Linear Optimization in Disney Travel

Ming Cheng

November 12, 2024

Abstract

In the enchanting world of Disney, where every corner holds a magical surprise, strategic planning is essential for an unforgettable experience. This research employs two models to determine the optimal solution, aiming to maximize visitor enjoyment while ensuring the shortest path for visiting all theme regions. These models stem from classic combination optimization problems: the knapsack problem and the shortest path problem. Utilizing an integer solver, suitable solutions were obtained for both models. It was observed that employing a more complex graph, notably the mixed graph, yielded superior solutions by enabling visitors to explore more attractions. Ultimately, the research integrated the shortest path solution into the enjoyment maximization problem, resulting in a more effective solution, albeit with a sacrifice some the enjoyment.

1 Introduction

In the enchanted kingdom of Disney where dreams and stories intertwine, careful planning is the essence of an unforgettable trip. Recognizing the importance of optimizing itineraries, the research advocates utilizing linear programming to schedule a visit that is not only enjoyable but also structurally efficient. Underlying this approach is a commitment to data completeness and accuracy, with necessary information sourced directly from the official Disney website. The data include average wait times for attractions, walking times between attractions and the favorability ranking of each attraction, providing a comprehensive data set for analysis. The analytical framework for this research is built around two LP models, each designed to address different aspects of the Disney experience.

2 Background

Combinatorial optimization is a dynamic field at the intersection of combinatorics and theoretical computer science. It utilizes combinatorial methods to tackle discrete optimization problems, which involve finding the optimal solution from a finite set of possibilities [7]. In the realm of computer science, the goal of combinatorial optimization is to enhance algorithms either by reducing the size of the potential solution set or by accelerating the search process itself. From the standpoint of combinatorics, it approaches complex problems by employing a fixed repertoire of mathematical structures such as sets, graphs, polytopes and posets.

This field addresses a variety of specific issues. This thesis will focus on applying two particular categories of combinatorial optimization: the shortest path problem and the knapsack problem.

2.1 Knapsack Problem

The earliest work on the knapsack problem was proposed in Mathews' "On the partition of numbers" [8]. The knapsack problem has been recognized for over a century, with its origins tracing back to 1897. According to folklore, the name "knapsack problem" was proposed by Tobias Dantzig. This topic's lasting popularity over the years is highlighted by its ranking as the 18th most popular algorithmic problem and the second among NP-hard problems in Skiena's study [9].

There are many types of knapsack problems. In Cacchiani and Martello's review, they introduced the single knapsack problem in detail, including the 0-1 knapsack problem, subset sum problem, bounded knapsack problem, unbounded knapsack problem, and other variations, while their second part mainly focuses on multiple, multidimensional, and quadratic knapsack problems.

The 0-1 knapsack problem(KP01) is the classic knapsack problem, which is described as follows: t here are n items in a vault, with each item i having a weight of s_i pounds and a value of v_i dollars. The goal is to select items to place in your knapsack in such a way that maximizes the total value of the items inside, while ensuring the total weight does not exceed the knapsack's capacity.We will use C to represent the capacity of knapsack in the model below.

The model is shown as:

$$\max \sum_{j=1}^{n} v_j x_j \tag{1}$$

$$\text{s.t.} \sum_{j=1}^{n} w_j x_j \leq C \tag{2}$$

$$x_j \in \{0, 1\} \tag{3}$$

where v_j is the value of item j, w_j is the weight of item j, and decision variable x_j takes the value of 1 if the item is selected and 0 otherwise.

The Subset Sum problem is a specialized case of the 0-1 Knapsack problem. To transform the Subset Sum problem into a Knapsack instance, we assign each object's value to be equal to its weight and set the knapsack capacity to match the desired target sum. Consequently, the objective becomes maximizing $\sum_{j=1}^{n} w_j x_j$ subject to the constraints outlined in equations (2) and (3) of the 0-1 Knapsack problem.

The Bounded Knapsack problem closely resembles the 0-1 Knapsack problem in its setup. It shares the same objective and constraint (2) as the 0-1 Knapsack problem. However, the variable range of x_j in the Bounded Knapsack problem extends from 1 to k instead of being limited to 0 and 1. To incorporate this, we introduce the additional constraint:

$$0 \leq x_j \leq k_j, \quad x_j \in Z, \quad (j = 1, \ldots, n) \tag{4}$$

where x_i represents the number of selected copies of item type j. Tamir [11] proposed a pseudo-polynomial algorithm with a time complexity of $O(n^3 \max\{w_j\}^2)$, contrasting with the $O(nC)$ algorithm by Kellerer et al. [12], where C is the capacity of the knapsack.

Similarly, the Unbounded Knapsack problem(UKP) follows a similar structure but allows for an unlimited number of copies of each item type. Hence, we also include the constraint:

$$x_i \geq 0, \quad x_i \in Z, \quad (j = 1, \ldots, n) \tag{5}$$

Poirriez and the others [26] introduced a method that melds dynamic programming, dominance rules, and branch and bound techniques to tackle the UKP precisely. Meanwhile, He and his team [27] put forth a hybrid strategy integrating valid inequality generation alongside the aforementioned techniques. They further adapted this approach to address the multidimensional problem.

