infeasible to generate the exact set of a Pareto front. Reasons for this can be that the number of Pareto optima is too large, or the determination of a single Pareto optimum is computationally difficult. Therefore, the goal is usually to identify a satisfactory Pareto set approximation (i.e., a set as close to the optimum as possible [58]. Approximate algorithms offer near-optimal solutions to computationally intractable problems (mostly) in a moderate amount of time and are the major alternatives for solving NP-hard optimization problems.

Approximate algorithms can be classified as heuristics and meta-heuristics. Heuristics are usually problem dependent and inflexible (i.e., defined for a specific problem). Meta-heuristics are problem independent and can be applied to a broad range of optimization problems. In Chapter 2, we found that meta-heuristic solutions are commonly used to solve OPs. Meta-heuristic solutions are also commonly used in multi-objective optimization in various research areas. Certain requirements must be considered when designing or selecting an appropriate algorithm to solve the MOOP. Some requirements are intuitive for our problem domain. For example, a slow computational time by the end algorithm is intolerable for a user-oriented system. Some issues are not so intuitive. Preserving uniform diversity implies that the selected solutions in the approximated Pareto front are diverse and uniformly distributed. In addition, the MOOP has been defined such that a single objective represents a single user's preference. We are unable to pre-determine the number of preferences a user might select. A user could have just one preference. They might also have two or more preferences. Therefore, an appropriate algorithm should be applicable to a dynamic number of objectives. A brief summary of the most important requirements of the desired algorithm is as follows:

- Tolerable computational time,
- Applicability to problem domain

- Uniform preservation of diversity,
- Applicability to large-scale problems,
- Applicability to a dynamic number of objectives, and
- Moderate implementation complexity.

In developing an algorithm to solve our optimization problem, we first address the basic methodologies for designing an approximation algorithm. We then analyze possible heuristic and meta-heuristics solutions. In this chapter, we assume a maximization problem in which function g means f_k has to be maximized.

3.3.1. Greedy Methods

Greedy methods help find sub-optimal solutions that satisfy some performance guarantee in NP-hard problems. The idea is to generate a solution incrementally by making the best possible choice according to a simple criterion at each decision-making point. For example, a greedy approach to solving the simple knapsack problem is to iteratively select a node with the least weight. This approach is best used together with some form of heuristic rather than as a stand-alone method. When used alone, the heuristics generate only a minimal number of different solutions.

3.3.2. Local Search

Local search techniques range from simple heuristics to complex meta-heuristics. The basic idea behind a local search is to start from an initial (partial or non-partial) candidate solution. The solution is improved by making local changes to it and moving from one candidate solution to a neighboring candidate solution until no further improvement is possible. A local change might involve removing elements from the ground set, adding elements, or swapping elements. Local search is a good heuristic for computing near-optimum solutions to reasonably sized problems. Usually, a neighborhood function $N : S \mapsto 2^S$ where S is the set of feasible solutions specifies for each solution $s \in S$ a subset $N(s)$ of neighbors of s (neighborhood relation), or solutions that are close to s [59].

For many problems, computing the locality gap (i.e., the difference between local and global optima) is complex, and the locality gap using a local natural search is commonly very large. Additionally, the algorithm might visit the exact location within the search space more than once. Thus, it can become trapped in a location far away from the global optimum. These situations all lead to very poor approximation. Consequently, many local search techniques use mechanisms to help escape a position or reduce the locality gap. For example, one could use a restart strategy when stuck in a local minimum in the iterative descent algorithm. Alternatively, one could relax the improvement criterion by performing a non-improving step. However, these strategies do not guaranty escaping an arbitrary local minimum [60].

Iterative Improvement

The iterative improvement algorithm is the most basic local search algorithm and forms the basis of most local search algorithms. Algorithm 1 (*IterativeDescent)* produces a simple iterative improvement (also known as iterative descent). The solution of the iterative descent is the optimum local value, which may or may not be a global optimum. Condition $g(s') > g(s)$ is the evaluation of the candidate solution. solution. Evaluation functions serve as guidance to a solution [60]. In optimization problems, the objective function's properties usually determine the evaluation function. Below, the algorithm assumes a maximization objective function:

Algorithm 1: General outline of iterative improvement local search

Input : Set of candidate solutions S, Neighborhood function N, Objective function g
Output: A local optimum solution $s \in S$

determine an initial candidate solution s
while $N(s)$ *contains better solution than* s **do**
    choose a solution $s' \in N(s)$ such that
        $g(s') > g(s)$
    $s := s'$
end

The simple iterative improvement algorithm can be extended to multi-objective optimization problems. For multiple objectives, the solution of the iterative improvement and all local search-based algorithms, in general, is defined as the approximate set of Pareto optimal solutions (Pareto front) found during the search process. A move improves the current solution when a newly generated candidate solution is added to the approximate Pareto front. This means that the evaluation condition of the single-objective iterative improvement algorithm changes. The Pareto local search proposed by Paquete et al. [61] adapts the notion of Pareto optimality in a simple local search. First, the algorithm randomly selects one non-visited candidate solution s and examines all neighbors of s. Second, it adds all neighbors of s that are non-dominated by the set of candidate solutions. It stops when the neighborhood of all candidate solutions has been examined [59]. Similar to the Pareto local search, the two-criteria local search [62] is another multi-objective adaptation of the single-objective iterative improvement algorithm. While the Pareto local search chooses only one candidate solution to examine its neighborhood and immediately updates the list of candidate solutions, the two-criteria local search examines the neighborhood of all candidate solutions in an iteration.

3.3.3. Stochastic Local Search

Many local search algorithms use randomized decisions, for example to determine initial solutions or when determining search steps. These are called stochastic local search (SLS)

algorithms. The majority of algorithms for computing high-quality approximations to the Pareto optimal set are based on stochastic local search (SLS). Typically, additional memory is used to store the best candidate solution from previous searches. If the solution is feasible, it is returned upon termination of the algorithm; the objective function is used to determine the quality of the candidate solutions. The performance of any SLS algorithm depends significantly on the underlying neighborhood relations and the size of the neighborhood. Determining the choice of a neighborhood relation to use for SLS is mostly problem specific. Insertion and swap are two widely used neighborhood relation-generating operators. SLS techniques are attractive because they allow solving problems using moderately generic and easily implementable algorithms that can also be extended with or adapted based on problem-specific knowledge [60]. They range from simple constructive algorithms and iterative improvement algorithms to general algorithm frameworks adapted to a specific problem under consideration. Popular SLS algorithms include variable neighborhood descent (VND), simulated annealing (SA), tabu search (TS), EAs, ACO, and many others.

Randomized Iterative Improvement

Using large neighborhoods is particularly beneficial to escape the local minima. However, there is an associated time complexity with performing search steps in a vast neighborhood. SLS algorithms try to implement the concept of large neighborhoods, but with trade-offs. A popular idea in SLS algorithms is to use large neighborhoods but reduce their size by never examining neighbors that are unlikely to yield any improvements in evaluation function value

Algorithm 2 is a SLS-based iterative improvement to Algorithm 1. Algorithm 2 uses randomization as a diversification mechanism to improve the algorithm's quality. The ρ is a noise parameter that corresponds to the probability of performing a random step instead of an improvement step [60]. The search can be terminated, for example, after a certain number of steps or after no improvement has been achieved in many steps. This algorithm outperforms the pure local search version of iterative descent; when it is run long enough, an optimal solution to any given problem instance can be found [60].

Variable Neighborhood Descent

VND algorithms benefit from the advantages of large neighborhoods by reducing the time consumed in the search steps. In the descent phase, standard small neighborhoods are used until a local optimum is encountered. Then, in the perturbation phase, the search process switches to a different neighborhood, which might help exit the corresponding basin of attraction and allow further search progress. Algorithm 3 highlights the general outline of a VND. Unlike the previously outlined algorithms, N in this algorithm is a set $\{N_1, ..., N_{imax}\}$ of neighborhood relations, usually ordered according to the increasing size of respective local neighborhoods. Variations of the VND algorithm have been widely and successfully applied in the single-objective domain in practice and are known to show optimal results.

Algorithm 2: General outline of randomized iterative improvement local search

Input : Set of candidate solutions S, Neighborhood function N, Objective function g
Output: A local optimum solution $s \in S$

```
while termination condition not satisfied do
    if there exists a solution s' ∈ N(s) with probability ρ then
        choose solution s' ∈ N(s) uniformly at random
    else
        if there exists g(s') > g(s) where s' ∈ N(s) then
            choose solution s'
        else
            choose a solution s' ∈ N(s) such that s' is maximal
        end
    end
    s := s'
end
```

The algorithm can adapt to refinement and solution definitions in a multi-objective setting as defined in the iterative refinement algorithm in the previous section (see [63] for an example adaptation).

Algorithm 3: General outline of variable neighborhood descent

Input : Set of candidate solutions S, Set of neighborhood relations N, Objective function g
Output: A local optimum solution $s \in S$

```
while i < imax do
    choose the most improving s' of s ∈ N_i
    if g(s') > g(s) then
        s := s'
        i := 1
    else
        i := i + 1
    end
end
```

Simulated Annealing

SA is related to a probabilistic iterative improvement. It is based on the idea that the probability of accepting a deteriorating step should depend on the respective deterioration in evaluation function value, such that the worse a step is, the less likely it is that it will be performed [60].

The algorithm has a proposal mechanism in which a neighbor s' is chosen at random, and the acceptance criteria are based on statistical mechanics (metropolis condition) as follows:

$$\rho_{accept}(T, s, s') := \begin{cases} 1, & \text{if} \quad g(s') \geq g(s); \\ exp(\frac{g(s)-g(s')}{T}), & \text{otherwise;} \end{cases}$$

ρ_{accept} is the probability of accepting s' and is parameterized by the temperature T used to decide whether the search accepts s' or stays at s. The condition above depicts the case where T is kept constant and can be adjusted throughout the search process according to an annealing schedule. The technique for temperature adjustment used in simulated annealing helps escape the local minima. However, the results of simulated annealing are limited since it requires a cooling process that is typically slow in practice [60]. Additional techniques such as neighborhood pruning, greedy initialization, low-temperature starts, and look-up tables for acceptance probabilities are typically used to achieve competitive results.

SA algorithms have been used for multi-objective optimization. Such adaptations require modifying the acceptance criterion. Serafini [64] provides guidelines to apply in computing the probability ρ_{accept} as follows:

$$\rho_{accept}(T, s, s') := \begin{cases} 1, & \text{if} \quad \vec{f}(s') \succ \vec{f}(s); \\ [0, 1), & \text{if} \quad \vec{f}(s) \succ \vec{f}(s'); \\ exp(\frac{g(s)-g(s')}{T}), & \text{otherwise;} \end{cases}$$

Tabu Search

TS is a meta-heuristic based on adaptive memory and responsive exploration [59]. It guides a local heuristic search routine to explore the solution space beyond the local optimum. The algorithm forbids steps to recently visited search positions by preventing the local search from immediately returning to a previously visited candidate solution [60]. This prevention step can be implemented by explicitly memorizing previously visited candidate solutions and ruling out any step that would lead back to those. In contrast to a simple descent method, TS permits worsening steps, but the moves are selected from a modified part of the neighborhood. Hence, the neighborhood of s is not static. TS algorithms are highly efficient and successfully used in a wide range of fields, including OP. It is, however, crucial to carefully choose a neighborhood relation and to use efficient caching and incremental updating schemes for the evaluation of candidate solutions [60]. In TS for multi-objective optimization, the central idea is to examine the neighborhood of a set of solutions, extract non-dominated solutions, and accept only some non-tabu solutions for inclusion in the approximate Pareto front.

3.3.4. Evolutionary Algorithms

EAs are meta-heuristics whose methodologies are based on Darwin's theory of evolution, where an individual in a population (set of candidate solutions) is referred to as a chromosome. A gene represents the individual's properties. EAs follow the same schema as shown in Algorithm 4 [59] below. The basic principle behind EAs is the survival of the fittest, where a fitness function based on the objective function, constraints, or some quality measure is used to select individuals from the population. For each generation (iteration), individuals compete to produce offspring [65]. An EA might use a crossover operator to recombine two or more individuals to produce new individuals for the next generation. A mutation operator might also be used to alter the gene (characteristics of an individual) of a chromosome. The used reproduction operators depend on the chosen solution representation and the problem formulation. For example, one-point crossover, uniform crossover, and flip mutation are commonly used for representing binary string solutions. In contrast, binary string encoding is typically used for knapsack problems [59].

Algorithm 4: General outline of evolutionary algorithms

```
1 P = apply τ on G to generate μ individuals (the initial population);
2 while termination criteria not met do
3 |   P' = apply θ on P;                                   /* selection */
4 |   P'' = apply ω_r on P'; r ∈ {1,..,noperators} ;       /* reproduction */
5 |   P = apply ψ on P and P'';                           /* replacement */
6 end
```

EAs are popularly used across different industries, and Pareto-based EAs are particularly suited for multi-objective optimization. Unlike traditional techniques such as TS, SA, and VND that originally output a single local optimum, Pareto-based EAs typically provide the whole set of Pareto optimal solutions in a single run, making them suitable for our problem domain. Examples of Pareto-based EAs in literature include the non-dominated sorting genetic algorithm (NSGA), strength pareto evolutionary algorithm (SPEA) and pareto memetic algorithm (PMA).The performance of these algorithms relies on specifying a robust fitness function and estimating a good fitness-sharing parameter. The fitness-sharing parameter helps adjust an individual's fitness based on the fitness of others. Furthermore, as in most population-based meta-heuristics, determining the initial population is important. If the initial population is not diverse, premature convergence might occur [66].

3.3.5. Ant Colony Optimization

ACO is a meta-heuristic that is part of swarm intelligence and inspired by ants. Artificial ants are designed to simulate the problem-solving behavior of ants in a colony. Ants coordinate their activities *stigmergy*, an indirect communication form through modification of the environment. The idea behind ant algorithms is to use a form of artificial stigmergy

to coordinate societies of artificial agents [67]. An effective solution is found only through cooperation among many individuals in the colony. Inspired by the pheromones deposited by real ants, virtual ants are designed to modify numeric values (virtual pheromones) associated with different problem states. The sequence of pheromone values associated with problem states (pheromone trail) enables communication. Ants can also forget the pheromone trail history and focus on new promising search directions through an evaporation mechanism [59].

