

Maximizing Fun at a Theme Park: the M⁷TP.

Michael Cardiff, Gwyneth Hughes, and Robert Bosch*

October 31, 2000

A large theme park presents its guests with an interesting and challenging problem: choosing rides to visit and an order in which to visit them. To solve this problem, visitors must balance profits (enjoyment obtained from rides) and costs (time spent waiting for rides and walking from one ride to another). In this article, we describe how to use discrete optimization—in particular, a straightforward modification of a classic integer programming (IP) formulation of the Traveling Salesperson Problem (TSP)—to solve this problem and thereby maximize fun at a theme park. To illustrate, we solve several instances of the M⁷TP, the problem of planning an itinerary for the Walt Disney World Magic Kingdom theme park (M⁷TP stands for “Maximizing Mirth and Merriment on Mickey Mouse’s Magical Mystery Tour Problem”).

Background

In the fall semester of 1999, two of the authors (Cardiff and Hughes) were enrolled in “Optimization,” an upper-level undergraduate mathematics class taught at Oberlin College by the third author (Bosch). One of the course requirements was a modeling project. For several weeks, Cardiff and Hughes spent many an hour brainstorming, trying to find a topic that was both useful and interesting. When fall break arrived, having not yet agreed upon a topic, they decided to put their mathematical woes on the back burner and escape to Disney World. While there, they made two observations:

1. First-time visitors to Disney World tend to spend a large proportion of time getting oriented and deciding which attractions to experience.
2. While most visitors to the Magic Kingdom are very happy—in accordance with Disney World’s claim that it is “The Happiest Place on EarthTM”—many small children and their parents appear to be in conflict.

*Dept. of Mathematics, Oberlin College, Oberlin, OH 44074 (bobb@cs.oberlin.edu)

Upon returning to Oberlin, Cardiff and Hughes realized that they could have used optimization to plan their visit to the Magic Kingdom. They discovered that the M⁷TP is very similar to what is known in the optimization field as the Selective Traveling Salesperson Problem (STSP). They purchased a guide book, [9], that contains ride durations and provides 0-to-5-star “fun factor” ratings for various age groups. For their modeling project, they decided to

1. construct an IP formulation of the M⁷TP,
2. use the information contained within [9] to construct M⁷TP instances for preschoolers and their parents,
3. solve the corresponding “preschool” and “parent” integer programming problems using CPLEX, a state-of-the-art linear optimizer, and then
4. examine differences between the respective optimal tours for preschoolers and parents in the hope of finding an explanation for the observed large number of parent/child disputes.

Cardiff and Hughes’s IP formulation of the M⁷TP is similar to Laporte and Martello’s IP formulation of the STSP [8] and can be thought of as a simple modification of Dantzig, Fulkerson, and Johnson’s seminal IP formulation of the TSP [2].

In the next section of this article, we discuss the TSP and describe Dantzig, Fulkerson, and Johnson’s IP formulation. In the final section, we present Cardiff and Hughes’ IP formulation of the M⁷TP, describe the strategy they followed when they solved the resulting integer programming problems, and discuss their results. In an appendix, we show how to use Excel to solve a small M⁷TP instance.

The TSP

The TSP is concerned with a salesperson, based in city 1, who must visit each of cities $2, 3, \dots, n$ exactly once and then return home. The TSP asks the following question: If the salesperson’s goal is to minimize total distance traveled, in what order should he or she visit the cities? Throughout this article, we assume that for each $1 \leq i < j \leq n$, the distance $c_{i,j}$ from city i to city j is equal to the distance $c_{j,i}$ from city j to city i .

For some sets of cities, the solution to the corresponding TSP instance is completely obvious. It is clear that if the cities are located on the circumference of a circle, as in Figure 1, then the optimal solution is to travel along the circumference, stopping at each city encountered on the way. For many sets of cities, however, the corresponding TSP instance seems to be very difficult. If the cities’ locations are distributed at random, as in Figure 2, a great deal of effort

may be required to find the optimal tour. Since there are $(n - 1)!/2$ possible tours and since $(n - 1)!/2$ grows very rapidly as n increases, a solution strategy based on the enumeration and examination of all possible tours certainly isn't viable. Even a computer that could generate and evaluate tours at a rate of one billion tours per second would require more than 60 million years to go through every one of the $(20 - 1)!/2 \approx 6.08 \times 10^{16}$ tours that are associated with a 20-city instance!