The problem of the Multiple-Choice Knapsack (MCKP) arises from a generalization of the KP01. In this scenario, the item set is partitioned into l classes, denoted as $N_1, \ldots, N_l$. The objective is to select precisely one item from each class. This problem is also recognized as the knapsack problem with generalized upper bound constraints.This problem has the same setup as equations (1) to (3) in the 0-1 knapsack problem, with an additional bound constraint:

$$\sum_{j \in N_i} x_j = 1 \quad (i = 1, \ldots, l). \tag{6}$$

The constraint (6) can use $\leq$ sign instead of $=$ [13]. Sbihi [15] developed a reactive Tabu search algorithm for a variant of the MCKP arising in budget planning over discrete periods responding to classes and having individual capacities. Bednarczuk et al. [16] presented a heuristic approach that removes the capacity constraint (2) and solves a bi-objective problem that maximizes the total profit and minimizes the total weight.

Similarly, my model is a variation of the knapsack problem, particularly resembling the MCKP. Like the multiple-choice knapsack, my model also organizes items into different "bin", but instead of setting an upper bound for the value of items like the multiple-choice knapsack problem, my model sets a lower bound. Moreover, while the multiple-knapsack problem allows any item to be added to any bin, my model specifies that each bin can only select from certain items. This increases the complexity of the model but better reflects the real-world situation it aims to simulate.

2.2 Shortest Path Problem

In graph theory, which explores mathematical structures modeling relationships between pairs of objects, the shortest path problem stands as a pivotal challenge. This involves finding a route between two nodes in a network that minimizes the sum of the edges' weights [25]. Applicable to various types of graphs, including undirected and directed, this problem seeks the most efficient path between points.

The shortest path problem appears in several variations [6], including:

- **Single-source shortest path (SSSP):** Finding the minimal-weight path from one node to all others in a weighted graph.
- **Breadth-first search (BFS):** Identifying the shortest path in terms of edge count in an unweighted graph from a source node to all others.
- **All-pairs shortest path (APSP):** Determining the shortest paths between every pair of nodes in a graph.
- **Single-source widest path (SSWP):** Locating the path that maximizes the weight of the minimum-weight edge from a source node to all others.

Among the numerous algorithms developed to address these problems, Dijkstra's algorithm is particularly notable for solving the SSSP problem with non-negative edge weights. Invented by Dijkstra in 1956, this algorithm was initially designed to determine the shortest route between two cities within a network of roads, coinciding with the development of efficient algorithms for network design, such as the minimum spanning tree algorithm [23].

While Dijkstra's algorithm focuses on calculating distances, it also allows for the reconstruction of the shortest paths by tracking the predecessors of each node. This feature underscores the algorithm's efficiency and adaptability in solving the shortest path problem, highlighting its importance in both theoretical and practical applications [4].

In addition to Dijkstra's algorithm, there are several other approaches for solving shortest path problems [24]. For example, the Bellman–Ford algorithm addresses the SSSP problem with negative edge weights. The A* algorithm optimizes single-pair shortest path searches using heuristics. The Floyd–Warshall and Johnson's algorithms are efficient for solving APSP problems, with Johnson's algorithm being particularly suited to sparse graphs. The Viterbi algorithm tackles the shortest stochastic path problem by adding a probabilistic weight to each node.

The widespread applicability of shortest path algorithms underscores their significance. They are essential in real-world scenarios such as optimizing transportation networks to minimize travel distances or enhancing urban planning to achieve greater efficiency [5].

2.3 Mixed Graph

In graph theory, a mixed graph is a graph that contains both directed and undirected edges. Such a graph is denoted as $G = (V, E, A)$ where V is the set consisting of all the vertices, E is the set of undirected edges, and A is the set of directed edges (or arcs), represented by (u, v) for directed edges, and $[u, v]$ for undirected edges.

Mixed graphs are commonly used in coloring problems. Similar to the traditional coloring problem, one aims to find the chromatic number $\gamma_\pi(G)$, which is the smallest k such that the graph G admits a k-coloring. Here, k represents the total number of colors, and we assign integers less than k to each vertex so that adjacent vertices have different colors, and the tail of any arc has a smaller color than its head [20]. Ries and Werra [17] introduced two problems: the traditional mixed graph coloring problem and one allowing vertices connected by directed edges to have the same color. Both problems demonstrate polynomial solvability in partial k-trees, for fixed k. This result aligns with conclusions from Ries [18] on the complexity results guaranteed for such problems, showing that the strong mixed graph coloring problem is NP-complete for planar bipartite graphs and for bipartite graphs with a maximum degree of 4 [17].

Mixed graphs also find utility in other problems. In the paper by Kocsány and Szádeczky-Kardoss [19], they primarily discuss mixed graph traversal optimization for the Vehicle Routing Problem. Their results indicate that the parking space search problem in a mixed graph can be reduced to a Traveling Salesman Problem, for which an Ant Colony Optimization algorithm is implemented to minimize the total traversal cost of the mixed graph.

2.4 CBC Solver

The Computational Infrastructure for Operations Research (COIN-OR) branch-and-cut solver, known as CBC, stands as a beacon of open-source innovation in the realm of mixed-integer programming (MIP) solutions. Its prowess lies not only in its ability to efficiently solve complex problems but also in its adaptability to diverse customization needs. CBC empowers operations research professionals with a versatile toolkit, offering a robust code base that invites exploration and experimentation [1].