Solutions are created incrementally by moving through available problem states and making stochastic decisions at each step. Additionally, many improved and efficient ACO algorithms make use of local searches to improve the probability of an ant choosing a component and consequently improve the quality of the solution constructed by the ants [68]. ACO can be applied to any combinatorial optimization problem for which a constructive heuristic can be defined. However, the problem must be mapped to a representation that artificial ants can use to build solutions [67]. Specifically, the problem must be representable as a construction graph $G_c = (C, L)$, where the set L fully connects the components C. The ants exploit G_c to search for an optimal solution. Fortunately, knapsack problems can be represented as a construction graph by representing the set of items as the set of components and fully connecting them. Therefore, ACOs algorithms can be applied to our problem domain. Nevertheless, the pheromone trails, heuristics information, and solution construction must be carefully considered for the MOOP. Strategies for using ACO on multi-objective problems include aggregating the objective functions and using one ant colony with one pheromone structure or using a different colony for each objective function and having multiple pheromone structures [69].

3.3.6. Bee Colony Optimization

Bee colony optimzation (BCO) is a meta-heuristic that represents a type of swarm intelligence inspired by bees. Artificial agents are created by partially simulating the real-life behavior of bees. Such behaviors include nectar exploration, mating during flight, food foraging, waggle dancing, and division of labor. Bee colony-based optimization algorithms are mainly based on food foraging, nest site search, and mating in the bee colony [66].

The ACO algorithm is based on nest site search behavior and consists of alternating forward and backward passes. Every artificial bee explores the search space during the forward pass. It applies a predefined number of moves that construct and improve the solution, thus yielding a new solution. After obtaining new partial solutions, the bees return to the nest and begin the backward pass in which all the artificial bees share information about their solutions [70]. The bee whose solution has the highest fitness score is selected to form the next bee population.

3.4. Designing Algorithms for Multi-Objective Problems

In the above sections, we define our optimization problem as a combinatorial multi-objective problem. Solving MOOP implies obtaining an approximation set of Pareto optimal solutions in such a way that the set fulfills the requirements of convergence to the Pareto front and uniform diversity [66]. Exact methods such as branch and bound algorithms, constraint programming, and dynamic programming are used for two-criteria optimization problems. However, such methods are better suited for small-scale problems. In research, optimization problems with more than three objectives are often classified as multi-objective problems. This is due to the added difficulty of handling a higher number of objectives. Unlike many objective problems, two-objective and three-objective problems can be comprehensively visualized by graphical means, which makes it easier for decision makers to analyze and make better decisions [71]. Heuristics methods are needed for this problem scale, and designing heuristics for MOOP requires additional concepts. The following sections summarize the concepts typically required for designing algorithms for MOOP.

3.4.1. Fitness Assignments

The fitness assignment measures the quality of a solution by assigning a scalar-valued fitness to a vector objective function. This procedure can be classified into scalar approaches, criterion-based approaches, dominance-based approaches, and indicator-based approaches.

Scalar Approaches

Scalar approaches typically transform the MOOP into single-objective problems. A scalar approach frequently used in designing solutions for multi-objective optimization problems consists of aggregating the objective functions into a single function using either addition, multiplication, or any combination of arithmetic operations. This approach has the advantage of simplifying the objective space such that only a single objective needs to be optimized. However, in order to avoid one function dominating another, aggregating the functions requires behavioral knowledge of each objective function [72]. The most common aggregation method is the weighted sum approach, which is defined as follows:

$$\vec{f} = \sum_{i=1}^{k} w_i f_i, \text{ where } w_i \geq 0 \text{ and } \sum_{i=1}^{k} w_i = 1$$

where w_i are coefficients representing the objectives' relative importance, which is usually unknown. The algorithm designer must solve the problem for different values of w_i and decide on the appropriate value for w_i based on their intuition. Constant multipliers c_i are often introduced to further reflect the importance of each objective. This helps normalize the vector function in approximately the exact numerical values. Other popular scalar approaches

found in the survey by Carlos et al. [72] use goal programming (minimize or maximize the absolute deviation from target to objectives) and ε-constraint methods (considering the objectives bound by some allowable levels ϵ_i). However, transforming a MOOP into a single-objective problem is not always feasible. Ascribing a hierarchy of importance to the objectives is a crucial step in aggregating the function. In our problem domain, the objective function already represents a user preference (i.e., each objective function is mapped to a particular user preference). Ultimately, assigning a level of importance to each objective function defeats the optimization problem's purpose because the objectives are not comparable.

Criterion-Based methods

Criterion-based methods are commonly used in population-based meta-heuristics (e.g., EAs and ACO). They conduct the search process by treating the various objectives separately and assigning them fitness values. This procedure usually occurs in parallel or sequentially. For example, parallel approaches are used in ACO (P-ACO [73]), where one ant colony tackles an objective. Sequential approaches search sequentially in a defined preference order; the order signifies the importance of each objective function [74].

Dominance-Based approaches

Dominance-based (or Pareto-based) approaches use dominance and Pareto optimality to guide the search process. In a single run, such approaches can generate a diverse set of Pareto optimal solutions and Pareto solutions in the concave portions of the convex hull of feasible objective space [66]. Most Pareto approaches use EAs. Compared to scalar methods, Pareto-based fitness assignment evaluates the quality of a solution in relation to the whole population by applying ranking methods. Ranks are scalar values obtained using dominance relation techniques. The rank assigned to a solution is considered its fitness value.

3.4.2. Diversity Preservation

Initial population selection and biased sampling during a search in population-based meta-heuristics are crucial for maintaining diversity. Diversity-preserving methods must be considered when designing meta-heuristics for MOOP; they generally penalize solutions that have high density in their neighborhoods. According to [75], diversity preservation strategies can be classified into kernel, nearest neighbor, and histogram methods.

Kernel methods Kernel methods define the neighborhood of a solution according to a kernel function that takes the distance between solutions as an argument. The kernel function is applied to all distances. For example, fitness sharing is popularly used in EAs as it degrades the fitness of an individual using a sharing function *sh*. The sharing function depends on a constant σ and the sum of the distance between individuals in the population. σ represents the non-similarity threshold. Nearest neighbor methods estimate the density of a solution

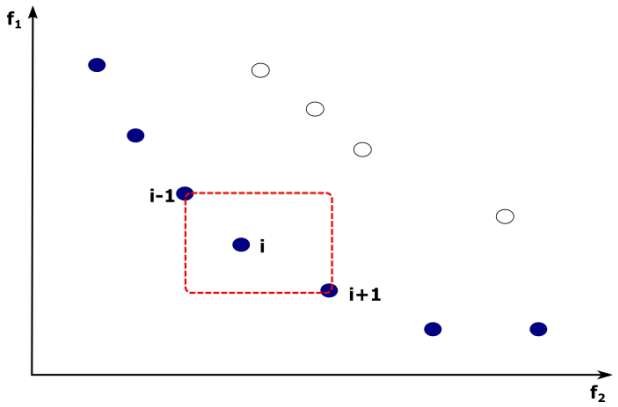


Figure 3.3.: Crowding distance sorting in NSGA-II

while taking into account the distance between a solution and its k^{th} nearest neighbor. For example, NSGA-II uses crowding distance sorting to sort solutions within a rank. A crowding distance of an individual i is illustrated in Figure 3.3. The crowding distance can be pictured as a cuboid on the i and its two nearest neighbors (left and right). Equation 3.1 shows the formula used to find the crowding distance for i in an objective f_k from F objectives; it is defined as the circumference between a solution and its left and right neighbors.

$$distance(i) = distance(i) + \frac{f_k(i+1) - f_k(i-1)}{f_k^{max} - f_k^{min}} \qquad \forall\ f_k \in F \tag{3.1}$$

More diversity is obtained using solutions with high crowding distances. Histogram-based diversity-preserving approaches partition the search space into several hyper-grids that define the neighborhood. The density around a solution is estimated by the number of solutions in the same box of a grid [66]. The technique used to measure the distance between solutions is essential for diversity preservation. Diversity in the decision space may be essential to improve the search for some problems; however, some only require diversity in the objective space.

3.5. Comparative Analysis of Choice Algorithms

So far in this chapter, we have extensively discussed different algorithmic approaches that are commonly used to solve combinatorial optimization problems. We have presented algorithms that are typically suited for MOOPs. Through our analysis, we can conclude that using traditional techniques such as local search, tabu search, etc., will be more difficult to implement for our MOOP because such techniques yield a single local optimum. A single-objective optimizer needs to be run multiple times when using such techniques. Additionally, good distribution (uniform diversity) is not guaranteed. Thus, a considerable amount of

adaptation is required to use these approaches for our problem domain. In contrast, modern algorithms such as the EAs, ACO, and BCO are best suited for multiple objective problems because they can return the Pareto optimal set in a single run. They can also be easily decomposed to find the optimal solution for single objectives. Hence, they do not require more effort than necessary to be implemented for our problem domain.

It is impossible to implement all algorithms considered suitable for our problem domain in this thesis. Thus, we compare EAs and ACO. These choices are based on their popularity for solving OPs, multi-objective knapsack problems, and multi-objective problems in general.

Table 3.1.: Comparison of choice algorithms.

	Evolutionary Algorithm	Ant-Colony
Computational time	✓	
Applicability to problem domain	✓	
Approximate Pareto front	✓	✓
Diversity Preserving	✓	✓
Applicable to large-sized objectives		✓
Easier to Implement	✓	

In table 3.1, EAs and ACOs are compared using the desired algorithmic properties listed in Section 3.3 of this chapter. The capabilities of widely used EA variants like NSGA-II, NSGA-III, SPEA2 were also taken into account. These scores are based on evaluation results from [69], [76], [77]. as multi-objective knapsack problems were used for evaluation in these studies. A checkmark is awarded when the algorithm has an advantage over the other in that criterion. For example, in comparison to ACOs, EAs have demonstrated faster computational time in solving multi-objective knapsack problems. However, it is unknown which of the two algorithms for approximating to the Pareto front works best. ACO is specifically designed to solve path-search problems. A graph representation of the knapsack model must be constructed, which increases the implementation complexity of the ACO. Unlike ACO, a number of EA variants can be used directly for our optimization problem without adaptation. Moreover, there are several frameworks focusing on implementing EAs that significantly decrease the time needed for implementation.
We can conclude that an EA is the best algorithmic approach for our study.

4. Design and Implementation

Implementing an EA requires a number of considerations. A robust fitness assignment method needs to be determined. A mechanism for uniform diversity preservation must be conceptualized, and a well-thought-out solution to address constraints in the problem definition must be designed. There are successful EAs that can be used for this; these algorithms have been well researched, and their domain applicability is well defined.

NSGA-II [78] is an EA that uses an elitist non-dominated sorting mechanism to approximate the Pareto front. It is an improved version of an initial NSGA algorithm developed by Srinivas et al. [79] in 1996. NSGA-II uses crowding function for diversity preservation and ranking of the Pareto fronts. However, it is better suited for two-objective problems and does not perform well on multi-objective problems. NSGA-III [80] was introduced to address the inability of NSGA-II to solve multi-objective problems (i.e., three or more objectives).

Other successful EAs applicable to our problem can be found in practice. In the following sections, we implement the NSGA algorithm and its heuristic adaptations to suit our problem definition.

4.1. Non-Dominated Sorting Genetic Algorithms

Designing domination-based evolutionary multi-objective (EMO) algorithms for multi-objective problems requires additional considerations. For instance, an increase in the number of objectives may cause an increase in the non-dominated population derived from the randomly generated population. Since domination-based EMO prioritizes non-dominated solutions, the solution space becomes rather large. Therefore, the search process becomes slow, and the algorithm becomes inefficient. With increased solution space also arises an increase in the number of neighbors. Evaluation of diversity measures (i.e., crowding distance) becomes computationally expensive. NSGA-III was designed to address these problems.

As defined in 3.1, our problem can be mapped to many objective optimization problems. Each objective function is a function of a POI and its characteristics. A chosen algorithm needs to efficiently compare more than three objectives, and this informs the decision to implement and adapt the NSGA-III to our problem domain. Next, we explore the aspects that comprise the NSGA-III and constitute our algorithm.

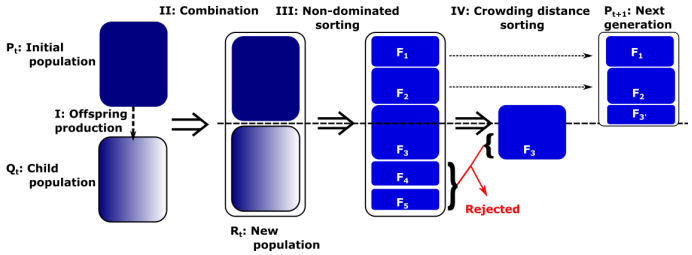


Figure 4.1.: NSGA-II overview

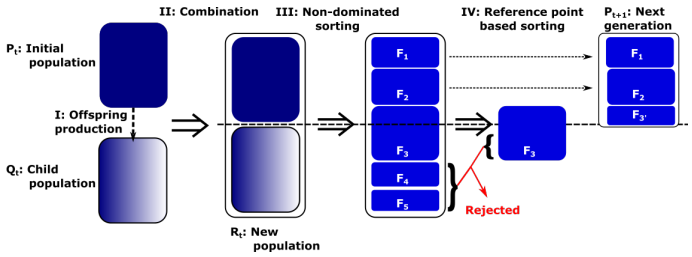


Figure 4.2.: NSGA-III overview

The NSGA-III Algorithm

The NSGA-III algorithm was developed to address the inefficiency of NSGA-II in handling multi-objective problems. In its most basic form, NSGA-III can be seen as the NSGA-II algorithm with minor changes; that is, most algorithmic steps in NSGA-III are from the NSGA-II algorithm. Figure 4.1 and 4.2 depict an overview of the major algorithmic steps in NSGA-II and NSGA-III, respectively. From these figures can be observed the similarity between the two algorithms. However, a major difference between them is in the last step of next-population generation. Unlike NSGA-II, which sorts individuals by crowding distance to decide what part of a Pareto front to reject or accept, NSGA-III uses a sorting technique based on a reference point. Sorting by crowding distance is explained in Section 3.4.2. The reference-point-based sorting technique and other intrinsic details of the NSGA-III are explained below.

