Integer Linear Programming Formulation for Multi-day Single-city Trip Plan

Moonyoung Kang, Cambridge, MA

Building a realistic multi-day trip plan covering multiple tourist attractions is not trivial because of complex open hours which needs to be satisfied, such as early close days, i.e. close at 12PM on Fridays, and annual close dates, i.e. closed on Thanksgiving Day. On top of this, building a time and money saving plan is more difficult and the problem can get worse with tourist attractions that provides time dependent discounts, such as free entry on Friday. In this paper, we present a multi-day trip planning algorithm which not only preserves various open hours conditions, but also is thrifty and exploits time dependent discounts. We tested our system with various tourist attractions in New York City, one of the largest tourist destinations in the world. Upon our knowledge, this is the first paper that builds a multi-day trip plan with consideration of open hours information and cost optimization.

Introduction

Imagine you are visiting New York City (NYC) for three days and decided to visit three tourist attractions: Metropolitan Museum of Art (MET), Museum of Modern Art (MoMA), and Guggenheim Museum (GM). You visited MET on Sunday and GM on Monday. However on Tuesday, on the last day of your trip, you find that MoMA is closed on Tuesdays. Above situation can be avoided if tourist attractions' open hours information is properly considered when making the trip plan. However, open hours vary greatly between tourist attractions, for instance, MET is closed on Mondays, MoMA is closed on Tuesdays, and GM is closed on Thursdays, and it is not trivial to build a feasible trip plan that satisfies open hours of multiple tourist attractions.

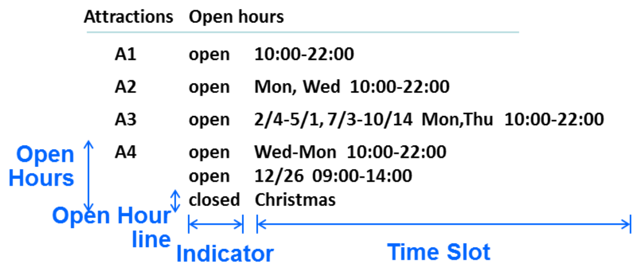
Making the problem worse, some tourist attractions provide “free entry” days, for instance, MoMA is free on Fridays after 4PM and GM is free on Saturdays after 5:45 PM, and travellers often prefer to visit tourist attractions on free entry dates. In this paper, we propose a multi-day tour planning algorithm that can satisfy open hours of the selected tourist attractions and at the same time exploits date specific discounts and services.

There has been works on tour planning systems previously. For instance, Hochmeir et al. (2005) proposed tour planning system for bicycle riders and Kurata (2011) proposed an interactive tour scheduling system, which builds a trip plan following the traveller’s preference. However, upon our knowledge, there has been no trip planning algorithm that considers tourist attractions’ open hours information and this is the first paper that can build trip plans which not only considers open hours information but also tries to optimize on trip cost.

The rest of the paper is structured as follows. Open hours processing algorithm section describes how the system normalizes open hours and processes it. Ticket selection section introduces ticket optimization algorithm. Trip planning section describes in detail how we build an integer linear programming formulation for tour optimization which operates based on the result of open hours and ticket selection information. These sections will be followed by conclusion and references.

Open hours processing algorithm

Fig. 1. Examples of open hours



Attractions	Open hours	
A1	open	10:00-22:00
A2	open	Mon, Wed 10:00-22:00
A3	open	2/4-5/1, 7/3-10/14 Mon,Thu 10:00-22:00
A4	open	Wed-Mon 10:00-22:00
	open	12/26 09:00-14:00
	closed	Christmas

In order to consider open hours data during trip planning, we normalized open hours into machine-readable form. Fig. 1 shows four examples of normalized open hours. Open hours consists of multiple lines. Each open hour line consists of two parts: indicator and time slot. Indicator determines whether the tourist attraction is open or closed during the time defined in the time slot and time slot defines when the open hour line is valid. Time slot consists of three subparts: dates, days, and hours. In case of A3 in Fig. 1, “2/4-5/1, 7/3-10/14” is the dates subpart, “Mon, Thu” is the days subpart, and “10:00-22:00” is the hours subpart. Each subpart is expected to repeat over time; dates subpart defined “2/4-5/1, 7/3-10/14” is expected to be valid between Feb. 4th – May 1st and Jul. 3rd – Oct. 14th of every year, and similarly, each days and hours repeat over weeks and days. Only one or two among three subparts of the time slot may be defined, and the more subpart is defined, the stricter the time slot becomes. For instance, tourist attraction A1 in Fig. 1 opens every day between 10AM and

10PM while tourist attraction A2 opens between 10AM and 10PM only on Mondays and Wednesdays. Open hour lines are sorted in the order of more general to specific, from top to bottom, following the convention.