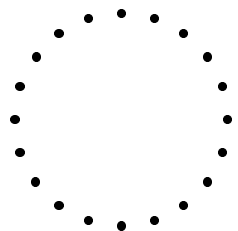


Figure 1

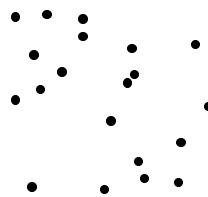


Figure 2

It should be noted that heuristic methods, such as “nearest neighbor” (which constructs a tour by moving from city 1 to its nearest neighbor, then to the nearest neighbor of city 1’s nearest neighbor, and so on) are designed to produce tours that are good but not necessarily optimal. They tend to perform very well on some TSP instances and very poorly on others. We do not consider them here; instead, we refer the interested reader to [6], [7], and [3].

In 1954, Dantzig, Fulkerson, and Johnson wrote a remarkable and extremely influential article that describes how integer programming can be used to find optimal solutions to large scale TSP instances. In 1954, “large scale” meant forty or fifty cities. Dantzig, Fulkerson, and Johnson’s example problem had 49 cities (the 48 U.S. state capitols and Washington, D.C.). Today, “large scale” means thousands of cities. In 1992, Applegate, Bixby, Chvátal, and Cook obtained an optimal solution to a 3,038 city TSP instance, an achievement considered by *Discover* magazine to be one of the top 50 science stories of the year. And in 1998, Applegate, Bixby, Chvátal, and Cook obtained an optimal solution to a TSP instance with 13,509 cities (all cities in the continental U.S. with population greater than 500). In August 2000, the four were awarded the Beale-Orchard-Hays prize, given once every three years for the best published paper on computational mathematical programming. The abstract of Applegate, Bixby, Chvátal, and Cook’s award-winning paper, [1], gives a completely clear picture of the high regard current TSP researchers have for Dantzig, Fulkerson, and Johnson’s work:

Following the theoretical studies of J.B. Robinson and H.W. Kuhn in the late 1940s and early 1950s, G.B. Dantzig, R. Fulkerson, and S.M. Johnson demonstrated in 1954 that large instances of the TSP could be solved by linear programming. *Their approach remains the*

only known tool for solving TSP instances with more than several hundred cities; over the years it has evolved further through the work of M. Grötschel, S. Hong, M. Jünger, P. Miliotis, D. Naddef, M. Padberg, W.R. Pulleyblank, G. Reinelt, G. Rinaldi, and others. We enumerate some of the refinements that led to the solution of a 13,509 city instance. (emphasis added)

Dantzig, Fulkerson, and Johnson’s IP formulation

Dantzig, Fulkerson, and Johnson’s formulation has one decision variable for each pair of cities. For each $1 \leq i < j \leq n$,

$$x_{i,j} = \begin{cases} 1 & \text{if cities } i \text{ and } j \text{ are adjacent} \\ & \text{on the salesperson’s tour, and} \\ 0 & \text{if they are not.} \end{cases}$$

The objective is to minimize

$$\sum_{i=1}^{n-1} \sum_{j=i+1}^n c_{i,j} x_{i,j}, \tag{1}$$

the total distance traveled, subject to constraints that guarantee that the 0-1 values assigned to the $x_{i,j}$ ’s give rise to a legitimate tour. At first glance, it may appear that

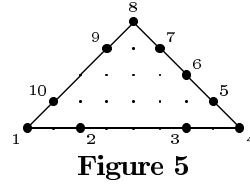
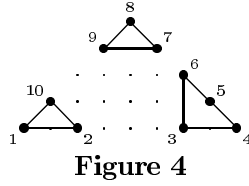
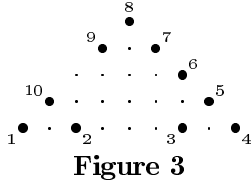
$$\sum_{i=1}^{j-1} x_{i,j} + \sum_{i=j+1}^n x_{j,i} = 2 \tag{2}$$

for each $1 \leq j \leq n$ are the only constraints needed. These “degree constraints” make sure that each city is adjacent to two others.