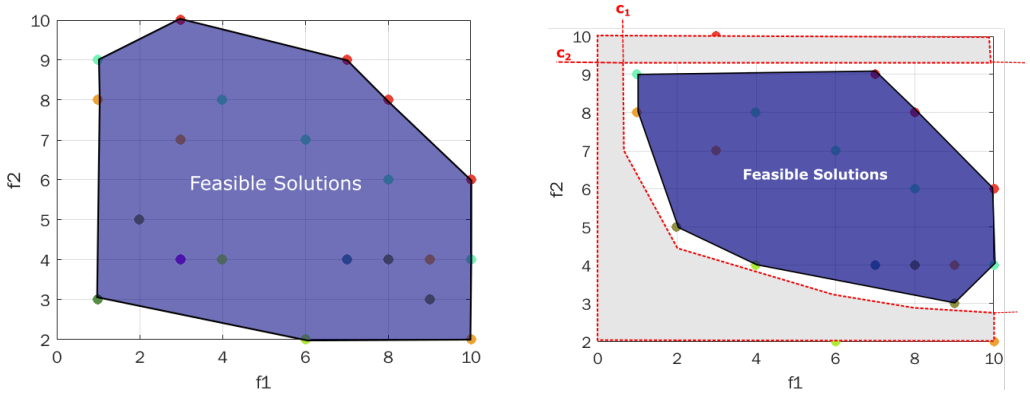
Initial Population

Let us consider P_t as a parent population of the t^{th} generation of the algorithm (i.e., iteration t). P_t with N chromosomes is either a population taken from previous generation $t-1$ or an initial population taken from a defined heuristic.
The heuristic used for initial population generation at the start of the algorithm is particularly important for our problem. Simply sampling randomly from the problem space would be logically inefficient. The number of objective problems in practice is typically capped at 15–20. NSGA-III was evaluated on 20-objective DTLZ1-DTLZ2. Considering that the objective function in our problem domain comprises a single POI and its characteristics, an intelligent

pre-selection process must be defined to reduce the problem space.

Constraint Handling

Figure 4.3a illustrates a typical solution search space for an unconstrained problem. For unconstrained problems, all members of the population are regarded as feasible; however, some individuals can become infeasible (i.e., members in the gray shaded area; Figure 4.3b), some individuals become infeasible (i.e., members in gray shaded area). Additional techniques are required for handling constraints in NSGA-III. Two papers address problems with NSGA-III both with [71] and without [81] constraints.



(a) Unconstrained solution search space (b) Constrained solution search space

Figure 4.3.: Solution Search Space

Consider an individual x in the solution space with the following constraints:

$$g_j(x) \leq b_j \qquad \text{(constraint 1)}$$
$$h_k(x) = 0 \qquad \text{(constraint 2)}$$

The authors of [81] suggest calculating a constraint valuation value $CV(x)$ to measure the degree to which x violates the constraints using the following formula:

$$CV(x) = \sum_{j=1}^{J} \langle g'_j(x) \rangle + \sum_{k=1}^{K} |h'_k(x)| \qquad (1)$$

where $g'_j(x)$ is the normalized constraint function for Constraint constraint 1 , such that $g'_j(x) = g_j(x)/b_j - 1 \leq 0$ and $h'_k(x)$ is the normalized constraint function for Constraint constraint 2. The angle bracket operator $\langle \alpha \rangle$ returns the negative of α, if $\alpha < 0$ and returns

zero, otherwise. Assuming that Constraint constraint 1 is a greater or equal constraint (i.e., $g_j(x) \geq b_j$), then $\langle \alpha \rangle$ returns the negative of α, if $\alpha > 0$ and returns zero, otherwise.

The notion of feasibility together with the constraint violation measure $CV(x)$, are combined to derive a constraint domination principle. Consider two individuals x_1 and x_2 in the solution space. An individual x_1 is said to constraint dominate x_2 if any of the following conditions holds:

1. x_1 is feasible and x_2 is infeasible;
2. Both x_1 and x_2 are infeasible and x_1 has a lower constraint violation (CV) value; or
3. Both x_1 and x_2 are feasible and x_1 dominates x_2 ($x_1 \succ x_2$).

Offspring Production

Three major aspects in generating the child population Q_t are defined as follows:

1. **Tournament Selection**: To select two individuals that will be eligible to produce offspring for the next generation, a tournament selection procedure is carried out. Two individuals are selected at random from P_t, and a binary selection of the better solution is applied, as shown in Algorithm 5. Individuals p_1 andp_2 are compared to select the one that does not violate the constraints. If both individuals violate the constraints, the constraint violation values for the two individuals are computed. Then, the algorithm selects the individual with the lesser constraint violation value. Two parents for reproduction (crossover and mutation) are selected using this tournament procedure.
2. **Cross over**: A crossover operator is used to generate offspring according to a crossover rate p_c. The crossover rate represents the proportion of parents on which a crossover operator will act. A uniform crossover ensures that each offspring will inherit some characteristics of both parents. Either the 1-point crossover or the n-point crossover is commonly used for binary representations. When using the 1-point crossover, a crossover site k is randomly selected. is randomly selected. Then, two offspring are created by exchanging the segments of the parents [66]. Consequently, n-point crossover implies n crossover sites.
3. **Mutation**: Mutation operators act on a single individual. An element of the representation is mutated (changed) according to some probability p_m. [66] recommends small values for p_m ($p_m \in [0.001, 0.01]$). A flip operator is commonly used when using binary representations of individuals.

During reproduction, duplicate offspring are checked and eliminated. A child population Q_t of size N is generated by repeating tournament selection and reproduction. R_t is generated by combining P_t and Q_t (i.e., $R_t = P_t \cup Q_t$). NSGA-III and NSGA-II use an elitist preservation technique (III and IV in figure 4.2) to preserve elite members of the new population.

Algorithm 5: Tournament selection procedure in constrained NSGA-III

Input : p_1, p_2
Output: p'

```
if feasible(p1) and not feasible(p2) then
    p' = p1
else if not feasible(p1) and feasible(p2) then
    p' = p2
end
else if not feasible(p1) and not feasible(p2) then
    if CV(p1) > CV(p2) then
        p' = p2
    end
    else if CV(p1) < CV(p2) then
        p' = p1
    end
    else
        p' = random(p1, p2)
    end
end
else if feasible(p1) and feasbile(p2) then
    p' = random(p1, p2)
end
```

Non-Dominated Sorting

Non-dominated sorting ranks members of the population R_t into different non-dominated levels (F_1, F_2, and so on). In optimization problems without constraints, the sorting is performed using individuals fitness values, while in optimization problems, individuals are ranked according to their constraint domination count. We introduce a new symbol $\succ_+$ for constraint domination. For example, let A, B, and C be selected members of the population such that $A \succ_+ B \succ_+ C$ (i.e., individual A constraint dominates individual B and C, individual B constraint dominates C). Figure 4.4 depicts A, B, and C on a graph. A has the highest domination count and hence belongs to domination level F_1. B and C belong to level F_2.

Thereafter, the number of feasible solutions N_f in R_t are counted. If $N_f \leq N$, all the feasible solutions are added to S_t to start the next generation P_{t+1}. The remaining $K = N - |P_{t+1}|$ population slots are filled with top-level infeasible solutions, that is solutions having smaller CV values.
If $N_f > N$, the algorithm follows the unconstrained selection procedure described in [71]. Each level is selected one at a time until a size N is reached or N is exceeded for the first time. Assuming the l^{th} level is the level that causes the size of S_t to exceed N, then solutions

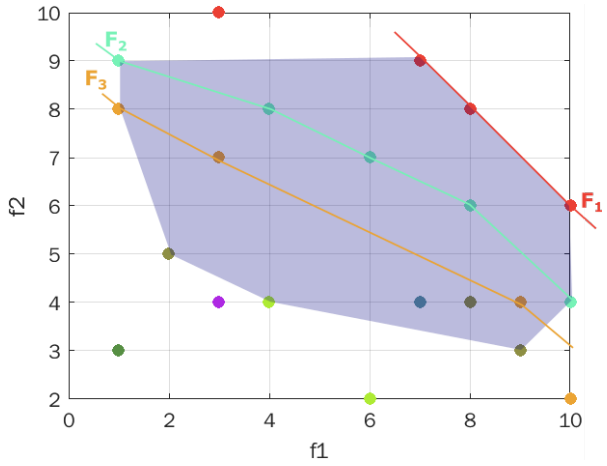


Figure 4.4.: Ranking of Pareto Frontier according to Constrained Domination

in level l are only partially accepted into S_t.

Reference Point-Based Sorting

NSGA-III uses a pre-defined set of reference points to ensure diversity in solutions. The reference points can either be preferential points supplied by the user or pre-defined in a structured manner [71].
In the case of structured reference points, NSGA-III uses a systematic approach that creates points on a normalized hyperplane. Points are placed on the hyperplane with the same orienteering in all axes. The total number of reference points (H) is given by the following:

$$H = \binom{M + p - 1}{p}$$

where M is the number of objective functions, and p is the number of divisions to consider along each objective. For example, using three objectives ($M = 3$) and four divisions ($p = 4$), the reference points are placed on a triangle with its apex at $(1, 0, 0), (0, 1, 0)$, and $(0, 0, 1)$. $H = \binom{6}{4}$, so 15 reference points are created. The obtained reference points are widely distributed on the entire normalized hyperplane, and the solutions are widely distributed on or near the Pareto-optimal front. Population members associated with the created reference points are emphasized. The population size is recommended to be the smallest multiple of four that is greater than H. In the case of user-supplied reference points, the user can mark H points on the normalized hyperplane.

After determining the reference points on a hyperplane, the algorithm adaptively normalizes the population members. First, the ideal point[1] of the population S_t is determined by

[1]The algorithm presented in the paper focused on minimization problems, thus the ideal point was differently defined. Since, we are faced with a maximization problem, our ideal point is defined differently.

identifying the maximum value (z_i^{max}) for each objective function $i = 1, 2, ..., M$ in $\cup_{\tau=0}^{t} S_\tau$ and by constructing the ideal point $\tilde{z} = (z_1^{max}, z_2^{max}, ..., z_M^{max})$. Using $f_i'(x) = z_i^{max} - f_i(x)$, each objective value of S_t is translated. Thereafter, the extreme point (z_i^{ext}) in each objective axis is identified. Given a weight vector d_i, that is close to the i^{th} objective axis, an achievement scalarizing function (ASF) is given by the following:

$$ASF(x, d_i) = max_{k=1}^{M} \frac{f_i'(x)}{d_i}, \qquad for\ i = 1, 2, ..., M.$$

z_i^{ext} is identified by finding a solution ($x \in S_t$) that results in the minimum corresponding ASF. These M extreme vectors are then used to represent an M-dimensional hyperplane. The intercept a_i of the i^{th} objective axis and the linear axis can then be calculated by $a_i = z_i^{max} - z_i^{ext}$. These maximum and extreme values are updated at every generation using the current population. Putting this all together, the objective function can be normalized using the following:

$$f_i^n(x) = \frac{z_i^{max} - f_i(x)}{z_i^{max} - z_i^{ext}}, \qquad for\ i = 1, 2, ..., M. \tag{4.1}$$

After normalizing the objective functions, each population member is associated with a reference point. A reference line for each reference point on the hyperplane is defined by joining the reference point with the ideal point. Then, a perpendicular distance between each population member and each reference line is calculated. The reference point that has the closest reference line to a population individual in the normalized objective space is considered to be associated with the population individual.

A niche-preserving procedure to generate the remaining $K = N - |P_{t+1}|$ population slots is performed. population slots is performed. A reference point may have one or more population members associated with it, or there might not be any associated reference point. Initially, the population members from $P_{t+1} = S_t \setminus F_l$ that are associated with the j^{th} reference point are counted as ρ_j (niche count). Then, the reference point set $J_{min} = j : argmin_j\ \rho_j$ having minimum ρ_j is identified. In case of multiple such reference points, a random $\tilde{j} \in J_{min}$ is selected from J_{min}. If there is no associated P_{t+1} member to the reference point $\tilde{j}$ ($\rho_{\tilde{j}} = 0$), then either of the following scenario is possible:

1. There exists one or members in front F_l that are associated with the reference point $\tilde{j}$, or
2. The front F_l does not have any member associated with the reference point $\tilde{j}$.

In case 1, the first case, the reference point with the shortest perpendicular distance from the reference line is added to P_{t+1}; then, the count $\rho_{\tilde{j}}$ for the reference point $\tilde{j}$ is incremented by 1. In case 2, the reference point is excluded from further consideration for the current generation.
In the event that $\rho_{\tilde{j}} \geq 1$, a randomly chosen member (if one exists) from the front F_l associated with the reference point $\tilde{j}$ is added to P_{t+1}. The count $\rho_{\tilde{j}}$ for the reference point $\tilde{j}$ is incremented by 1. The procedure is repeated K times until all vacant slots in $\rho_{\tilde{j}}$ are filled.

4.2. Implementation and Algorithmic Approaches

In this section, details of the implementation and several approaches to solving the many objective OPs discussed in Section 3.1 are presented. Our approach consists of an optimization process that comprises varying initialization steps and constraint-handling techniques. Algorithm 6 outlines the general structure of the approach, which addresses both initialization and the constraint-handling technique. It is important to note a fundamental difference between the non-dominated sorting explained in Section 19 and outlined in Algorithm 6. Our approach performs a non-dominated sorting of the fitness values of individuals, as it is performed in the case of NSGA-III without constraints. Additionally, we have chosen a criterion-based fitness assignment for our approach. Therefore, non-dominated sorting is solely used for approximating the Pareto front, in our implementation.

The implementation part of this thesis is realized using *Python*. *Pandas* [82] and *NumPy* [83] are Python packages that provide functionalities for storing and processing the data. The dataset used for this work is stored in a local MySQL database. NumPy and Pandas are used to transform the dataset after querying stored data from the database. During runtime, the transformed dataset is stored as Pandas tables. An existing framework containing NSGA-III is selected to reduce implementation time. The evolutionary computation framework DEAP [84] is chosen because of the flexibility of use it guarantees via explicit algorithms and transparent data structures. It can also be used with the Python multi-processing module in order to bypass the Python GIL and parallelize the optimization process. Due to the high dimensionality of the considered problems, efficiency through parallelization provides a benefit for the calculations in test cases. Lastly, DEAP is an open-source framework with a vast number of examples, which helps better understand the framework.