Diving into the intricacies of branch-and-cut algorithms, CBC emerges as a platform where customization flourishes. From tailoring the traversal order of nodes in the search tree to employing specialized branching criteria, CBC facilitates advanced adjustments crucial for problem-specific optimization. Furthermore, its design seamlessly integrates with tailored heuristics for swift generation of MIP-feasible solutions and supports the inclusion of both standard and user-defined cut generation techniques within the linear program (LP) relaxations.

Implemented in C++, CBC serves as a callable library, inviting developers to harness its capabilities within their own applications. Leveraging the COIN-OR open solver interface (OSI), CBC fosters interoperability with a spectrum of LP solvers, with the COIN-OR LP solver (CLP) emerging as a popular choice [1]. Whether utilized as a standalone branch-and-bound solver or as a comprehensive branch-and-cut solution, CBC stands ready to tackle optimization challenges.

In its quest for efficiency, CBC leans on the COIN-OR Cut Generation Library (CGL) for cut generation, ensuring compatibility with a variety of industry-standard cut generators. This symbiotic relationship with other components of the COIN repository underscores CBC's collaborative nature, reinforcing its position as a cornerstone of the Operations Research community.

2.5 SCIP

SCIP (Solving Constraint Integer Programs) is primarily designed as a comprehensive branch-cut-and-price framework. This framework is adept at solving various optimization problems, with a particular focus on Mixed-Integer Linear Programs (MILPs) and Mixed-Integer Nonlinear Programs (MINLPs).

The computational foundation of SCIP is the branch-and-bound algorithm, a well-established method for finding optimal solutions to combinatorial optimization problems. This algorithm partitions the feasible set into smaller, convex subsets, calculating lower and upper bounds for each. The main process involves:

1. Computing initial bounds for the solution space.
2. Using the branch-and-price technique, where the linear relaxation of each node in the branch-and-bound tree is solved using column generation.
3. Optionally incorporating cutting planes to strengthen the relaxation, referred to as branch-price-and-cut.

According to Desrosiers [2], the strategy begins by excluding sets of columns from the LP relaxation to decrease both computational and memory demands. Columns are then reintroduced into the LP relaxation as necessary. This method leverages the observation that in large-scale problems, most columns remain non-basic and have corresponding variable values of zero at the optimal solution, indicating their irrelevance in finding the solution.

The process often starts with a reformulation technique, such as the Dantzig-Wolfe decomposition, to establish what is known as the Master Problem. This decomposition is strategically performed to obtain a problem formulation that yields superior bounds when the relaxation is solved, compared to the bounds obtained from the original problem formulation's relaxation.

3 Data

Since my model is analyzing the Walt Disney Magic Kingdom theme park, most of the data I use comes from Disney's official website and Thrill Data, including Magic Kingdom's attractions' average waiting time, which is calculated from November 2023 to April 2024. The averages of wait times for different time periods in April were calculated [22], walking times between attractions, and average walking times [21].

The ranking of attractions is very subjective. In order to ensure the authenticity and objectivity of the data as much as possible, I extracted the ranking data of multiple websites and calculated the average of the rankings of projects provided by different resources as the project ranking of my model. This approach aims to avoid excessive bias. Since these project rankings are averages, these rankings are in fractional form rather than our traditional integer rankings.

4 Models

4.1 Maximizing Enjoyment

The first model is designed to optimize the overall enjoyment of a visit within the confines of an h-hour day. This is implemented through an integer programming approach that seeks to maximize visitor satisfaction based on the popularity rankings of each attraction. Here, e_i denotes the popularity ranking of attraction i, and x_i, our decision variables, are binary, taking the value of 1 if attraction i is chosen and 0 otherwise.

The objective of this model is:

$$\text{Max} \sum_{i=1}^{n} p_i x_i \tag{7}$$

Subject to:

$$\sum_{i=1}^{n} T_i x_i + 7(\sum_{i=1}^{n} x_i - 1) \leq 60 * h \tag{8}$$

$$x_1 + x_2 + x_3 + x_4 \geq 1 \tag{9}$$

$$x_5 + x_2 + x_6 \geq 1 \tag{10}$$

$$x_7 + x_8 + x_3 + x_9 + x_{10} + x_{11} \geq 1 \tag{11}$$

$$x_{12} + x_{13} + x_{14} \geq 1 \tag{12}$$

This integer program incorporates two key constraints. The first constraint addresses the time limitation. It ensures that the total waiting time for selected attractions, combined with the average walking time between successive attractions, remains within the $60h$ minutes duration of the park visit. Mathematically, this is expressed as the sum of the waiting times for all chosen attractions plus the walking times between them must not exceed the available time in the park.

We implement $(\sum_{i=1}^{n} x_i - 1)$ in constraint (8) to represent the total walking time. This is because we have a total of n attractions, and the total number of paths in between is the sum of the total selected attractions minus one. In this constraint,"7" represents the average walking time from one attraction to another.

Additionally, to ensure a comprehensive experience covering all thematic areas of the park, the model includes constraints that at least one attraction from each thematic region must be selected. These are detailed in constraints (9) to (12), which mandate the selection of at least one attraction per theme region to fulfill the requirement of thematic coverage.