Unfortunately, many more constraints are needed to guarantee that the $x_{i,j}$ ’s describe a tour. To see why, consider Figures 3 and 4. Figure 3 displays the locations of 10 cities. The distances are Euclidean; $c_{1,2} = 2$, $c_{1,3} = 6$, \dots , $c_{9,10} = 2\sqrt{2}$. Figure 4 displays what happens when (1) is minimized subject to (2) for each $1 \leq j \leq 10$ and $x_{i,j} \in \{0,1\}$ for each $1 \leq i < j \leq 10$. (The optimal solution has $x_{1,2}^* = x_{1,10}^* = x_{2,10}^* = x_{3,4}^* = x_{3,6}^* = x_{4,5}^* = x_{5,6}^* = x_{7,8}^* = x_{7,9}^* = x_{8,9}^* = 1$. All other variables equal zero.) Note that the Figure 4 solution satisfies the degree constraints; each city is adjacent to two others. But clearly, no salesperson could actually adopt the Figure 4 solution; the Figure 4 “tour” contains three “subtours”.

Fortunately, subtours are easily broken. To break the subtour $S_1 = \{1, 2, 10\}$ (i.e. the subtour that connects cities 1, 2, and 10), one can add the constraint

$$x_{1,2} + x_{1,10} + x_{2,10} \leq 2.$$



This constraint states that no more than two of the three “edges” $\{1, 2\}$, $\{1, 10\}$, and $\{2, 10\}$ can be used. Similarly, to break the subtour $S_2 = \{7, 8, 9\}$, one can add the constraint

$$x_{7,8} + x_{7,9} + x_{8,9} \leq 2.$$

And to break the subtour $S_3 = \{3, 4, 5, 6\}$, one can impose the constraint

$$x_{3,4} + x_{3,5} + x_{3,6} + x_{4,5} + x_{4,6} + x_{5,6} \leq 3.$$

(If more than three of the edges $\{3, 4\}$, $\{3, 5\}$, $\{3, 6\}$, $\{4, 5\}$, $\{4, 6\}$, $\{5, 6\}$ are used, there must be a subtour connecting cities 3, 4, 5, and 6.)

To make sure that absolutely no subtours will appear, one could impose a constraint

$$\sum_{\substack{1 \leq i < j \leq n \\ i, j \in S}} x_{i,j} \leq |S| - 1 \tag{3}$$

for each $S \subseteq \{1, \dots, n\}$ such that $3 \leq |S| \leq n - 3$. The problem is that there are too many subtour elimination constraints—a total of $\binom{n}{3} + \binom{n}{4} + \dots + \binom{n}{n-3}$ of them—for this to be viable. When $n = 10$, there are almost one thousand subtour elimination constraints; when $n = 20$, there are over one million!

Dantzig, Fulkerson, and Johnson’s solution to the “subtour problem” was to add subtour elimination constraints as they are needed. Their procedure can be implemented as follows:

- Step 0:** Let $S = \emptyset$. (S is the set of encountered subtours).
- Step 1:** Minimize (1) subject to (2) for each $1 \leq j \leq n$, (3) for each $S \in \mathcal{S}$, and $x_{i,j} \in \{0, 1\}$ for each $1 \leq i < j \leq n$.
- Step 2:** Check the solution for subtours. If there are $k > 0$ subtours S_1, \dots, S_k , replace S with $S \cup \{S_1, \dots, S_k\}$ and return to Step 1. If there are none, stop; the solution gives rise to a legitimate, optimal tour. (When the solution is free of subtours, it satisfies *all* of the subtour elimination constraints, not just the ones that were included in S .)

Typically, only a very small fraction of the subtour elimination constraints are added.

When this procedure is applied to the TSP instance displayed in Figure 3, Step 1 is executed twice—the first time with $S = \emptyset$, and the second time with

$\mathcal{S} = \{\{1, 2, 10\}, \{3, 4, 5, 6\}, \{7, 8, 9\}\}$. Figure 5 displays the optimal solution to the second “Step 1” optimization problem. Note that in this small example, the optimal tour was obtained with the addition of only three subtour elimination constraints, although nearly one thousand possible subtours exist.

It should be noted that Dantzig, Fulkerson, and Johnson actually replaced the discrete constraints $x_{i,j} \in \{0, 1\}$ for each $1 \leq i < j \leq n$ with the continuous constraints $0 \leq x_{i,j} \leq 1$. Applegate, Bixby, Chvátal, and Cook (and other current TSP researchers) do this as well. The advantage of doing this is that each “Step 1” optimization problem becomes a linear programming problem, and linear programming problems are much, much easier to solve than integer programming problems. The drawback is that the solution to the “Step 1” optimization problem may satisfy all subtour elimination constraints yet still not correspond to a legitimate tour; some of the $x_{i,j}$ ’s may take on fractional values. To proceed, one can try to find some other constraint (not a subtour elimination constraint) that is satisfied by all tours but is not satisfied by the fractional solution. See [1], [4], and [5] for details.