The initialization and constraint-handling technique of the optimization process is realized in different ways. The following sections describe several variations that are implemented and evaluated in this thesis.

4.2.1. Population Encoding

Individuals of the population are encoded as binary arrays. An example individual of the population is encoded as $[1, 0, 1, 1]$ where each bit represents a region under consideration. Thus, a population is a nested array of individuals. For example, a population of two individuals is encoded as $[[1, 0, 1, 1], [1, 1, 1, 1]]$. Each individual is also assigned a so-called *strategy*; an array of index positions in the randomly selected regions that helps in correctly mapping the bits to the appropriate region.

4.2.2. Start Population

The first step in each approach is to initialize the first population from which the EA begins. The performance of the algorithm can vary depending on the starting population

Algorithm 6: General outline of optimization process

Input : In Initialization, CHT Constraint Handling Technique, M No of Objectives
Output: Individuals in first non-dominated front

```
Calculate the number of reference point (H) to place on the hyper-plan using p and M
Generate the initial population via method defined by In
Evaluate fitness of individuals depending on CHT
Realize the non-dominated sorting of fitness values
while # generations ≤ MAXGEN do
    Select two parents P1 and P2 using the tornament method
    Apply crossover between P1 and P2 with a probability P_c
    Evaluate fitness of individuals depending on CHT
    Realize the non-dominated sorting of fitness values
    Normalize the individuals
    Associate individuals with the reference points
    Apply the niche preservation operation
    Keep the niche obtained solutions for the next generation
end
Evaluate fitness of individuals depending on CHT
Realize the non-dominated sorting of fitness values
return feasible individuals belonging to the first non-dominated front
```

used. A main criterion is good diversity at the beginning of the optimization process. Four initialization techniques are used in this thesis. The first is a naive method that randomly generates individuals for the start population. We adapt the metric used in [6] to create a second method that generates individuals having high scores on one or more of the defined constraints. The third method uses only feasible individuals for the starting population. The last method is a hybrid of similarity-based and feasibility-based initialization.

Random Initialization

The random initialization technique is a naive approach used by the baseline NSGA-III algorithm. To generate an individual, a binary array of all 1s is generated via NumPy. Then, the flip-bit operator provided by DEAP is used to randomly flip the bits of the array. Variant 1 in Figure 4.5 shows the results of the comparison between the number of feasible and infeasible individuals generated for the initial population using this approach. The preliminary test was designed to generate 92 individuals for the starting population, and it was repeated 100 times. On average, only 0.81% of individuals were feasible.

Similarity-Based Initialization

Similarity-based initialization incorporates the similarity metric used in [6] and is performed in two steps:

1. Rank the regions according to similarity metric, and
2. Randomly generate individuals from regions with higher rank

In step 1, a similarity value between the desired score for preference f_q and the actual score of preference f_c is computed as follows:

$$sim_{feature}(f_q, f_c) = 1 - \frac{|f_q - f_c|}{max(f_q, f_c)}$$

where f_c is the normalized score in a range of $[0, 1]$. The desired score for each input preference is defined as 1 ($f_c = 1$). A ranking of all regions computed using the weighted sum average of $sim_{feature}$ is then computed using the below metric:

$$similarity(q, c) = \frac{\sum_{i=1}^{n} w_i * sim_i(q_i, c_i)}{\sum_{i=1}^{n} w_i}$$

The weight w_i prioritizes specific preferences over the other. For example, activity preferences selected by the user have higher priority than other influential parameters such as safety. In our implementation, we prioritize user activity preferences and weather for the chosen months. Safety from crime has second priority, hence it has less weight. We do not consider parameters such as budget or minimal stay duration for each region for the similarity metric. After obtaining a ranking of similarities between the regions and preferences, regions having ranks below 0.7 are not considered for the next steps. For more information on the similarity metric, refer to [6].

In Step 2, the random initialization described in Section 4.2.2 is used to generate the starting population. Variant 2 in Figure 4.5 shows the results of the comparison between the number of feasible and infeasible individuals generated for the initial population using this approach. Test parameters similar to those in Variant 1 are used. Results show that this approach does not improve the proportion of feasible to infeasible individuals in an initial population (0.5% average count of feasible individuals).

Feasibility-Based Initialization

A third approach to generating the initial population is applying a feasibility-based initialization. In this method, a randomly generated individual is first tested for its adherence to all defined constraints. An individual is deemed infeasible if it does not meet all constraint criteria. Feasible individuals are added to the starting population, while infeasible individuals are discarded. This process is repeated until a population size N is attained. An issue with

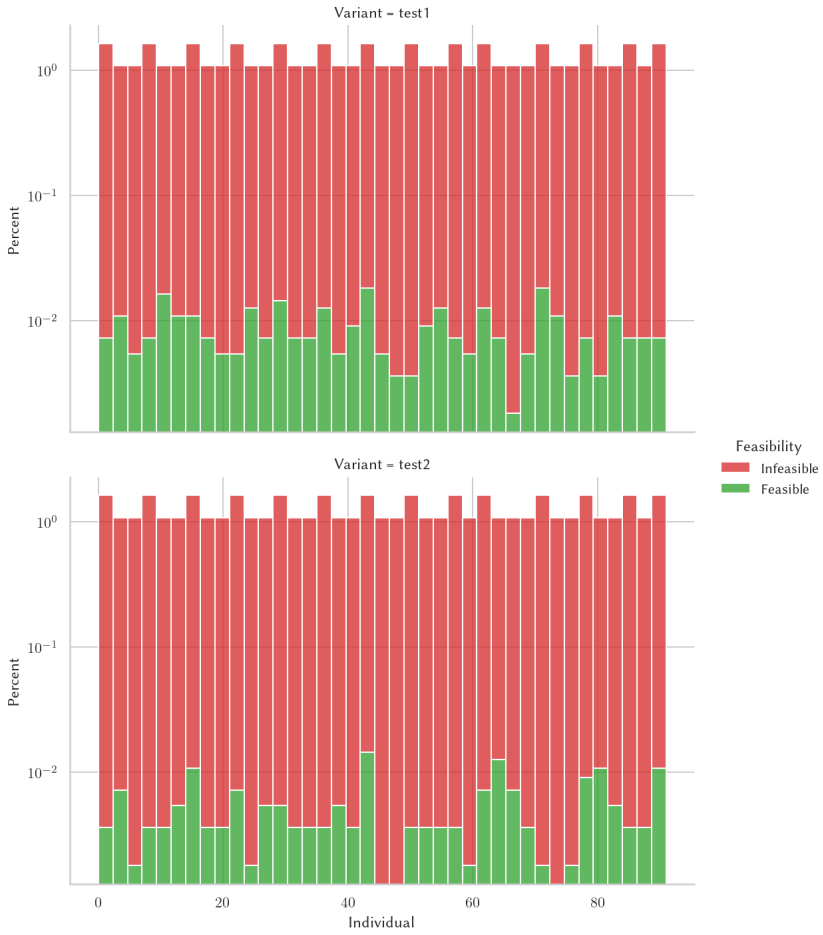


Figure 4.5.: Resulting initial population distribution of 92 randomly initialized individuals (100 trials)

this approach is the large amount of time needed to generate population size N. Depending on the population size N and the number of objectives used, more computational time is often needed to obtain N feasible individuals. Hence, algorithm variants using this approach are assigned a smaller population size.

Hybrid Initialization

The hybrid approach to initial population generation implemented in this thesis comprises feasibility-based and similarity-based initialization. In this approach, the similarity metric described in 4.2.2 is first applied to generate a ranking of the regions. Regions having a rank below 0.7 are removed from further consideration. Next, possible region combinations are randomly generated, and only feasible individuals of size N are allowed into the starting population.

4.2.3. Constraint Handling Technique

Handling of constraints and evaluation of the fitness of individuals in DEAP are tightly coupled due to the interfaces provided in the framework. In this section, two approaches implemented for handling constraints are described. In both approaches, the constrained objective function is transformed into an unconstrained objective function by applying penalty functions.

Distance-Based Penalties

In this approach, the fitness value of infeasible individuals is replaced with a penalized fitness value $f_i^{\text{penalty}}(x)$ using the penalty metric shown below:

$$f_i^{\text{penalty}}(\mathbf{x}) = \Delta_i - \sum_{j=1}^{N_{con}} d_{ij}(\mathbf{x})$$

where Δ_i is the smallest possible score any individual of the population could obtain. In our case, Δ_i is set as $[0, 0, 0]$. N_{con} represents the total constraints defined in our problem formulation (i.e., $N_{con} = 5$). d_{ij} sums the distance of the individual to the feasible region for each constraint j. Depending on the constraint, each region that comprises an individual is penalized independently. Specifically, for the budget (2) and duration (3) constraints, d_i^k for a specific region in individual i is computed as follows:

$$d_i^k = \frac{c_k}{CV_i(x) * q_k}$$

$CV_i(x)$ is the sum over the normalized constraints and represents the degree of constraint violation of individual i as described in Equation 1. c_k is the actual budget or duration needed for the region k, while q_k is the user-input budget or duration.

Regions that in a combination violate Constraint 4 are assigned the value $d_i^k = \frac{c_k}{q_k}$, where c_k is the number of user preferences the region does not fulfill, and q_k is the total number of user preferences. For all other constraints (5 and 7), the region combinations are penalized together with a constant value.

Quadratic Penalty

This approach transforms the objective function $z_i(x)$ into an unconstrained objective fitness function of the following form:

$$f_i(x) = z_i(x) - \sum_{j=1}^{N_{con}} P_j(x)$$

The penalty function P_j is a quadratic function of the penalty coefficient for a given constraint. In the implementation, the penalty coefficient was chosen as the constraint violation degree $CV_i(x)$ over all constraints. Thus, the fitness value of an individual$i$ is computed as follows:

$$f_i(x) := \begin{cases} z_i(x), & \text{if} \quad CV_i(x) \leq 0; \\ [CV_i(x)]^2, & \text{otherwise;} \end{cases}$$

Preliminary tests show better results using a quadratic increase of the violation degree as opposed to a linear increase.

4.3. Discussion

In this chapter, we presented and described in great detail NSGA-III as the baseline algorithm chosen for implementing the solution to our optimization problem. We described various heuristic solutions implemented in this thesis to modify the baseline algorithm. The objective function of the initial problem formulation was transformed in order to better handle the defined constraints. An implication of this transformation is that feasible members of the population will dominate infeasible members during non-dominated sorting. The tournament selection procedure described in Algorithm 5 was relaxed such that individuals having higher fitness values are prioritized. Comparison between individual degrees of constraint violation is thus eliminated. However, the niche preservation operation is still used to select between individuals having the same Pareto front.

5. Evaluation

The research topic of this thesis is how to best combine items (particularly regions) to improve travel recommendations. After much consideration and research, a genetic EA was selected as the best algorithmic solution to the MOOP. In this chapter, we describe the conducted experiments and their results. The experiments are presented in two parts. First, an online evaluation of the different algorithm variants is conducted. Then, we filter the results to eliminate some algorithm variants. The second part is the offline user survey. In the survey, participants are requested to rate different region combinations on a five-point Likert scale. In the following sections, the experimental setup as well as the online and offline studies are explained in depth.

5.1. Experimental setup

The experimental setup consists of combinations of elements of an approach that together form an algorithm variant. Figure 5.1 how the various algorithmic approaches described in Section 4.2 are combined to form six separate algorithm variants. Variant 1 is the baseline algorithm and is the closest to NSGA-III. Therefore, it can be described as an implementation of the unconstrained NSGA-III described in Part I of the NSGA-III paper [85].

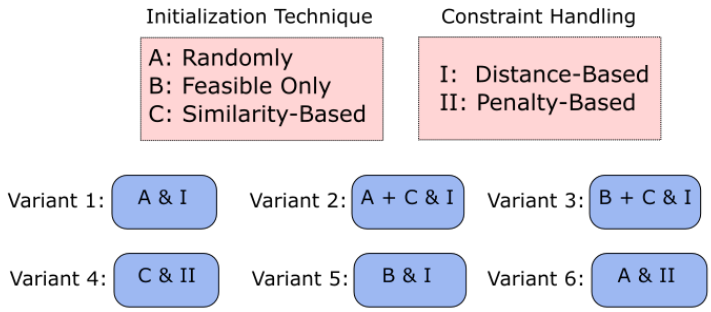


Figure 5.1.: Combination of algorithmic approaches to form algorithm variants

Table 5.1 summarizes the test setup parameters used for all variants. Each algorithm variant terminates at the 150^{th} generation. This generation number for termination was selected because after several tests, it was discovered to be the limit to further improvement to the Pareto front. Moreover, unnecessary computational time that limits the usage of the RS needed to be avoided. During the various test cases, the number of objectives M was fixed

at four. This number was chosen as it was considered sufficient to test the algorithms. In addition, increasing the number of objectives impacts the computational speed of the algorithm variants.

Each algorithm variant underwent 10 test runs for each sample input query. Three inputs (see 5.2) were carefully chosen to capture different travel preferences a user might have.

Table 5.1.: Constant test setup parameters for all algorithmic variants.

Parameter	Value
Maximum generation	150
Number of divisions on every objective axis p	12
Population Size N (Variants 1, 2, 4, and 6)	92
Population Size N (Variant 3 and 5)	40
Crossover operator	Two point
Crossover probability P_c	0.3
Mutation operator	Flip bit
Mutation probability P_m	1/length of input preferences
Number of test runs per variant	10
Number of sample queries	3

Table 5.2.: Input queries used for tests

Input	Values			
	Preferences	Budget	Duration in Weeks	Preferred Months
1	Beach, Water sports, Entertainment, Culinary, Safety from crime	3,000	3	July, August
2	Nature and Wildlife, Hiking, Culture	6,000	6	December, January, February
3	Cities and Architecture, Culture, Culinary, Shopping, Safety from crime	4,000	4	September, October

5.2. Online Evaluation

After several runs of the different algorithm variants, the goal of the online evaluation of the test results is to compare the effectiveness of each variant based on the following:

1. How many regions is it able to combine as a single suggestion,
2. The best total score from its list of suggested region combinations,
3. Computational time required, and
4. How diverse its suggestions are from other variants.