4.2 Minimizing the Walking Time

The second model under consideration is an integer program that utilizes decision variables x_{ij} and parameters T_{ij}. In this setup, x_{ij} is a binary variable where $x_{ij} = 1$ indicates the selection of path ij, and $x_{ij} = 0$ indicates it is not chosen. The parameter T_{ij} represents the walking time between attractions. The objective of this model is to minimize the total walking time across all thematic regions within Disney, with each theme consisting of several attractions represented as nodes in Figure 1.

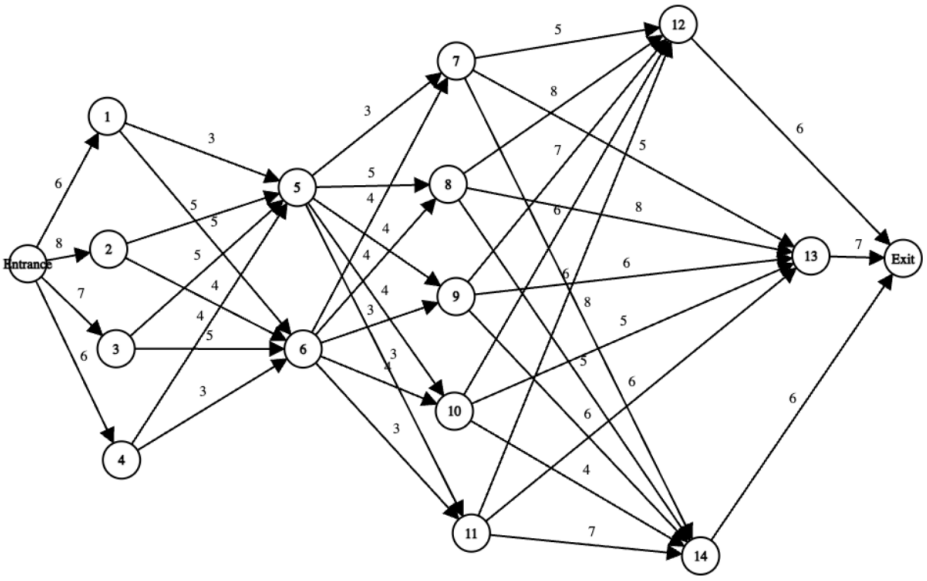

Figure 1: Finding the shortest path in Disney (Simple Directed Graph)

In addition to the pathways connecting each theme area as depicted in Figure 1, Figure 2 introduces inner-connected paths between attractions within the same theme area. This feature provides visitors with greater flexibility to explore any attraction within a theme area, while adhering to walking time constraints. This mixed graph design closely resembles real-world scenarios, with nodes denoted by "En" representing entry points and "Ex" representing exits.