The M⁷TP

In this section, we present Cardiff and Hughes’ IP formulation of the M⁷TP, the problem of constructing a “fun maximizing” itinerary for the Walt Disney World Magic Kingdom theme park.

Assumptions

To obtain a simple, tractable model, Cardiff and Hughes made the following assumptions:

1. In order to maximize the breadth of experience for first-time visitors, each attraction can be visited at most once.
2. The fun rating of an attraction is defined to be the number of stars given to that attraction in [9]. The goal of a park visitor is to maximize fun, the total number of stars gained on a tour.
3. The visitor’s length of stay at the park is one day. The Magic Kingdom normally operates between the hours of 9:00 AM and 7:00 PM.
4. Wait times are known and do not vary.
5. Food is both ubiquitous and uniformly good in the Magic Kingdom. In other words, both the quality and price of food are relatively constant throughout the Magic Kingdom. Because of this, meals will not alter the

optimal tour of the park. With this in mind, restaurants are not considered to add to total fun, although two hours will be allotted for eating.

Data

Let

- n = the number of rides in the park,
- p = the length of stay, less time for meals (in minutes),
- f_j = the fun factor of ride j (in stars),
- t_j = the ride time for ride j (in minutes),
- w_j = the wait time for ride j (in minutes), and
- $c_{i,j}$ = the distance between ride i and ride j (in minutes).

The park's rides are numbered from 1 to n ; the park's entrance is considered to be ride 0. As mentioned earlier, the fun factors and ride times were taken from [9]. Distances were measured using a map-measuring tool and a map downloaded from <http://www.waltdisneyworld.com>.

Unfortunately, neither [9] nor the official Walt Disney World site yielded any information about average wait times for each ride, and if any guidebook exists which contains information on average wait times for all rides, it is unknown to the authors. The wait times that were used were recorded by Cardiff and Hughes during their visit to the park.

Variables

There are two sets of variables. The variables in the first set are used to specify which rides are visited. For each $0 \leq j \leq n$,

$$y_j = \begin{cases} 1 & \text{if ride } j \text{ is visited,} \\ 0 & \text{if not.} \end{cases}$$

The variables in the second set are like the variables in the TSP; they are used to specify which rides are adjacent on the tour. For each $0 \leq i < j \leq n$,

$$x_{i,j} = \begin{cases} 1 & \text{if rides } i \text{ and } j \text{ are visited consecutively,} \\ 0 & \text{if not.} \end{cases}$$

Cardiff and Hughes' IP formulation

Cardiff and Hughes came up with the following IP formulation of the M⁷TP:

$$\max \quad \sum_{j=0}^n f_j y_j \quad (4)$$

$$\text{s.t.} \quad \sum_{i=0}^{j-1} x_{i,j} + \sum_{i=j+1}^n x_{j,i} = 2y_j \quad \text{for each } 0 \leq j \leq n, \quad (5)$$

$$\sum_{\substack{1 \leq i < j \leq n \\ i,j \in S}} x_{i,j} \leq |S| - 1 \quad \text{for each } S \subseteq \{1, 2, \dots, n\}, \quad (6)$$

$$\sum_{j=0}^n (t_j + w_j) y_j + \sum_{i=0}^{n-1} \sum_{j=i+1}^n c_{i,j} x_{i,j} \leq p, \quad (7)$$

$$y_0 = 1, \quad (8)$$

$$y_j \in \{0, 1\} \quad \text{for each } 1 \leq j \leq n, \quad (9)$$

$$x_{i,j} \in \{0, 1\} \quad \text{for each } 0 \leq i < j \leq n. \quad (10)$$

The objective function, (4), is the total amount of fun accumulated on the tour. The first two sets of constraints, (5) and (6), can be thought of as TSP constraints. They are analogous to (2) and (3), respectively. Each constraint in (5) ensures two things: first, that if a particular ride j is visited, then it is adjacent to two other rides, and second, that if ride j isn't visited, then it is adjacent to no other rides. Each constraint in (6) eliminates a subtour. Note that in the M⁷TP, no subtours containing ride 0 (the entrance to the park) are eliminated. Any subtour that contains the entrance to the park is a feasible, though not necessarily optimal, solution to the M⁷TP.