5.2.1. Algorithm Variants Comparison

To compare the different algorithm variants, we present two perspectives of the results. First, we show the generational score of each variant. Then, we show the distribution of the obtained scores for each variant alongside the computational times.

In figure 5.2, the average feasible and infeasible scores at each generation are depicted for each variant with a 95% confidence interval. The gray dotted line represents the constant average over all the feasible scores of all algorithm variants; it shows the minimum achievable score for each variant. Variants 1, 5, and 6 obtained average scores above the target line across generations. The distribution of scores for Variants 4 and 6 are very noisy. This can be explained by the heavy penalty values used by both variants. At first glance, we can infer that Variants 1, 5, and 6 are the best-performing variants. To make an informed decision, a second perspective on the results is presented.

Statistical data obtained through experiments showed that the scores generated by Variants 1, 4, and 5 outperform all other variants in terms of the average total score achieved. Figure 5.3 shows the score (per combination) and the average time taken in seconds for each evaluated algorithm variant. On the y-axis, the color intensity corresponds to the average number of generated combinations per run. A gray dotted line on each variant indicates the average amount of computation time needed by the variant.

It is observed that Variant 3 requires the greatest amount of time (249.1s), while Variant 4 requires the least amount of time (71.4s). However, Variants 2, 3, and 4 are the worst-performing algorithms, having average scores of 10.3, 12.25, and 11.20, respectively. Variant 5, with an average score of 29.54, is statistically the best-performing algorithm. Variants 5 and 6 generated high numbers of region combinations. Variant 1 also generated high scores; however, its maximum obtained score is an outlier.

5.2.2. Results

Informed by the two perspectives on the data from the experiments, we obtain a measure of the effectiveness of the variants as we discussed at the beginning of the online evaluation

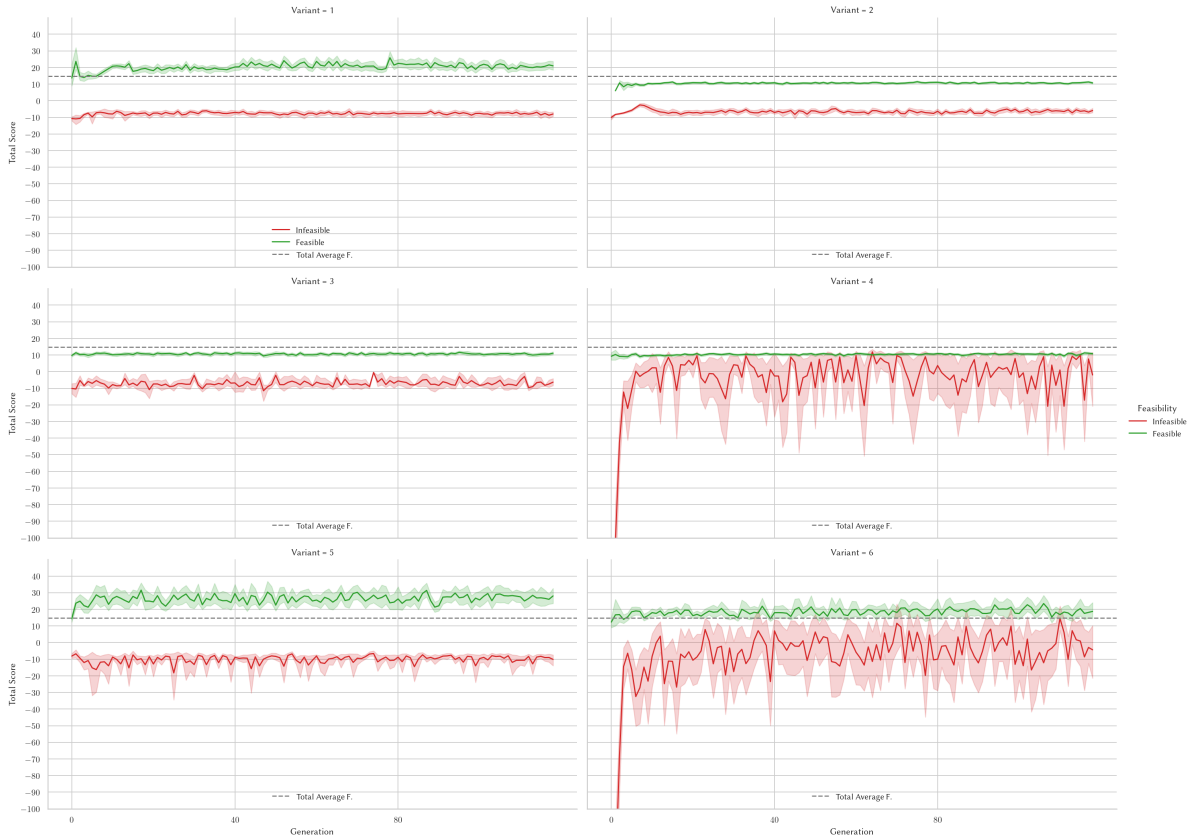


Figure 5.2.: Average total score across generations. The gray dotted line represents a constant overall average score from all feasible combinations.

section. Variants 1, 5, and 6 rank the highest on all metrics and are able to suggest diverse region combinations. However, Variants 2, 3, and 4 suggest similar region combinations on most runs.

Table 5.3.: Similarity in suggested region combination by variants

Input	Variant	Suggested Regions
1	2, 3, 4	Central Africa, Sahel East
2	2, 3, 4	Qatar and Bahrain, Iraq and Kuwait
3	2, 3, 4	Greenland

Table 5.3 shows the best region combinations suggested by the three variants. It is noteworthy that the best region combinations from the three low-ranking variants are exactly the same. A summary of the performance of the variants on each metric is presented in Table 5.4. We can conclude that Variants 1, 5, and 6 should be the selected variants for the offline evaluation.

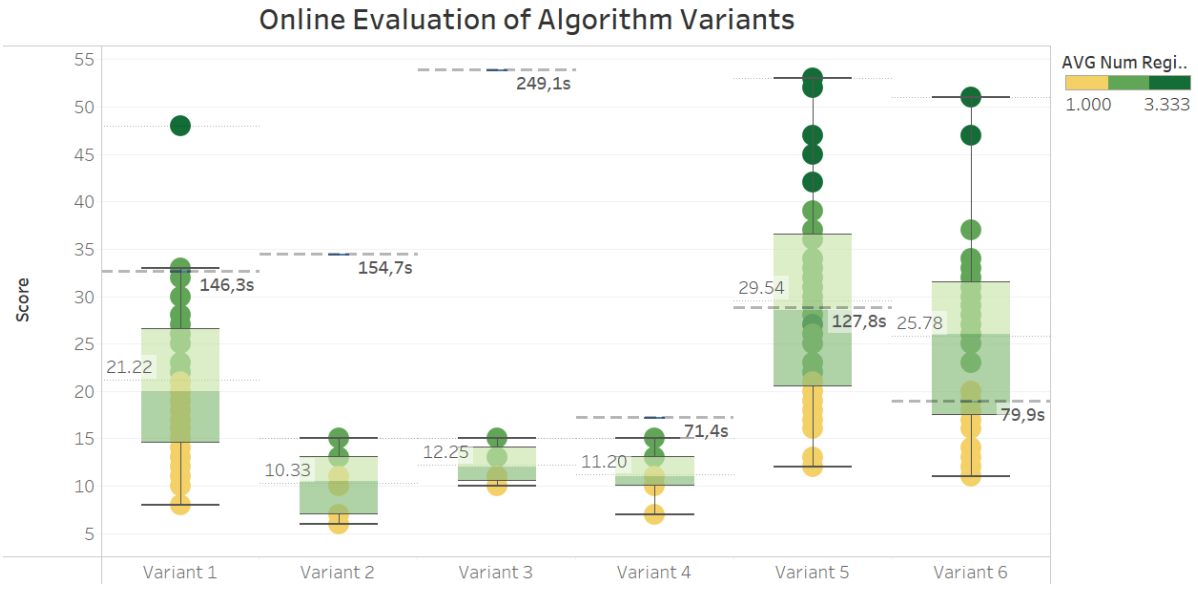


Figure 5.3.: Online Experimentation Results (For Values Table, see appendix A.1)

However, for diversity, a fourth variant was selected. The worst performing variant (Variant 2) was chosen as the extra variant for offline evaluation. Thus, results from Variants 1, 2, 5, and 6 are to be further evaluated in the user survey.

Table 5.4.: Variants comparison of effectiveness Metrics

	Computational time	# Regions	Achievable Score	Non-Similarity
Variant 1		✓	✓	✓
Variant 2				
Variant 3				
Variant 4	✓			
Variant 5	✓	✓	✓	✓
Variant 6	✓	✓	✓	✓

5.3. Offline Evaluation

The offline evaluation was designed with the aim of evaluating the performance of the 4 designed variants of the genetic evolutionary algorithm. We conducted a study comprising participants of varying travelling knowledge. Respondents were giving the opportunity to evaluate for their preferred travel styles.

5.3.1. Procedure of the Survey

The user survey was composed of suggested region combinations and minimum stay durations obtained from algorithm Variants 1 (baseline), 4, 5, and 6. Region combinations having the highest score from each variant were used for the three sample input queries (scenarios shown in Table 5.2) presented to the respondents. A typical region combination was shown to the participants as presented in Figure 5.4.

Region	Minimum Duration (weeks)
USA South	1
USA Hawaii	1
USA Alaska	1

Figure 5.4.: Suggested region combinations and stay durations from Variant 5 as presented to participants (all presented combinations can be seen in appendix figures A.1, A.2, & A.3)

First, the participants were asked general background information about their travel knowledge and styles. Specifically, they questioned regarding the following:

1. How often they travel;
2. How good they are at planning travels,
3. Their preferred travel activities from a list of activities, and
4. Their level of trust.

This allowed us to match the travelers to the travel scenarios and also to categorize them as advanced, basic, or average travelers.

In total, 12 suggested region combinations were presented to the participants, and they were asked to rate each suggestion on a five-point Likert scale ranging from 1 to 5 (strongly disagree to strongly agree). Point 0 was reserved for participants who did not relate to the travel scenario. The participants rated each recommendation based on the following:

1. Accuracy of the suggestion,
2. Overall satisfaction with the suggestion,
3. How well the regions fit together (i.e, cohesiveness), and
4. Diversity of the recommendations.

Table A.2 in the appendix describes the specific wording of the survey questions and the response types for each category.

5.3.2. Results

A total of 104 participants with varying travel knowledge took part in the survey. Each participant could relate to at least two of the input scenarios (i.e., minimum of 204 responses per variant). Hence, we evaluated each algorithm variant on at least two travel types. Figure 5.5 compares the algorithm variants based on the responses obtained on a diverging bar chart.

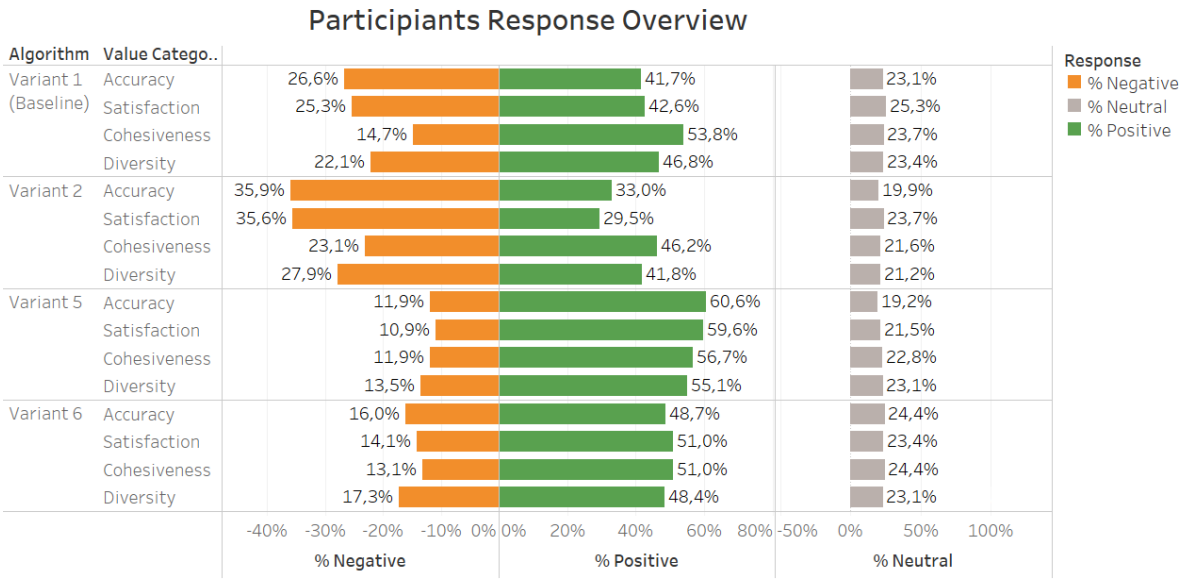


Figure 5.5.: Comparison of the algorithm variants based on responses obtained per variant on each question category. Please note that negative signs on the axis only represent the direction of the diverging chart.

Each row represents the results for an algorithm variant, and this is further grouped based on the question categories. On the x-axis, the responses are grouped into percentage negative, percentage positive, and percentage neutral. For each question category, the percentage negative represents the percentage of the total responses regarding a particular variant with ratings less than 3 ($(0, 2]$). Percentage positive represents ratings from 4 to 5 ($[4, 5]$) on a question category. Participants who rated a suggestion with the value of 3 were considered neutral to ensure that we could accurately rate responses as positive or negative.

From the diverging chart, it can be observed that Variant 5 was the best-performing variant relative to baseline. More than half of the respondents rated it highly in all question categories. It ranked highest in accuracy, satisfaction, cohesiveness, and diversity. An improvement in accuracy, 39.8% improvement in satisfaction, 5.5% in cohesiveness, and 17.8% in diversity was achieved relative to the baseline for this variant. Compared to other variants in each category, no variant performed better than Variant 5.