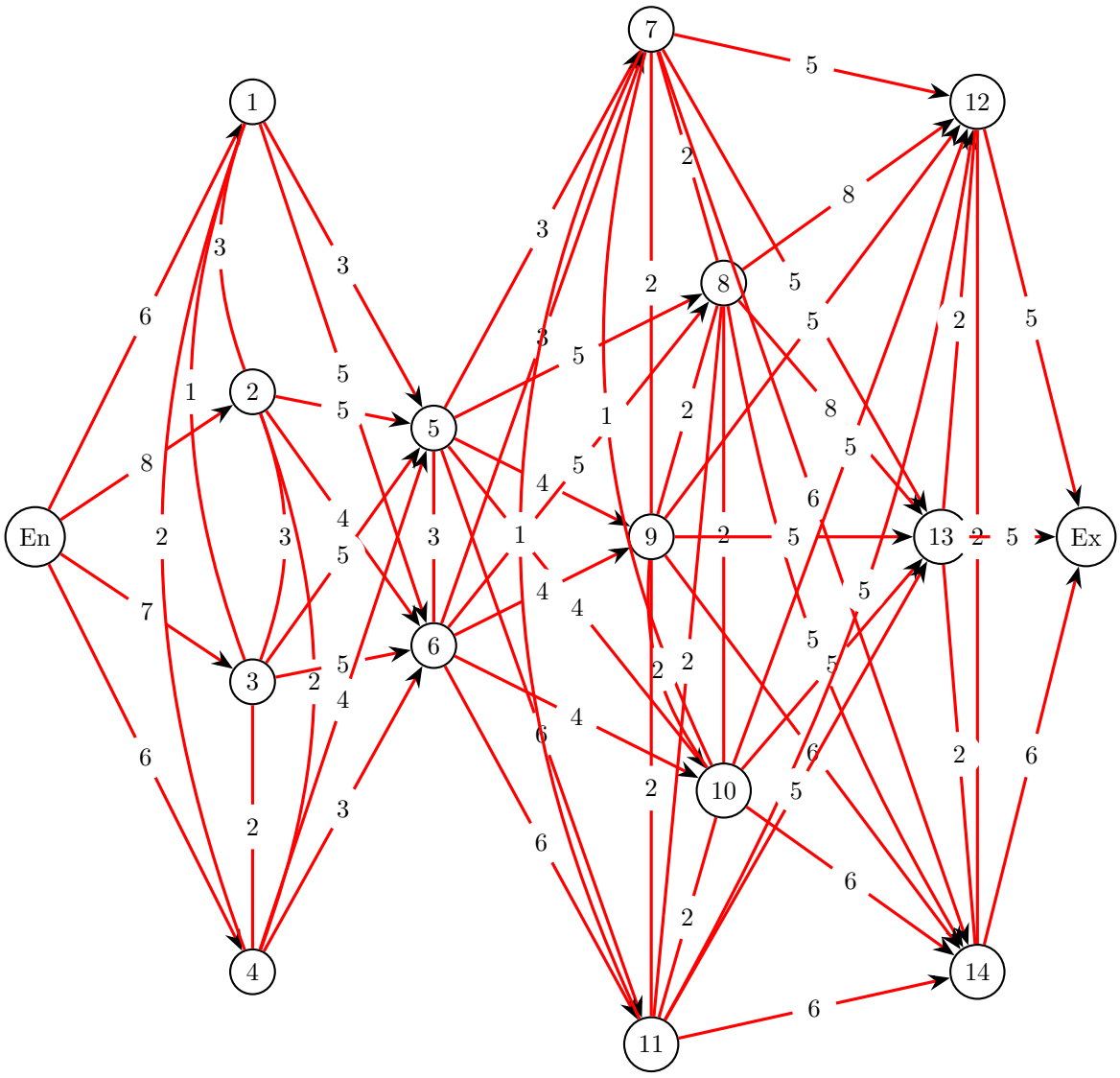

Figure 2: Finding the shortest path in Disney (Mixed Graph)

The model is showing below:

$$\text{Min} \sum T_{ij} x_{ij} \tag{13}$$

Subject to:

$$x_{En1} + x_{En2} + x_{En3} + x_{En4} = 1 \tag{14}$$

$$x_{A1} = \sum_{i=5}^{6} x_{1i} \tag{15}$$

$$x_{A2} = \sum_{i=5}^{6} x_{2i} \tag{16}$$

$$x_{A3} = \sum_{i=5}^{6} x_{3i} \tag{17}$$

$$x_{A4} = \sum_{i=5}^{6} x_{4i} \tag{18}$$

$$\sum_{i=1}^{4} x_{i5} = \sum_{i=7}^{11} x_{5i} \tag{19}$$

$$\sum_{i=1}^{4} x_{i6} = \sum_{i=7}^{11} x_{6i} \tag{20}$$

$$\sum_{i=5}^{6} x_{i7} = \sum_{i=12}^{14} x_{7i} \tag{21}$$

$$\sum_{i=5}^{6} x_{i8} = \sum_{i=12}^{14} x_{8i} \tag{22}$$

$$\sum_{i=5}^{6} x_{i9} = \sum_{i=12}^{14} x_{9i} \tag{23}$$

$$\sum_{i=5}^{6} x_{i10} = \sum_{i=12}^{14} x_{10i} \tag{24}$$

$$\sum_{i=5}^{6} x_{i11} = \sum_{i=12}^{14} x_{11i} \tag{25}$$

$$\sum_{i=7}^{11} x_{i12} = x_{12Ex} \tag{26}$$

$$\sum_{i=7}^{11} x_{i13} = x_{13Ex} \tag{27}$$

$$\sum_{i=7}^{11} x_{i14} = x_{14Ex} \tag{28}$$

$$x_{12Ex} + x_{13Ex} + x_{14Ex} = 1 \tag{29}$$

This is a typical shortest path problem formulated on a directed weighted graph, where each edge's cost is determined by the walking time between nodes. To identify the shortest route through the park, starting from the entrance (source node) and concluding at the exit (sink node), the model ensures that for each selected node, there is precisely one incoming and one outgoing path. This is achieved by ensuring that each node, if chosen, has exactly one incoming and one outgoing connection, which keeps the path continuous and directed. Conversely, for any node that is not part of the selected path, both the sum of its incoming and outgoing pathways should be zero, ensuring no discontinuities in the route.

5 Result of Models

5.1 Result and Discussion of the Maximizing Enjoyment Problem

The solution we described for this model utilizes the CBC (Coin-or branch and cut) solver provided by Google's OR-Tools. It is widely used for solving linear and integer optimization problems, making it a suitable choice for problems like the modified multiple knapsack problem. In the context of the multiple knapsack problem with multiple constraints and groups, CBC's capabilities are particularly useful. It can effectively handle the binary decision variables and the linear constraints that define both the weight limits and the requirement to select at least one item from each bin. Moreover, CBC's ability to handle such mixed integer programming challenges ensures that the solution is not only optimal according to the set objective function but also adheres to all specified constraints, such as the minimum item requirements from each group and total weight constraints.

The implementation of this model addresses a variant of the multiple knapsack problem that significantly differs from the classic knapsack problem in several ways, particularly in how items are grouped and selected. In the classic knapsack problem, the goal is typically to maximize the total value of items placed into a single knapsack without exceeding a weight limit. In contrast, this modified

version introduces multiple "bins" or groups, each requiring at least one item to be selected, which introduces a layer of complexity in decision-making.

This partitioning ensures that each group contributes at least one attraction, reflecting real-world scenarios where attractions belong to different theme regions that need to be represented in the solution. The optimal solution achieved an objective value of approximately 77.1 with a total time of 420, distributed across the group as follows:

- **group 1:** Attraction 4 was selected
- **group 2:** Attraction 6 was selected, illustrating the necessity of picking from smaller groups.
- **group 3:** Attractions 7, 9, and 10 were selected, indicating this group's significant contribution to the total value and weight, likely due to the higher ranking of attractions within this region.
- **group 4:** Attractions 12, 13, and 14 were all selected, showing a dense selection from this bin, perhaps due to the favorable weight-values ratio contributing effectively towards optimizing the objective.