Constraint (7) guarantees that the tour takes no more than p minutes. The lefthand side of (7) is the sum of total ride time (waiting time plus riding time) and total travel time. Constraint (8) makes sure that the entrance to the park is part of the tour.

Solution strategy

Cardiff and Hughes' solution strategy was virtually identical to that of Dantzig, Fulkerson, and Johnson. They began by minimizing (4) subject to (5), (7), (8), (9), and (10). They then added subtour elimination constraints as needed.

Results

The optimal tours for preschoolers and their parents (for a ten-hour visit to the park, with two hours allotted for meals) are displayed in Figures 6 and 7. We

found it interesting that the optimal tours have the same basic shape. Indeed, the fact that both tours visit all five “lands” that comprise the Magic Kingdom is a testament to how well planned the entire park is. But if we look at the actual rides that both groups visit, we see that the two groups only agree on 12 rides. Surely, the fact that the routes are so similar, but the attractions visited are so different could be a source of parent/child dispute, because children (parents) would then be able to see all the rides that they wanted to go on, but would be dragged onto other ones by their parents (children).

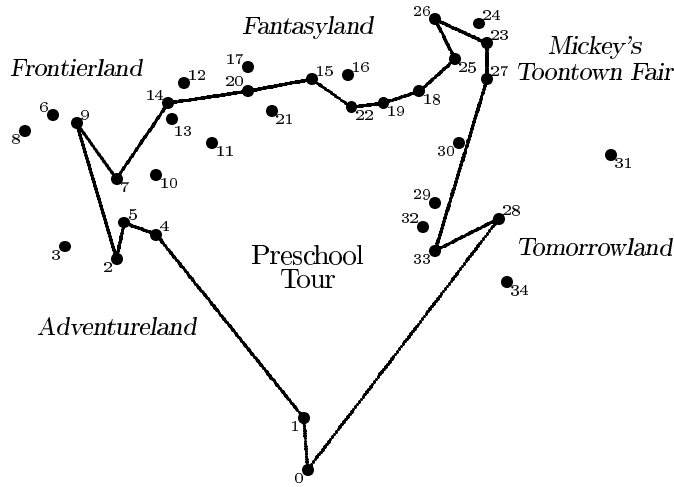


Figure 6

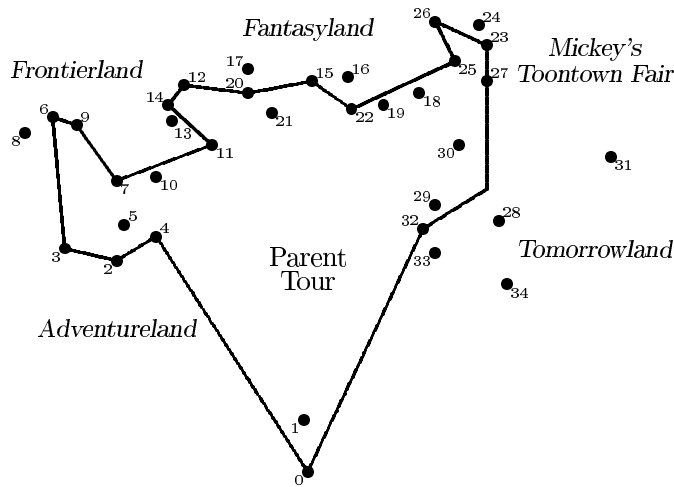


Figure 7

Table 1 contains some additional information about the optimal tours: the names of the rides, the fun values for preschoolers, and the fun values for parents.