Variant 2 performed worse than baseline on all metrics. It ranked last in overall satisfaction,

followed by accuracy, diversity, and cohesiveness. It showed a 20.8% decrease in accuracy, a 30.8% decrease in satisfaction, a 14.3% decrease in cohesiveness, and a 10.6% decrease in diversity relative to the baseline.

Variant 6 was the second-best performing variant; 48.7% of the respondents found it accurate, and 51% were satisfied with its results. However, compared to baseline, it performed worse in terms of cohesiveness of region combinations.

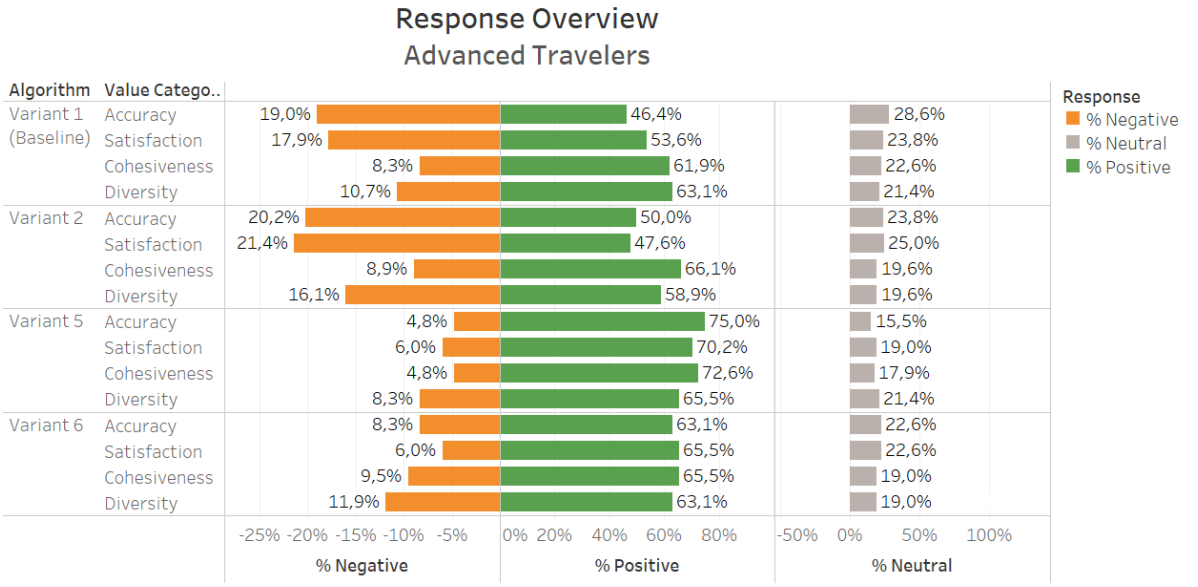


Figure 5.6.: Response overview filtered by travel knowledge of participants – Advanced Knowledge Travelers Only

A Kruskal-Wallis [86] test was computed to determine whether there is a statistically significant difference between the median ratings of the algorithm variants concerning the responses gotten from the survey. For better test accuracy, we did not take neutral responses into account. The test showed a significant difference ($p \leq 0.001$) between the algorithm variants with regard to the ratings by the respondents. As a consequence of the statistically significant results from the Kruskal-Wallis test, a Dunn's test was conducted to determine the significance level between the variant pairs. Table 5.5 reveals the results of the test. There is a significant difference between all algorithm variants. The pairs of Variant 1 and 6 and the pairs of Variant 5 and 6 are less different from each other. However, using a significance level of 0.05, all variant pairs are statistically significantly different from one another,

We created travel knowledge categories for the participants according to their travel planning ability and travel experience. Participants with travel experience and planning abilities above three were categorized as advanced travelers. Participants who rated their travel experience and planning ability below two were classified as basic travelers. All other travel experiences and travel planning ability rating combinations were categorized as having average travel knowledge.

Table 5.5.: Dunn-Bonferroni-Tests for pairwise comparison of the level of significance between the algorithm variants. It tells us which variants are statistically significantly different. Adj. p: Values adjusted with Bonferroni correction

	Test Statistic	**Std. Error**	**Std. Test Statistic**	**p**	**Adj. p**
Variant 1 - Variant 2	292.41	44.88	6.52	<.001	<.001
Variant 1 - Variant 5	-248.62	42.47	-5.85	<.001	<.001
Variant 1 - Variant 6	-138.4	43.29	-3.2	.001	.006
Variant 2 - Variant 5	-541.03	44.65	-12.12	<.001	<.001
Variant 2 - Variant 6	-430.81	45.43	-9.48	<.001	<.001
Variant 5 - Variant 6	110.22	43.05	2.56	.01	.042

An interesting observation made when filtering the responses for advanced travelers only (5.6) was that compared to baseline, travelers with more than average travel knowledge were more satisfied with the cohesiveness and accuracy of the region suggestions from Variant 2. Furthermore, based on the positive responses of advanced travelers, Variant 6 performed at least as good as the baseline in terms of diversity and better than the baseline in terms of accuracy, satisfaction, and cohesiveness.

This insight from the responses of advanced travelers is crucial because they tend to have more travel domain knowledge. A second Kruskal-Wallis test was conducted to test if this observed difference in the responses by travelers with advanced knowledge and other traveler is statistically significant. The test revealed that there is a significant difference ($p \leq 0.001$) between the advanced, average, and basic travelers with respect to their response values. Figure 5.7 illustrates a side-by-side comparison of the ratings of all algorithm variants by advanced, average, and basic travelers. It is evident that advanced travelers are more generous with their evaluations than all other travel knowledge categories.

5.4. Discussion

The results obtained from the offline evaluation are in line with the results obtained through the online evaluation of the variants. As demonstrated above, Variants 5 and 6 were the best-performing algorithm variants, while Variant 2 was the worst-performing variant. Therefore, the overall performance of the algorithm variants is directly dependent on the performance of the variants in the online evaluation.

Variant 6 was unable to improve baseline in terms of cohesiveness. However, advanced travelers rated Variant 6's suggestions as more cohesive than baseline. From the results of the second Kruskal-Wallis test, there is a statistically significant difference between advanced travelers and all other travel knowledge categories. For participants to provide more precise evaluations of the cohesiveness of the regions, they must have good travel domain knowledge. Advanced travelers tend to have a better understanding of how the regions fit together. Thus,

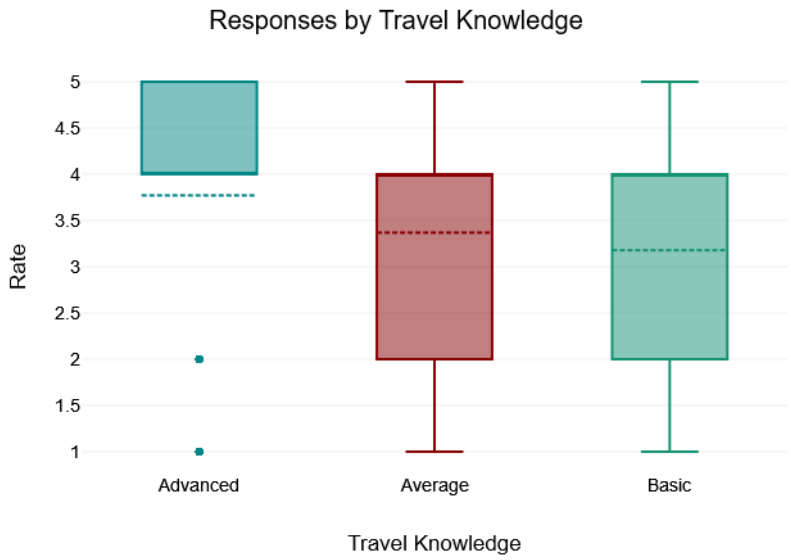


Figure 5.7.: Side-by-Side comparison of ratings from advanced, average, and basic travelers

we cannot totally rule out Variant 6 because of its lower score in the category of cohesiveness. Moreover, Variant 6 took the least computational time of all the algorithm variants (ϕ 79.9s), and a fast computational time is vital to ensure the usefulness of a RS. Hence, we can conclude that overall, Variant 5, Variant 6, and Variant 1 rank the highest.

5.5. Implications

The results from this evaluation reveal the high potential of initializing the start population with feasible individuals only. A large portion of the computational time used by the feasibility-based method is spent searching for feasible individuals. The tests performed in this work used a random search method. A hypothesis is that an efficient search heuristic would significantly improve the time performance of the feasibility-based initialization method. In Section 3.3 various search heuristics such as the iterative local search, TS, and others were explained. The computational time for the evolutionary process with the feasibility-based approach is shorter because a smaller population size is used. Therefore, improving the start population search process will significantly improve the computational speed of the algorithm.

Additionally, penalty-based constraint handling has proven effective, as shown in the evaluation of algorithm Variant 6. A new variant that uses a combination of a feasibility-based initialization and penalty-based constraint-handling of future offspring could provide effective results.

The computational speed of the other algorithm variants can still be improved through further

tests. A high population number and a high maximum generation have a negative influence on the computational time of the evolutionary algorithm in general. Additional experiments must be conducted to test the trade-off limit between population number and the maximum generation that provide acceptable item combinations.

6. Conclusion

In this thesis, we formally introduced the problem of item combination in destination RSs as a MOOP. Empirical research into how current state-of-the-art algorithms used for solving OPs and general MOOPs can be extended to efficiently combine items in destination RSs was performed. A comparison of choice algorithms was made, and it was decided that an evolutionary algorithm is best-suited for our problem class. The NSGA-III, a dominance-based genetic algorithm, was chosen to obtain a sequence of candidate solutions that satisfy user constraints.

Six heuristic variations of the algorithm were implemented. Evaluations conducted using an online study, and through a user survey showed that initializing the algorithm with feasible population members only and penalizing the individuals based on their distance to a feasible area had the highest rating of all variants on all defined metrics. Pure randomness in the initialization of population members showed poor results. However, incorporating composite penalization of the individuals generated by random initialization improved performance significantly.

Similarity-based initialization, as used in [6] and further investigated in this work, produced poor results. This initialization technique restricts the search space such that there are a limited number of regions to combine during the actual evolutionary process. Combining items for recommendation is heavily dependent on the defined constraints. Similarity-based initialization does not take the constraints into account. Hence, during the evolutionary step in which the actual constraints determine the fitness of an individual, the limited regions in the population perform poorly. This also explains the poor performance in terms of diversity shown by the travel package algorithm developed by Wörndl and Herzog; the algorithm also used an equivalent similarity metric.

In general, this thesis has shown the potential applicability of EAs in efficiently combining items in destination RSs.

6.1. Limitations

The results obtained on metrics such as accuracy, satisfaction, cohesiveness, and diversity by this work are above average but imply possible improvement points through further research. Higher ratings from users in terms of accuracy and satisfaction are needed. The computational time of the algorithm is crucial for the usability of a RS. Evaluations have

shown that a considerable amount of time is needed to complete the evolutionary steps. Therefore, the current implementations of this thesis are not suitable for synchronous RS. However, the algorithm could be implemented into an asynchronous RS. The underlying data for this research influenced the limitations on recommended travel budgets for a region. Some regions in the database had no data regarding the recommended budget. Consequently, we excluded the regions' budget recommendations from suggested region combinations.

6.2. Future Research

Research, implementation, and evaluations from this thesis can be considered as a baseline for future work on item combination for destination RSs. We investigated the applicability of genetic EAs to this topic, and we introduced possibly applicable algorithms, for example ACO. A possible direction for future work could be to investigate this algorithm type and compare its results with the results obtained in this work.

Further research into EAs for item combination in RSs has to be done. With this work as a baseline, investigating the trade-offs between the number of objectives, maximum generation, and population size could be a possible research focus. Finding the right balance between the three parameters is vital to improving computational speed while ensuring optimal item combinations.

Additionally, results from user survey proved that travelers with advanced knowledge are significantly different from travelers with lesser knowledge about destinations. This suggests a need for further user surveys with travel experts. Wörndl and Herzog had initially conducted their survey with travel experts only. Their survey method is indeed a better approach for evaluating results from an implemented algorithm.

An improved database of regions with the needed budgets and updated scores for different activities is vital and constitutes possible future work. A transformation and enhancement of the region names with ISO 3166 transformation would significantly increase the real-world usage of the database. For example, it would be possible to obtain the latitude and longitude coordinates of the regions if their ISO codes were added to the database. This would improve accuracy in routing and the cohesiveness of recommended regions.

A. Appendix

Table A.1.: Results from statistical online evaluation Of variants

Variant	Score							Avg Time (s)
	mean	std	min	25%	50%	75%	max	
1	18.76	6.97	8.0	14.25	17.5	21.0	48.0	146.3
2	11.05	2.57	6.0	10.0	11.0	13.0	15.0	154.75
3	12.54	1.9	10.0	11.0	13.0	15.0	15.0	249.12
4	11.53	1.9	7.0	10.0	11.0	13.0	15.0	71.42
5	26.82	8.48	12.0	20.0	27.0	33.0	53.0	127.77
6	22.03	9.1	11.0	16.0	19.0	29.0	51.0	79.91

Table A.2.: Category and wording of survey questions

Question category	Wording	Response Type
Travel Knowledge	How often do you travel for vacation?	5 points Likert Scale
	How good are you at planning vacations?	
Scenario Match	What are your preferred travel experiences?	Multi Choice from Beach, Watersports, Entertainment, Shopping Culinary, Hiking, Nature & Wildlife, Cities & Architecture
Trust	Do you tend to trust a person/thing, even though you have little knowledge of it	5 points Likert scale
Accuracy	The recommended regions matched the input preferences	
Satisfaction	This recommender system gave me good suggestions for this scenario	
Diversity	The regions recommended are diverse	
Cohesiveness	The recommended regions fit well together	

Table A.3.: Dunn-Bonferroni-Tests for pairwise comparison between the travel knowledge of respondents.