The result shows that to achieve the maximum enjoyment based on the attractions' ranking (77.1 in total) within the 60*8-minute time constraint, we need at least 420 minutes, where attractions 4, 6, 7, 9, 10, 12, 13, and 14 are selected.

Unlike the classic problem, where the decision is purely based on individual item merit relative to weight and value, this problem requires at least one item from each predetermined group, adding a constraint that necessitates strategic selection across diverse groups.

The result suggests that the solution effectively balanced the requirements to maximize value while adhering to the time constraint and the need for diverse representation from each group. The selected items from group 3 and group 4 show a heavier weight contribution, which could indicate these items were critical in achieving a higher value or ranking within the allowed time limit. This strategic selection highlights the model's complexity and its capability to address layered constraints effectively, showcasing its applicability to more nuanced scenarios than those addressed by the classic knapsack problem.

5.2 Result and Discussion of the Shortest Path Problem

I implemented a shortest path optimization model using Google's OR-Tools, specifically utilizing the SCIP solver. More details about this implementation will be discussed in the following section. The model is structured to represent a directed graph where nodes are organized into distinct groups, each representing a distinct theme area. Each node within a group can only connect to nodes in the subsequent group with directed edges. The script begins by defining nodes and categorizing them into groups such as 'Entry', 'Group1', 'Group2', 'Group4', and 'Exit'. It then specifies the inter-group connections along with their respective weights, which dictate the walking time required to traverse from one node to another.

Flow conservation constraints ensure that the amount of flow entering a node equals the amount of flow leaving it.For the Entry and Exit, specific constraints are added. These constraints ensure exactly one outgoing flow from the Entry and exactly one incoming flow to the Exit. Additionally, no outgoing and incoming flow is allowed from the Exit and Entry. This setup ensures the path starts at the source ('Entry') and ends at the sink ('Exit') without any additional paths extending beyond the endpoint.

Analysis of Two Different Graph Structures Using the SCIP Solver:

Simple Directed Graph: This graph structure strictly contains edges from one group to another in a directed fashion. The optimal path found was:

```
Entry -> x1 -> x5 -> x7 -> x12 -> Exit
```

with a total minimum traversal time of 22 minutes. This path represents the quickest route under the constraint that only one attraction can be visited in each group.

Mixed Graph with Inner-group Connections: This more complex structure allows for bidirectional connections within each group, offering more flexibility in the choice of paths. The optimal path for this configuration was:

```
Entry -> x1 -> x6 -> x8 -> x10 -> x12 -> Exit,
```

taking only 21 minutes. This result not only suggests a quicker traversal time compared to the simple directed graph but also includes more attractions, enhancing the visitor experience.

The mixed graph allows for a richer set of pathways by incorporating inner-group flexibility. This additional complexity leads to potentially better outcomes, as demonstrated by the shorter optimal path time and the inclusion of more attractions. The presence of inner-group connections significantly impacts the optimization results, providing a path that maximizes the number of attractions visited while minimizing the time spent. This suggests that incorporating the mixed graph into the model can lead to solutions that are not only optimal in terms of cost or time but also enhance the overall experience. The results underline the importance of graph structure in optimization problems. For practical applications like planning visitor routes in theme parks, the introduction of realistic elements such as inner-group paths can make a substantial difference. It allows for a more enjoyable visitor experience by offering more flexibility in the choice of attractions without increasing the total transit time.

Comparing the two graph models highlights the benefit of using a more detailed and realistic graph representation. It facilitates finding better solutions at potentially lower costs, underscoring the value of detailed modeling in operational and strategic planning. This approach maximizes visitor satisfaction and operational efficiency, making it highly beneficial for scenarios where user experience and time are critical factors, such as in theme park management or urban transport planning.

6 Integrating Shortest Path Solutions into the Enjoyment Maximization Model

6.1 Integrated Shortest Path Problem into Enjoyment Maximization Problem

In this section, we aim to refine the enjoyment maximization model by integrating solutions from the shortest path problem. Retaining the original objective and constraints (7)–(12), we introduce additional constraints to incorporate these solutions more effectively. The new constraints are specified as follows:

$$x_1 = 1 \tag{30}$$

$$x_6 = 1 \tag{31}$$

$$x_8 = 1 \tag{32}$$

$$x_{10} = 1 \tag{33}$$

$$x_{12} = 1 \tag{34}$$

The modified model, with its stringent constraints ensuring the inclusion of specific items (x1, x6, x8, x10, x12), has led to an optimized solution with a different objective and output compared to the previous model. This setup provides a practical example of integrating solutions from a shortest path problem into a broader objective of enjoyment maximization, such as visiting attractions efficiently in a theme park scenario.

The solution output for the new model is as follows:

Objective Value: 74.2, indicating the total value of the selected items.