Ride	Preschool		Parent		
	fun	on tour?	fun	on tour?	
0 Entrance	0.0	Yes	0.0	Yes	•
1 WDW Railroad	4.0	Yes	3.0	No	•
2 Jungle Cruise	3.5	Yes	3.0	Yes	•
3 Pirates of the Carribean	3.0	No	4.5	Yes	•
4 Swiss Family Treehouse	3.5	Yes	3.0	Yes	•
5 Enchanted Tiki Room	4.5	Yes	2.5	No	•
6 Thunder Mountain	3.0	No	4.0	Yes	
7 Country Bear Jamboree	3.5	Yes	3.0	Yes	•
8 Splash Mountain	0.0	No	5.0	No	
9 Raft to Tom Sawyer Island	5.0	Yes	3.0	Yes	•
10 Diamond Horseshoe Saloon Revue	2.0	No	3.5	No	
11 Hall of the Presidents	1.0	No	4.0	Yes	
12 Haunted Mansion	2.0	No	4.0	Yes	
13 Liberty Belle Riverboat	3.0	No	3.0	No	•
14 Mike Fink Keelboats	3.0	Yes	2.5	Yes	•
15 Cinderella's Golden Carousel	4.0	Yes	3.0	Yes	
16 Dumbo	5.0	No	1.5	No	
17 Small World	3.0	No	3.0	No	
18 Mad Tea Party	4.0	Yes	2.0	No	•
19 Winnie the Pooh	5.0	Yes	3.0	No	•
20 Peter Pan	3.5	Yes	3.5	Yes	•
21 Legend of the Lion King	3.0	No	3.0	No	•
22 Snow White's Scary Adventures	2.0	Yes	2.5	Yes	•
23 Donald's Boat	4.0	Yes	1.5	Yes	•
24 Mickey's Country House	3.0	No	2.5	No	
25 Minnie's Country House	3.0	Yes	2.5	Yes	•
26 Toontown Hall of Fame	4.0	Yes	3.0	Yes	•
27 Barnstormer	4.0	Yes	2.5	Yes	•
28 Astro-Orbiter	4.0	Yes	2.0	No	
29 ExtraTERRORestrial encounter	2.0	No	4.0	No	
30 Tomorrowland Speedway	3.5	No	0.5	No	
31 Space Mountain	0.0	No	4.0	No	
32 Timekeeper	2.5	No	4.0	Yes	
33 Buzz Lightyear	3.0	Yes	2.0	No	
34 Carousel of Progress	2.0	No	3.0	No	
		Total = 67.5		Total = 53.5	

Table 1

It turns out that if a group of visitors follows the optimal “preschool” tour, its preschool members will accumulate 67.5 stars and their parents will accumulate 47.5 stars. If it follows the optimal “parent” tour, its children will obtain 54.5 stars and their parents will obtain 53.5 stars. The •’s in Table 1 identify the rides that form a “compromise” tour (which was found by maximizing “parent fun” while keeping “kid fun” above 65 stars). If the visitors follow the compromise

tour, the preschoolers will obtain 65.5 stars (97.0% of their optimal value) and their parents will obtain 51 stars (95.3% of their optimal value).

Finally, it should be noted that we used the optimization package CPLEX (version 6.6) to solve our integer programming problems. In our search for the optimal preschool tour, we added a total of seven subtour elimination constraints. While hunting for the optimal parent tour, we added ten. Each “Step 1” optimization problem required less than one second of CPU time on an 800 MHz Pentium III computer.

Ideas for future research

We would like to create a new, time-based model in which ride waiting time depends on time of day. Ideally, this model would allow for the inclusion of meals, shows (which take place only at specific times during the day), and the “fast pass” (which gives a visitor access to a shorter queue for a popular ride, but only during a certain hour-long time window).

References

- [1] D. Applegate, R. Bixby, V. Chvátal, and W. Cook, “On the solution of traveling salesman problems”, *Documenta Mathematica* Extra Volume III (1998), 645-656.
- [2] G.B. Dantzig, D.R. Fulkerson, and S.M. Johnson, “Solution of a large-scale travelling salesman problem”, *Operations Research* **2** (1954) 393-410.
- [3] B.L. Golden and W.R. Stewart, “Empirical analysis of heuristics,” in E.L. Lawler, J.K. Lenstra, A.H.G. Rinnooy Kan, and D.B. Shmoys, eds., *The Traveling Salesman Problem* (Wiley, New York, 1985) pp. 207-249.
- [4] M. Grötschel and M.W. Padberg, “Polyhedral theory,” in: E.L. Lawler, J.K. Lenstra, A.H.G. Rinnooy Kan, and D.B. Shmoys, eds., *The Traveling Salesman Problem* (Wiley, New York, 1985) pp. 251-305.
- [5] M. Grötschel and M.W. Padberg, “Polyhedral computations,” in: E.L. Lawler, J.K. Lenstra, A.H.G. Rinnooy Kan, and D.B. Shmoys, eds., *The Traveling Salesman Problem* (Wiley, New York, 1985) pp. 307-360.
- [6] D.S. Johnson and C.H. Papadimitriou, “Performance guarantees for heuristics,” in: E.L. Lawler, J.K. Lenstra, A.H.G. Rinnooy Kan, and D.B. Shmoys, eds., *The Traveling Salesman Problem* (Wiley, New York, 1985) pp. 145-180.