	Test Statistic	Std. Error	Std. Test Statistic	p	Adj. p
Advanced - Average	270.87	32.56	8.32	<.001	<.001
Advanced - Basic	373.27	70.16	5.32	<.001	<.001
Average - Basic	102.41	71.72	1.43	.153	.46

Region	Minimum Duration (weeks)
Eastern Europe	1
North Europe	1

(a) Variant 1 (baseline)

Region	Minimum Duration (weeks)
Central Africa	2
Sahel East	1

(b) Variant 4

Region	Minimum Duration (weeks)
USA South	1
USA Hawaii	1
USA Alaska	1

(c) Variant 5

Region	Minimum Duration (weeks)
USA Hawaii	1
USA Texas	1

(d) Variant 6

Figure A.1.: Recommendations generated for input scenario 1 per variant

Region	Minimum Duration (weeks)
Nepal	1
Maldives	1
India	1

(a) Variant 1 (baseline)

Region	Minimum Duration (weeks)
Qatar and Bahrain	1
Iraq and Kuwait	2

(b) Variant 4

Region	Minimum Duration (weeks)
Egypt	1
Sahel West	1
Southern Africa	2

(c) Variant 5

Region	Minimum Duration (weeks)
Ecuador	1
Peru	2

(d) Variant 6

Figure A.2.: Recommendations generated for input scenario 2 per variant

Region	Minimum Duration (weeks)
Eastern Europe	1
Central Europe	1
British Islands	1

(a) Variant 1 (baseline)

Region	Minimum Duration (weeks)
Greenland	1

(b) Variant 4

Region	Minimum Duration (weeks)
Taiwan	1
South Korea	1

(c) Variant 5

Region	Minimum Duration (weeks)
Taiwan	1
Mongolia	2

(d) Variant 6

Figure A.3.: Recommendations generated for input scenario 3 per variant

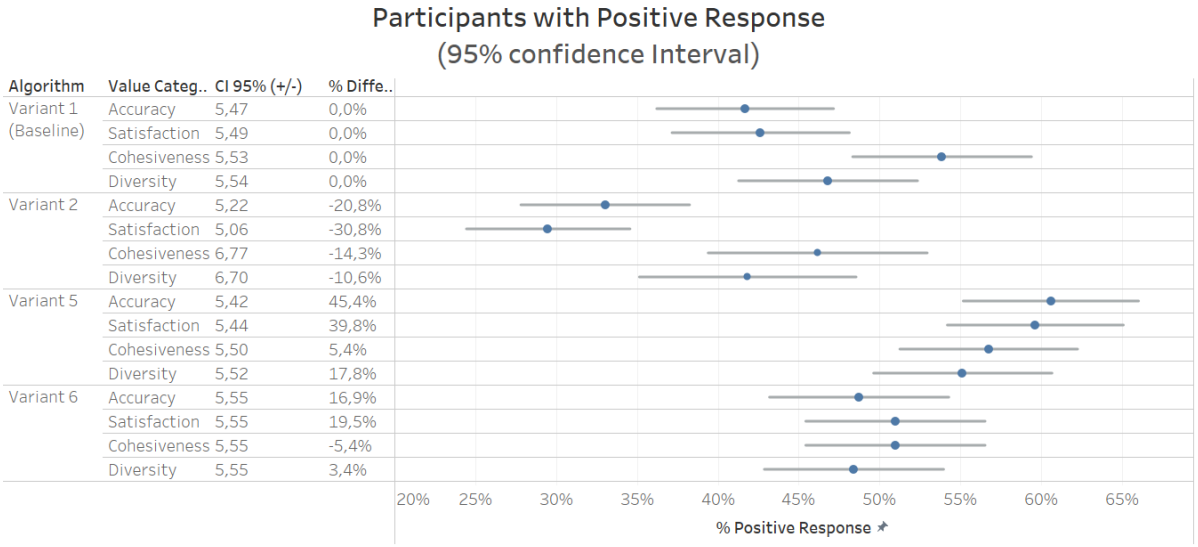


Figure A.4.: Margin of error in the results of the survey using a 95% confidence interval on the positive responses; The percentage improvement of each record on the different categories compared to baseline is shown. The percentage improvement is calculated as the percentage difference for each category relative to the baseline. From the results obtained, we can ascertain with 95% confidence that each variant will always generate the same results, with an approximately [-6, 6] margin of error.

List of Figures

List of Tables

Glossary

candidate solution are potential solutions that may possibly be encountered during an attempt to solve the given problem instance; but unlike solutions, they may not satisfy all the conditions from the problem definition. 16, 57

DEAP DEAP - Distributed Evolutionary Algorithms in Python - is a novel evolutionary computation framework for rapid prototyping and testing of ideas. It seeks to make algorithms explicit and data structures transparent. 34, 35, 38, 57

GIL The python global intepreter lock is a mutex (or a lock) that allows only one thread to hold the control of the Python interpreter plural. 57

Tensor can be seen as a generalization of matrices and are often used inter-changeably with matrices in literature. 6, 57

TUM A university in Germany, Bavaria, in the city of Munich. 57

Acronyms

ACO ant colony optimization. 7, 8, 18, 21, 22, 24, 26, 52, 57

AI artifical intelligence. 4, 57

BCO bee colony optimzation. 22, 26, 57

CPS cyber-physical systems. 4, 57

CV constraint violation. 30, 31, 57

EA evolutionary algorithm. 8, 18, 21, 24, 26, 27, 34, 40, 51, 52, 57

EMO evolutionary multi-objective. 27, 57

GA genetic algorithm. 57

HOSVD higher-order singular value decomposition. 6, 57

ILP integer linear program. 57

IOT internet-of-things. 4, 57

LBSN location-based social network. 5–7, 57

LDA latent dirichlet allocation. 6, 57

LP linear program. 57

MCKP multiple-choice knapsack problem. 57

MOOP multi-objective orienteering problem. 2, 3, 7, 8, 12, 13, 15, 22–25, 40, 51, 57

MTG mobile tourist guide. 4, 57

NSGA non-dominated sorting genetic algorithm. 21, 25–32, 34, 35, 39, 40, 51, 56, 57

OP orienteering problem. 1–3, 7–10, 12, 14, 15, 20, 26, 34, 51, 57

OPSP OP with stochastic profits. 7, 57

OPTW orienteering problem with time windows. 7, 57

OR operations research. 4, 57

PMA pareto memetic algorithm. 21, 57

POI Point of Interest. 1, 2, 6, 7, 10, 13, 27, 28, 57

PSO particle swarm optimization. 7, 57

RS recommender system. 1–6, 10, 12, 40, 49, 51, 52, 57

SA simulated annealing. 18, 19, 21, 57

SLS stochastic local search. 18, 57

SPEA strength pareto evolutionary algorithm. 21, 26, 57

TDOP time dependent orienteering problem. 7, 57

ThOP thief orienteering problem. 7, 57

TOP team orienteering problem. 7, 8, 57

TOPTW team orienteering problem with time windows. 7, 8, 57

TS tabu search. 18, 20, 21, 49, 57

TSP traveling salesman problem. 1, 2, 57

TTDP tourist trip design problem. 1–4, 7, 57

VND variable neighborhood descent. 18, 21, 57

Bibliography

[1] F. Ricci, L. Rokach, and B. Shapira, "Introduction to Recommender Systems Handbook", in *Recommender Systems Handbook*, Springer US, 2011, pp. 1–35. DOI: 10.1007/978-0-387-85820-3_1.

[2] R. Burke, "Hybrid Recommender Systems : Survey and and Experiments. User Modelling and User-Adapted Interaction", *User Modeling and UserAdapted Interaction*, vol. 12, 2002, ISSN: 09241868. DOI: 10.1023/A:1021240730564.

[3] G. Linden, B. Smith, and J. York, "Amazon.com recommendations: Item-to-item collaborative filtering", *IEEE Internet Computing*, vol. 7, no. 1, 2003, ISSN: 10897801. DOI: 10.1109/MIC.2003.1167344.

[4] X. Amatriain, "Big & personal: Data and models behind Netflix recommendations", in *Proc. of 2nd Int. Workshop on Big Data, Streams and Heterogeneous Source Mining: Algorithms, Systems, Programming Models and Applications, BigMine 2013 - Held in Conj. with SIGKDD 2013 Conf.*, 2013. DOI: 10.1145/2501221.2501222.

[5] W. Wörndl, "A web-based application for recommending travel regions", in *UMAP 2017 - Adjunct Publication of the 25th Conference on User Modeling, Adaptation and Personalization*, New York, NY, United States: Association for Computing Machinery, Inc, Jul. 2017, pp. 105–106, ISBN: 9781450350679. DOI: 10.1145/3099023.3099031.

[6] D. Herzog and W. Wörndl, "A Travel Recommender System for Combining Multiple Travel Regions to a Composite Trip", *CEUR Workshop Proceedings*, vol. 1245, pp. 42–47, Jan. 2014.

[7] P. Thiengburanathum, "An intelligent destination recommendation system for tourists", *PQDT - UK & Ireland*, no. March, 2018.

[8] Y. M. Arif, H. Nurhayati, F. Kurniawan, S. M. S. Nugroho, and M. Hariadi, "Blockchain-Based Data Sharing for Decentralized Tourism Destinations Recommendation System", *International Journal of Intelligent Engineering and Systems*, vol. 13, no. 6, 2020, ISSN: 21853118. DOI: 10.22266/ijies2020.1231.42.

[9] H. Alrasheed, A. Alzeer, A. Alhowimel, N. Shameri, and A. Althyabi, "A Multi-Level Tourism Destination Recommender System", in *Procedia Computer Science*, vol. 170, Elsevier B.V., 2020, pp. 333–340. DOI: 10.1016/j.procs.2020.03.047.

[10] P. Vansteenwegen and D. V. Oudheusden, "The Mobile Tourist Guide: An OR Opportunity", *OR Insight*, vol. 20, no. 3, pp. 21–27, 2007. DOI: 10.1057/ori.2007.17.

[11] D. Gavalas, C. Konstantopoulos, K. Mastakas, and G. Pantziou, "A survey on algorithmic approaches for solving tourist trip design problems", *Journal of Heuristics*, vol. 20, no. 3, pp. 291–328, 2014, ISSN: 15729397. DOI: 10.1007/s10732-014-9242-5.

[12] T. T., "Heuristic Methods Applied to Orienteering", *Journal of Operational Research Society*, vol. 35, p. 797, 1984.

[13] A. Gunawan, H. C. Lau, and P. Vansteenwegen, *Orienteering Problem: A survey of recent variants, solution approaches and applications*, Dec. 2016. DOI: 10.1016/j.ejor.2016.04.059.

[14] M. Balabanović and Y. Shoham, "Content-Based, Collaborative Recommendation", *Communications of the ACM*, vol. 40, no. 3, 1997, ISSN: 00010782. DOI: 10.1145/245108.245124.

[15] C. Chu, A. L. Hsu, K. H. Chou, P. Bandettini, and C. P. Lin, "Does feature selection improve classification accuracy? Impact of sample size and feature selection on classification using anatomical magnetic resonance images", *NeuroImage*, vol. 60, no. 1, 2012, ISSN: 10538119. DOI: 10.1016/j.neuroimage.2011.11.066.

[16] S. Schiaffino and A. Amandi, *Polite personal agents*, 2006. DOI: 10.1109/MIS.2006.15.

[17] H. Gao, J. Tang, X. Hu, and H. Liu, "Content-aware point of interest recommendation on location-based social networks", in *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*, AAAI Press, Jan. 2015, pp. 1721–1727. DOI: 10.5555/2886521.2886559.

[18] D. Lian, Y. Ge, F. Zhang, *et al.*, "Content-Aware Collaborative Filtering for Location Recommendation Based on Human Mobility Data", in *2015 IEEE International Conference on Data Mining*, in: IEEE, Nov. 2015, pp. 261–270, ISBN: 978-1-4673-9504-5. DOI: 10.1109/ICDM.2015.69.

[19] P. Lops, M. de Gemmis, and G. Semeraro, "Content-based Recommender Systems: State of the Art and Trends", in *Recommender Systems Handbook*, Springer US, 2011, pp. 73–105. DOI: 10.1007/978-0-387-85820-3_3.

[20] R. Burke, "Knowledge-based recommender systems", *Encyclopedia of library and information systems*, vol. 69, p. 180, May 2000.

[21] I. Y. Choi, Y. U. Ryu, and J. K. Kim, "A recommender system based on personal constraints for smart tourism city", *Asia Pacific Journal of Tourism Research*, vol. 26, no. 4, 2021, ISSN: 17416507. DOI: 10.1080/10941665.2019.1592765.

[22] A. Montejo-Ráez, J. M. Perea-Ortega, M. Á. García-Cumbreras, and F. Martínez-Santiago, "Otiûm: A web based planner for tourism and leisure", *Expert Systems with Applications*, vol. 38, no. 8, 2011, ISSN: 09574174. DOI: 10.1016/j.eswa.2011.02.005.

[23] G. Adomavicius and A. Tuzhilin, *Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions*, 2005. DOI: 10.1109/TKDE.2005.99.

[24] M. A. Ghazanfar and A. Prügel-Bennett, "Building switching hybrid recommender system using machine learning classifiers and collaborative filtering", *IAENG International Journal of Computer Science*, vol. 37, no. 3, 2010, ISSN: 1819656X.

[25] Z. Cheng, J. Caverlee, K. Lee, and D. Z. Sui, "Exploring Millions of Footprints in Location Sharing Services", *Icwsm*, vol. 2010, no. Cholera, 2011. DOI: papers3://publication/uuid/0C46BD5D-4908-4A8A-BD06-5BCB2F1DE282.

[26] G. Liao, S. Jiang, Z. Zhou, C. Wan, and X. Liu, "POI recommendation of location-based social networks using tensor factorization", in *Proceedings - IEEE International Conference on Mobile Data Management*, vol. 2018-June, 2018. DOI: 10.1109/MDM.2018.00028.

[27] L. Huang, Y. Ma, Y. Liu, and A. K. Sangaiah, "Multi-modal Bayesian embedding for point-of-interest recommendation on location-based cyber-physical–social networks", *Future Generation Computer Systems*, vol. 108, 2020, ISSN: 0167739X. DOI: 10.1016/j.future.2017.12.020.

[28] C. Cheng, H. Yang, M. R. Lyu, and I. King, "Where you like to go next: Successive point-of-interest recommendation", in *IJCAI International Joint Conference on Artificial Intelligence*, 2013.

[29] C. Cheng, H. Yang, I. King, and M. R. Lyu, "Fused matrix factorization with geographical and social influence in location-based social networks", in *Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence*, AAAI Press, 2012, pp. 17–23.