- **group 1:** Attraction 1 and 4 were selected
- **group 2:** Attraction 6 was selected
- **group 3:** Attractions 8, 9, 10 and 11 were selected, indicating this group's significant contribution to the total value
- **group 4:** Attractions 12, 13 were selected

Total Selected Weight: 417

The inclusion of items mandated by the shortest path solution (x1, x6, x8, x10, x12) reflects an operational strategy where paths are optimized perhaps for time or distance. By locking in these attractions, the model mimics a scenario where certain key routes or stops are prioritized for operational efficiency. The total value of the selected items decreased from 77.1 to 74.2. This reduction of approximately 3 units highlights a trade-off between operational efficiency and maximal enjoyment. Essentially, prioritizing certain items for their strategic importance, such as minimizing walking times between attractions, does result in a slight compromise on the overall value.The adjustment in selected items led to a decrease in total weight from 420 to 417, symbolizing a small but notable increase in efficiency. This suggests that the paths chosen not only fit operational criteria but also do so with a slightly reduced "cost".

Most items are from group 3, indicating a concentration of value or strategic importance in this group under the new constraints. Conversely, group 4, previously the most selected group, has fewer selections, showing a shift in focus based on the new operational parameters.

The outcome demonstrates a practical application of optimization where efficiency is balanced with enjoyment. The model suggests that adhering strictly to a path optimized for efficiency can lead to overall effective outcomes with marginal reductions in total value. For a theme park visitor, this might mean spending less time walking, thus potentially increasing the qualitative aspect of the visit despite a quantitative drop in total value scored.

This approach can be particularly useful in scenarios where time is limited or when certain attractions or paths must be prioritized due to external factors ,such as scheduled events or anticipated crowding, offering a pragmatic balance between seeing more with a slight compromise on the peak possible enjoyment.

In summary, the model highlights how integrating strategies from different optimization problems,such as shortest paths and knapsack problem, can yield solutions that while not optimal by one metric alone, provide a balanced, practical outcome that could enhance overall experience or efficiency in real-world applications.

6.2 Integrated Enjoyment Maximization Problem into Shortest Path Problem

We integrated the solution from the enjoyment maximization problem, selecting nodes x_4, x_6, x_7, x_9, x_{10}, x_{12}, x_{13}, and x_{14} into our shortest path problem. The resulting path is as follows:

$$\text{Entry} \to x_4 \to x_6 \to x_7 \to x_9 \to x_{10} \to x_{12} \to x_{13} \to x_{14} \to \text{Exit}$$

This path not only adheres to the shortest path constraints but also ensures maximization of enjoyment by incorporating strategically selected nodes that enhance the overall experience. The selected path represents a blend of both objectives, aiming to optimize the route while enriching the journey with high-value attractions.

The chosen path leads to an objective value of 30, which is approximately 33% greater than the objective value in the simple mixed graph shortest path solution. This increase in the objective value signifies a substantial time trade-off in exchange for enhanced enjoyment. Such a trade-off might be considered acceptable if the total time remains within our allowable limits, suggesting a balance between efficiency and experiential quality.

The integration of enjoyment maximization distinctly alters the path compared to a straightforward shortest path approach. By selecting additional attractions, specifically from the fourth theme area, the path underscores the significance of certain attractions that contribute extensively to visitor satisfaction. This strategic choice highlights the importance of Theme Area 4 in enhancing the visitor experience, suggesting its high priority in route planning.

While this model successfully balances between efficiency and enjoyment, it does not incorporate some potentially critical factors such as waiting times, which could affect the actual time spent and thus the feasibility of the path under real conditions. Moreover, the greater objective value compared to the simple model indicates a higher trade-off, which may not always be desirable, especially under stricter time constraints.

Considering the trade-offs and the omission of certain dynamic factors like waiting times, our analysis leans towards preferring the previous model. The earlier model provided a more efficient

solution with more selected attractions, implying a better optimization of time while still enhancing enjoyment. This makes the previous model a more comprehensive and practical choice for scenarios where time efficiency is paramount.

7 Conclusion

This thesis delved into the sophisticated application of integer programming to optimize visitor experiences at theme parks by addressing two distinct but interrelated challenges: maximizing enjoyment and minimizing walking time. Initially, we designed a model to maximize enjoyment by utilizing attraction popularity rankings, ensuring that visitors could experience a well-rounded visit within a specific time frame. This model was carefully structured to not only prioritize attractions based on their appeal but also to incorporate logistical considerations by enforcing a series of constraints that assured coverage across various thematic areas and managed total visiting time effectively.

Subsequently, we integrated the principles of the shortest path problem into the enjoyment maximization model to refine the route selection process further. This integration was aimed at enhancing operational efficiency by specifying paths that minimized walking times without significantly detracting from overall enjoyment. By enforcing the selection of certain key attractions identified through shortest path analysis, the new model achieved a delicate balance between reducing travel time and maintaining high visitor satisfaction.

The insights from both models combined to demonstrate the powerful synergy between maximizing value, based on attraction popularity, and operational efficiency, focused on time management. This approach not only highlighted the utility of mixed integer programming in complex real-world applications but also underscored the importance of strategic constraint management. The results demonstrated that even with stringent requirements, it is possible to craft solutions that meet multiple objectives effectively. This enhances the overall quality of visitor experiences in theme parks.

Ultimately, the exploration confirmed that sophisticated optimization techniques could provide substantial benefits in settings that demand a balance between enjoyment and efficiency. By carefully aligning the models with realistic operational constraints and visitor preferences, we showcased how theoretical models could be practically applied to improve both satisfaction and logistical efficiency in a dynamic environment like a theme park. This serves as a compelling example of how advanced mathematical modeling can be crucial in strategic planning and operational management across various industries.