- [7] R.M. Karp and J.M. Steele, "Probabilistic analysis of heuristics," in E.L. Lawler, J.K. Lenstra, A.H.G. Rinnooy Kan, and D.B. Shmoys, eds., *The Traveling Salesman Problem* (Wiley, New York, 1985) pp. 181-205.
- [8] G. Laporte and S. Martello, "The selective travelling salesman problem", *Discrete Applied Mathematics* **26** (1990) 193-207.
- [9] B. Sehlinger, *Mini Mickey: The Pocket-Sized Unofficial Guide to Walt Disney World 1999* (Macmillan, New York, 1999).

Appendix

This year, the Mathematical Society of America has decided to hold its annual meeting in sunny Orlando, Florida. As a professional mathematician, you will be attending the meeting, and because of fond childhood memories of Disney World, you find yourself trying to schedule a quick three-hour visit to the park. Since your time in the park is limited, you decide to use integer programming to plan your itinerary. To minimize the time you spend traveling from ride to ride, you decide to limit yourself to the rides that were your childhood favorites: Pirates of the Carribean (ride 3), Thunder Mountain (ride 6), and the rides of Tomorrowland (rides 28 through 34).

Solving the problem with Excel

1. Make an Excel spreadsheet look like Figure 8.

	A	B	C	D	E	F	G	H	I	J	K	L	M	
1				Ride-to-ride distances										
2		3	6	28	29	30	31	32	33	34		Fun	Time	
3	0	9.5	11	9	7.5	9	10.5	8	9	9		0	0	
4	3		2.5	9	7.5	9	10.5	7.5	8.5	9.5		4.5	27.5	
5	6			10	8.5	9.5	11.5	8	9	10		4	38.5	
6	28				1.5	1.5	2	1.5	1	1		2	31.5	
7	29					1.5	3	0.5	1	1.5		4	47	
8	30						3	2	2.5	2.5		0.5	44.5	
9	31							3.5	3	2.5		4	53	
10	32								1.5	2		4	35	
11	33									1		2	21	
12	34											3	28	
13				Are two rides adjacent?										
14		3	6	28	29	30	31	32	33	34		Do?	Deg.	
15	0	0	0	0	0	0	0	0	0	0		1		
16	3		0	0	0	0	0	0	0	0		0		
17	6			0	0	0	0	0	0	0		0		
18	28				0	0	0	0	0	0		0		
19	29					0	0	0	0	0		0		
20	30						0	0	0	0		0		
21	31							0	0	0		0		
22	32								0	0		0		
23	33									0		0		
24	34											0		
25		Total ride time							Subtour			LHS	RHS	
26		Travel time												
27		Total time												
28		Time in park				180								
29		Total fun												

Figure 8

The top half of the spreadsheet contains the data: ride-to-ride distances (in minutes), fun values (in stars), and ride times (waiting plus on-ride, in minutes). The bottom half contains the decision variables. Cells B15 through J23 are the $x_{i,j}$'s, and cells L15 through L24 are the y_j 's. (For example, cell F18 equals 1 if rides 28 and 30 are adjacent on the tour and 0 otherwise. Cell L18 equals 1 if ride 28 is visited and 0 if it isn't.)

2. Next, you must express the auxiliary variables—cells F25, F26, F27, and F29 and cells M15 through M24—in terms of the decision variables. Cell F25 equals total ride time, the total amount of time you spending waiting for rides and riding on them (i.e. the first term of the lefthand side of constraint (7)). To express cell F25 in terms of the decision variables, click on cell F25, type

$$= \text{SUMPRODUCT}(M3:M12,L15:L24)$$

and press *Enter*. (Excel's SUMPRODUCT formula is for computing the dot product of two vectors—in this case, the vector comprised of cells M3 through M12, and the vector comprised of cells L15 through L24.)

Cell F26 equals travel time (i.e. the second term of the lefthand side of constraint (7)). To express cell F26 in terms of the decision variables, click on it, type

$$= \text{SUMPRODUCT}(B3:J11,B15:J23)$$

and press *Enter*.