[30] C. Chen, Z. Liu, P. Zhao, J. Zhou, and X. Li, "Privacy preserving point-of-interest recommendation using decentralized matrix factorization", in *32nd AAAI Conference on Artificial Intelligence, AAAI 2018*, 2018.

[31] H. Gao, J. Tang, X. Hu, and H. Liu, "Exploring temporal effects for location recommendation on location-based social networks", in *Proceedings of the 7th ACM conference on Recommender systems*, New York, NY, USA: ACM, Oct. 2013, pp. 93–100, ISBN: 9781450324090. DOI: 10.1145/2507157.2507182.

[32] Y. Koren, R. Bell, and C. Volinsky, "Matrix factorization techniques for recommender systems", *Computer*, vol. 42, no. 8, 2009, ISSN: 00189162. DOI: 10.1109/MC.2009.263.

[33] J. Vandewalle and D. Callaerts, "Singular Value Decomposition: a powerful concept and tool in signal processing", 1990.

[34] S. Zhao, T. Zhao, H. Yang, M. R. Lyu, and I. King, "STELLAR: Spatial-temporal latent ranking for successive point-of-interest recommendation", in *30th AAAI Conference on Artificial Intelligence, AAAI 2016*, 2016.

[35] D. Yang, D. Zhang, Z. Yu, and Z. Wang, "A sentiment-enhanced personalized location recommendation system", in *Proceedings of the 24th ACM Conference on Hypertext and Social Media - HT '13*, New York, New York, USA: ACM Press, 2013, pp. 119–128, ISBN: 9781450319676. DOI: 10.1145/2481492.2481505.

[36] Q. Liu, S. Wu, L. Wang, and T. Tan, "Predicting the next location: A recurrent model with spatial and temporal contexts", *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, vol. 30, no. 1, pp. 194–200, Feb. 2016. DOI: 10.5555/3015812.3015841.

[37] P. Vansteenwegen, W. Souffriau, and D. V. Oudheusden, *The orienteering problem: A survey*, 2011. DOI: 10.1016/j.ejor.2010.03.045.

[38] B. L. Golden, L. Levy, and R. Vohra, "The orienteering problem", *Naval Research Logistics (NRL)*, vol. 34, no. 3, pp. 307–318, 1987.

[39] R. Ramesh and K. M. Brown, "An efficient four-phase heuristic for the generalized orienteering problem", *Computers & Operations Research*, vol. 18, no. 2, pp. 151–165, 1991.

[40] W. Souffriau, P. Vansteenwegen, J. Vertommen, G. V. Berghe, and D. V. Oudheusden, "A personalized tourist trip design algorithm for mobile tourist guides", *Applied Artificial Intelligence*, vol. 22, no. 10, pp. 964–985, 2008.

[41] L. Ke, C. Archetti, and Z. Feng, "Ants can solve the team orienteering problem", *Computers and Industrial Engineering*, vol. 54, no. 3, 2008, ISSN: 03608352. DOI: 10.1016/j.cie.2007.10.001.

[42] M. Wagner, "Stealing items more efficiently with ants: A swarm intelligence approach to the travelling thief problem", in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 9882 LNCS, 2016. DOI: 10.1007/978-3-319-44427-7_25.

[43] K. D. Mukhina, A. A. Visheratin, and D. Nasonov, "Orienteering Problem with Functional Profits for multi-source dynamic path construction", *PLoS ONE*, vol. 14, no. 4, Apr. 2019, ISSN: 19326203. DOI: 10.1371/journal.pone.0213777.

[44] R. Martín-Moreno and M. A. Vega-Rodríguez, "Multi-Objective Artificial Bee Colony algorithm applied to the bi-objective orienteering problem", *Knowledge-Based Systems*, vol. 154, 2018, ISSN: 09507051. DOI: 10.1016/j.knosys.2018.05.005.

[45] A. Z. Şevkli and F. E. Sevilgen, "StPSO: Strengthened particle swarm optimization", *Turkish Journal of Electrical Engineering and Computer Sciences*, vol. 18, no. 6, 2010, ISSN: 13000632. DOI: 10.3906/elk-0909-18.

[46] M. Schilde, K. F. Doerner, R. F. Hartl, and G. Kiechle, "Metaheuristics for the bi-objective orienteering problem", *Swarm Intelligence*, vol. 3, no. 3, 2009, ISSN: 19353812. DOI: 10.1007/s11721-009-0029-5.

[47] Y. Lu, U. Benlic, and Q. Wu, "A memetic algorithm for the Orienteering Problem with Mandatory Visits and Exclusionary Constraints", *European Journal of Operational Research*, vol. 268, no. 1, 2018, ISSN: 03772217. DOI: 10.1016/j.ejor.2018.01.019.

[48] H. Bouly, D. C. Dang, and A. Moukrim, "A memetic algorithm for the team orienteering problem", *4OR*, vol. 8, no. 1, 2010, ISSN: 16142411. DOI: 10.1007/s10288-008-0094-4.

[49] A. Divsalar, P. Vansteenwegen, K. Sörensen, and D. Cattrysse, "A memetic algorithm for the orienteering problem with hotel selection", *European Journal of Operational Research*, vol. 237, no. 1, 2014, ISSN: 03772217. DOI: 10.1016/j.ejor.2014.01.001.

[50] J. Wu, M. Wagner, S. Polyakovskiy, and F. Neumann, "Evolutionary computation plus dynamic programming for the Bi-objective travelling thief problem", in *GECCO 2018 - Proceedings of the 2018 Genetic and Evolutionary Computation Conference*, 2018. DOI: 10.1145/3205455.3205488.

[51] L. M. Faeda and A. G. Santos, "A Genetic Algorithm for the Thief Orienteering Problem", in *2020 IEEE Congress on Evolutionary Computation, CEC 2020 - Conference Proceedings*, 2020. DOI: 10.1109/CEC48606.2020.9185848.

[52] G. Kobeaga, M. Merino, and J. A. Lozano, "An efficient evolutionary algorithm for the orienteering problem", *Computers and Operations Research*, vol. 90, 2018, ISSN: 03050548. DOI: 10.1016/j.cor.2017.09.003.

[53] Q. Pan and X. Wang, "Independent travel recommendation algorithm based on analytical hierarchy process and simulated annealing for professional tourist", *Applied Intelligence*, vol. 48, no. 6, 2018, ISSN: 15737497. DOI: 10.1007/s10489-017-1014-0.

[54] N. Labadie, J. Melechovský, and R. Wolfler Calvo, "Hybridized evolutionary local search algorithm for the team orienteering problem with time windows", *Journal of Heuristics*, vol. 17, no. 6, 2011, ISSN: 15729397. DOI: 10.1007/s10732-010-9153-z.

[55] D. C. Dang, R. El-Hajj, and A. Moukrim, "A branch-and-cut algorithm for solving the Team Orienteering Problem", in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 7874 LNCS, 2013. DOI: 10.1007/978-3-642-38171-3_23.

[56] M.-H. E., "A TABU search heuristic for the team orienteering problem", *Computers and Operations Research*, vol. 32, p. 1379, 2005.

[57] J. J. Burg, J. Ainsworth, B. Casto, and S.-D. Lang, "Experiments with the "Oregon Trail Knapsack Problem"", *Electronic Notes in Discrete Mathematics*, vol. 1, pp. 26–35, Mar. 1999, ISSN: 15710653. DOI: 10.1016/S1571-0653(04)00003-4.

[58] C. Fonseca and J. Knowles, "A tutorial on the performance assessment of stochastic multiobjective optimizers", *TIK-Report*, vol. 214, 2005. DOI: 10.3929/ethz-b-000023822.

[59] T. F. Gonzalez, *Handbook of approximation algorithms and metaheuristics*. 2007. DOI: 10.1201/9781420010749.

[60] H. Holger H and S. Thomas, *Stochastic Local Search*, 9. Elsevier, 2005, vol. 53, ISBN: 9781558608726. DOI: 10.1016/B978-1-55860-872-6.X5016-1.

[61] L. Paquete, M. Chiarandini, and T. Stützle, "Pareto Local Optimum Sets in the Biobjective Traveling Salesman Problem: An Experimental Study", in *Metaheuristics for Multiobjective Optimisation*, 535th ed., Berlin, Heidelberg: Springer, 2004, pp. 177–199. DOI: 10.1007/978-3-642-17144-4_7.

[62] E. Angel, E. Bampis, and L. Gourvès, "Approximating the Pareto curve with local search for the bicriteria TSP(1,2) problem", *Theoretical Computer Science*, vol. 310, no. 1-3, 2004, ISSN: 03043975. DOI: 10.1016/S0304-3975(03)00376-1.

[63] A. Duarte, J. J. Pantrigo, E. G. Pardo, and N. Mladenovic, "Multi-objective variable neighborhood search: an application to combinatorial optimization problems", *Journal of Global Optimization*, vol. 63, no. 3, 2015, ISSN: 15732916. DOI: 10.1007/s10898-014-0213-z.

[64] P. Serafini, "Simulated Annealing for Multi Objective Optimization Problems", in *Multiple Criteria Decision Making*, 1994. DOI: 10.1007/978-1-4612-2666-6_29.

[65] A. P. Engelbrecht, *Computational Intelligence: An Introduction: Second Edition*. 2007. DOI: 10.1002/9780470512517.

[66] E. G. Talbi, *Metaheuristics: From Design to Implementation*. 2009. DOI: 10.1002/9780470496916.

[67] M. Dorigo and T. Stützle, "The Ant Colony Optimization Metaheuristic", in *Ant Colony Optimization*, 2018. DOI: 10.7551/mitpress/1290.003.0004.

[68] T. Stützle and H. H. Hoos, "MAX-MIN Ant System", *Future Generation Computer Systems*, vol. 16, no. 8, 2000, ISSN: 0167739X. DOI: 10.1016/S0167-739X(00)00043-1.

[69] I. Alaya, C. Solnon, and K. Ghédira, "Ant colony optimization for multi-objective optimization problems", in *Proceedings - International Conference on Tools with Artificial Intelligence, ICTAI*, vol. 1, 2007. DOI: 10.1109/ICTAI.2007.108.

[70] D. Teodorović, "Bee colony optimization (BCO)", *Studies in Computational Intelligence*, vol. 248, 2009, ISSN: 1860949X. DOI: 10.1007/978-3-642-04225-6_3.

[71] K. Deb and H. 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*, vol. 18, no. 4, pp. 577–601, 2013.

[72] C. A. Coello Coello, "A Comprehensive Survey of Evolutionary-Based Multiobjective Optimization Techniques", *Knowledge and Information Systems*, vol. 1, no. 3, 1999, ISSN: 0219-1377. DOI: 10.1007/bf03325101.

[73] K. Doerner, W. J. Gutjahr, R. F. Hartl, C. Strauss, and C. Stummer, "Pareto ant colony optimization: A metaheuristic approach to multiobjective portfolio selection", *Annals of Operations Research*, vol. 131, no. 1-4, 2004, ISSN: 02545330. DOI: 10.1023/B:ANOR.0000039513.99038.c6.

[74] P. C. Fishburn, "Exceptional Paper—Lexicographic Orders, Utilities and Decision Rules: A Survey", *Management Science*, vol. 20, no. 11, 1974, ISSN: 0025-1909. DOI: 10.1287/mnsc.20.11.1442.

[75] M. T. Emmerich and A. H. Deutz, "A tutorial on multiobjective optimization: fundamentals and evolutionary methods", *Natural Computing*, vol. 17, no. 3, 2018, ISSN: 15729796. DOI: 10.1007/s11047-018-9685-y.

[76] T. Lust and J. Teghem, "The multiobjective multidimensional knapsack problem: A survey and a new approach", *International Transactions in Operational Research*, vol. 19, no. 4, 2012, ISSN: 09696016. DOI: 10.1111/j.1475-3995.2011.00840.x.

[77] K. Florios, G. Mavrotas, and D. Diakoulaki, "Solving multiobjective, multiconstraint knapsack problems using mathematical programming and evolutionary algorithms", *European Journal of Operational Research*, vol. 203, no. 1, 2010, ISSN: 03772217. DOI: 10.1016/j.ejor.2009.06.024.

[78] H. Jain and K. Deb, "An improved adaptive approach for elitist nondominated sorting genetic algorithm for many-objective optimization", in *International Conference on Evolutionary Multi-Criterion Optimization*, 2013, pp. 307–321.

[79] N. Srinivas and K. Deb, "Muiltiobjective optimization using nondominated sorting in genetic algorithms", *Evolutionary computation*, vol. 2, no. 3, pp. 221–248, 1994.

[80] W. Mkaouer, M. Kessentini, A. Shaout, *et al.*, "Many-objective software remodularization using NSGA-III", *ACM Transactions on Software Engineering and Methodology (TOSEM)*, vol. 24, no. 3, pp. 1–45, 2015.

[81] H. Jain and K. Deb, "An evolutionary many-objective optimization algorithm using reference-point based nondominated sorting approach, part II: Handling constraints and extending to an adaptive approach", *IEEE Transactions on evolutionary computation*, vol. 18, no. 4, pp. 602–622, 2013.

[82] W. McKinney, "Data Structures for Statistical Computing in Python", in *Proceedings of the 9th Python in Science Conference*, 2010. DOI: 10.25080/majora-92bf1922-00a.

[83] C. R. Harris, K. J. Millman, S. J. van der Walt, *et al.*, *Array programming with NumPy*, 2020. DOI: 10.1038/s41586-020-2649-2.

[84] F. A. Fortin, F. M. De Rainville, M. A. Gardner, M. Parizeau, and C. Gagńe, "DEAP: Evolutionary algorithms made easy", *Journal of Machine Learning Research*, vol. 13, 2012, ISSN: 15324435.

[85] K. Deb and J. Sundar, "Reference point based multi-objective optimization using evolutionary algorithms", in *Proceedings of the 8th annual conference on Genetic and evolutionary computation*, 2006, pp. 635–642.

[86] A. Vargha and H. D. Delaney, "The Kruskal-Wallis Test and Stochastic Homogeneity", *Journal of Educational and Behavioral Statistics*, vol. 23, no. 2, 1998, ISSN: 10769986. DOI: 10.3102/10769986023002170.