References

[1] John Forrest, Robin Lougee-Heimer, CBC User Guide, *In INFORMS TutORials in Operations Research*,Published online: 14 Oct 2014; 257-277, https://doi.org/10.1287/educ.1053.0020

[2] Jacques Desrosiers, Marco E. Lübbecke, Branch-Price-and-Cut Algorithms, *Wiley Encyclopedia of Operations Research and Management Science*, (2010),https://or.rwth-aachen.de/files/research/publications/branch-and-price.pdf

[3] M. J. Todd, *Linear and Quadratic Programming in Oriented Matroids*, Journal of Combinatorial Theory, Series B 39, 105-133 (1985)

[4] Sniedovich, Moshe. "Dijkstra's algorithm revisited: the dynamic programming connexion." *Control and Cybernetics*, 35.3 (2006): 599-620. http://eudml.org/doc/209437.

[5] Zhao Jun-Jun, Liu Shi-Feng, Zhang Zhi-Yun, *Traffic Shortest Path Application Based on the Network Segmentation Technology*,Journal of Kuming Metallurgy College, 31(5): 60-64 (2015).

[6] "Shortest Path Problem." NVIDIA Developer, developer.nvidia.com/discover/shortest-path-problem,(2024).

[7] Maltby, Henry, and Eli Ross. "Combinatorial Optimization: Brilliant Math & Science Wiki." *Brilliant*, (2024), brilliant.org/wiki/combinatorial-optimization/.

[8] G. B. Mathews. On the partition of numbers. *Proceedings of the London Mathematical Society*, s1-28:486–490, 1896.

[9] S. Skiena. Who is interested in algorithms and why?: lessons from the stony brook algorithms repository. *ACM SIGACT News*, 30:65–74, 1999.

[10] Thomas E. O'Neil, 0/1-Knapsack vs. Subset Sum: A Comparison using AlgoLab,*49th Annual Midwest Instruction and Computing Symposium* ,New York,2016,pp. 137 -146.

[11] A. Tamir. New pseudopolynomial complexity bounds for the bounded and other integer knapsack related problems.*Operations Research Letters*, 37:303–306, 2009.

[12] H. Kellerer, U. Pferschy, and D. Pisinger. *Knapsack Problems*. Springer Nature Book Archives Millennium. Springer, 2004

[13] R. Bellman. Comment on Dantzig's paper on discrete variable extremum problems.*Operations Research*, 5:723–724, 1957.

[14] C. He, J.Y.T. Leung, K. Lee, and M.L. Pinedo. An improved binary search algorithm for the multiple-choice knapsack problem. *RAIRO - Operations Research*, 50:995–1001, 2016.

[15] A. Sbihi. Adaptive perturbed neighbourhood search for the expanding capacity multiple-choice knapsack problem. *Journal of the Operational Research Society*, 64:1461–1473, 2013.

[16] E.M. Bednarczuk, J. Miroforidis, and P. Pyzel. A multi-criteria approach to approximate solution of multiple-choice knapsack problem. *Computational Optimization and Applications*, 70:889–910, 2018.

[17] B. Ries, D. de Werra,On two coloring problems in mixed graphs,*European Journal of Combinatorics*, 29:3,2008,pp. 712-725.,https://doi.org/10.1016/j.ejc.2007.03.006.

[18] B. Ries,Coloring some classes of mixed graphs,*Discrete Applied Mathematics*,155:1,2007,pp. 1-6., https://doi.org/10.1016/j.dam.2006.05.004.

[19] K. László, S.K. Emese. "Application of mixed graph traversal optimization for the vehicle routing problem"., 20222149-2154. 10.23919/ECC55457.2022.9838025.

[20] Hansen, P., Kuplinsky, J. de Werra, D. Mixed graph colorings. *Mathematical Methods of Operations Research 45*, 145–160, 1997. https://doi.org/10.1007/BF01194253

[21] "Destinations." *Walt Disneyworld*, 2024, disneyworld.disney.go.com/destinations/.

[22] "Wait Times at Walt Disney World." *Thrill Data*, 2024, www.thrill-data.com/waits/chain/wdw/.

[23] Hamilton, Richards. "Edsger Wybe Dijkstra." *Edsger W. Dijkstra - A.M. Turing Award Laureate*, 2019, amturing.acm.org/award_winners/dijkstra_1053701.cfm.

[24] Cherkassky, Boris V.; Goldberg, Andrew V.; Radzik, Tomasz . "Shortest paths algorithms: theory and experimental evaluation". *Mathematical Programming*,1996, Ser. A. 73 (2): 129–174. doi:10.1016/0025-5610(95)00021-6. MR 1392160.

[25] "Shortest Path Problem." *Wikipedia*, Wikimedia Foundation, 28 Apr. 2024, en.wikipedia.org/wiki/Shortest_path_problemCITEREFCherkasskyGoldbergRadzik1996.

[26] V. Poirriez, N. Yanev, and R. Andonov. A hybrid algorithm for the unbounded knapsack problem. *Discrete Optimization*, 6:110–124, 2009.

[27] X. He, J.C. Hartman, and P.M. Pardalos. Dynamic-programming-based inequalities for the unbounded integer knapsack problem. *Informatica*, 27:433–450, 2016.