Cell F27 is the total time in the park, the sum of total ride time and travel time. To define it, type

$$= \text{SUM}(F25:F26)$$

in cell F27 and press *Enter*.

Cell F29 equals total fun, the number of stars you accumulate during your visit (i.e. the objective function (4)). To express it in terms of the decision variables, type

$$= \text{SUMPRODUCT}(L3:L12,L15:L24)$$

in cell F29 and press *Enter*.

The remaining auxiliary variables—cells M15 through M24—are the lefthand sides of the degree constraints (5). To express M18, the “degree” of ride 28, in terms of the decision variables, click on it and type `SUM(`. Then hold down the *Ctrl* key (if you are using a PC) or the apple key (if you are using a Macintosh), and use your mouse to select cells D15 through D17 and E18 through J18 (the cells that correspond to the “edges” that touch ride 28). Note that cells stay selected even when you let go of your mouse button and select more cells. Finally, type `)` and press *Enter*. Cell M18 now contains the formula `SUM(D15:D17,E18:J18)`. Cells M15 through M17 and M19 through M24 are treated similarly.

3. The next step is to make Excel aware that you would like to solve an optimization problem. To do this, choose the *Solver* from the *Tools* menu.
4. To specify the objective function, click once in the *Set Target Cell* field and type `F29` (or use your mouse to select cell F29). Then make sure that *Max* is selected in the *Equal To* field.
5. To specify the decision variables, click once in the *By Changing Cells* field. Then, hold down the *Ctrl* key (if you are using a PC) or the apple key (if you are on a Macintosh), and use your mouse to select all cells that correspond to decision variables. **Note:** Do not select cell L15. It should remain at its current value of 1; the entrance to the park *must* be visited.
6. Now it is time to add the constraints. To add a constraint, click on the *Add* button. A window that resembles Figure 9 will appear.

Cell Reference:		<div style="border: 1px solid black; display: inline-block; padding: 2px;"><=</div> <div style="border: 1px solid black; display: inline-block; padding: 2px; margin-left: 5px;">▼</div>	Constraint:	
-----------------	--	---	-------------	--

Figure 9

To add the degree constraint for ride 0, click once in the *Cell Reference* field and type `M15` (or use your mouse to select cell M15). Then, change the “<=” to an “=”. Finally, click once in the *Constraint* field, type `2*L15`, and click on the *OK* button. To add the degree constraints for the other rides, repeat the process, replacing cells L15 and M15 first with L16 and M16, then with L17 and M18, and so on.

To add the time constraint (7), type `F27` in the the *Cell Reference* field and `F28` in the *Constraint* field.

Finally, make Excel aware that the variables are binary (i.e. that constraints (9) and (10) must hold). Unfortunately, this must be done in stages. First, add a constraint that states that cells B15 through J15 are binary. (Click once in the *Cell Reference* field, use your mouse to select cells B15 through J15, and change the “<=” to “bin”.) Next, add constraints that state that cells C16 through J16 are binary, that cells D17 through J17 are binary, and so on. Don’t forget to add a constraint that says that cells L16 through L24 are binary.

7. After all of the constraints have been added, click on the *Options* button, check the *Assume Linear Model* box, and click on the *OK* button. Then click on the *Solve* button. Excel’s Solver will solve the integer programming problem and place the optimal values of the variables in the corresponding cells. To keep the solution, click on the *OK* button.
8. Now, examine the optimal solution. You should find that the optimal “tour” accumulates 17.5 stars but has two subtours: 0-3-6-0 and 32-33-34-32. To eliminate the second one (the first one contains the entrance and therefore

should not be eliminated), type

$$= \text{SUM}(I22, J22, J23)$$

in cell L26 and the number 2 in cell M26, open the *Solver* tool, and add a constraint that states that cell L26 should be less than or equal to cell M26.

9. Solve the revised integer programming problem. Examine the optimal solution. You should find that it is free of subtours and that the optimal tour, 0-6-3-29-32-0, accumulates 16.5 stars.

Web page

To obtain a copy of the Excel file (and a copy of the C program we used to generate the CPLEX “.lp” files for the preschool and parent M⁷TP instances), point your browser to

<http://www.oberlin.edu/~math/people/faculty/bosch/m7tp-page.html>