Table 1. Theorems from open hours

Theorem 1. Open hour lines are ordered from more general cases to more specific cases, from top to bottom. The lower line always dominates upper line.

Theorem 2. For any matching date, a single open hour line affects the entire date, not only during the hours defined.

Theorem 3. Closed open hour line has no hours subpart.

Table 1 shows theorems that can be inferred from normalized open hours. Theorem 1 describes that lower lines always dominates upper lines. From Fig. 1, you can infer that attraction A4 should be closed on Wed, Dec. 25th, 2013, because it is Christmas, even though first open hour line describes A4 to be open on Wednesdays. Theorem 2 describes that for any date that satisfies open hour line's date subpart, the open hour line affects the whole date, not only the time defined in the hours subpart. For instance, on Thursday, Dec. 26th, 2013, A4 should be open until 10PM from the definition of first open hour line. However, because each open hour line affects the whole day of matching dates, A4 will close at 2PM on Dec. 26th based on the second open hour line. Theorem 3 is corollary of Theorem 2; if each open hour line affects the whole date, any "close" open hour line will make the tourist attraction not visitable for the given whole date regardless of the hours subpart.

Table 2. Availability calculation algorithm for tourist attractions

```
procedure avail_hours (attraction, trip_hours)
  avail_hours = trip_hours
  foreach openhour in reverse(attraction.openhours)
    overlap = intersect (avail_hours, openhour)
    if overlap == NULL:
      continue

    avail_hours -= overlap.days
    if openhour.indicator == "open"
      foreach day in overlap:
        if overlap >= attratom.expected_duration:
          emit overlap

    if avail_hours == NULL:
      break
```

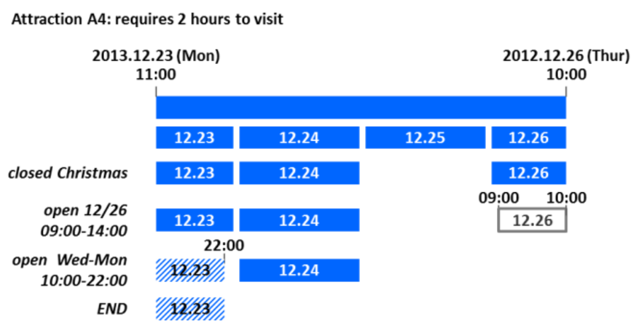


Fig. 2. Example run of algorithm in Table 2

Table 2 describes the availability calculating algorithm in python-like pseudo-code which determines whether each tourist attraction is visitable during a trip and Fig. 2 shows an example run of the algorithm using attraction A4's open hours in Fig. 1 with the assumption that 2 hours are required to visit A4.

The algorithm first scans open hour lines in reverse order which enables the algorithm to succeed or fail fast. If overlap is not found between open hour line and trip schedule, the algorithm go on to next open hour line and continues. If overlap is found, the algorithm first removes overlapped days from trip's schedule because these days should not be reconsidered in the following procedure. For each matching date, if the open hour line's indicator is "closed", the algorithm does nothing and continues to the next open hour line. In Fig. 2, you can see that "closed Christmas" open hour line removes Dec. 25th from possible visit dates.

If the indicator is "open", the algorithm checks whether the overlapping time is longer than the required visit hours. If not, the algorithm does nothing and continues to the next open hour line. In Fig. 2, "open 12/26 09:00-14:00" open hour line has only 1 hour overlap with available time on Dec. 26th, which is shorter than two hours, the required visit hours of attraction A4. Therefore, Dec. 26th is removed from the possible date of visit. If the overlapping time is longer than the required visit hours, the algorithm emits the overlap as the new possible visit hours. In Fig. 2, the algorithm emits Dec. 23rd as the new possible visit time from open hour line "open Wed-Mon 10:00-22:00". In the end, the algorithm quits if there is no more day or open hour line left to process.

The set of visitable items will later be intersected with related attractions' open hours.

Ticket Selection

As stated in Introduction section, some tourist attractions often provide discount tickets on certain dates. Since trip schedule should not prevent travelers from buying discount tickets, ticket selection must happen before trip planning.

In this paper, we used ticket selection algorithm of Kang (2012). The algorithm finds the cheapest ticket combination over multiple tourist attractions considering trip schedule, properties, such as age, of travellers, and existence of time-dependent tickets, such as free Friday admission of MoMA. The algorithm returns a set of tickets that minimize user cost. If returned set of tickets contains any time-dependent ticket, the attraction's open hours will be intersected with the time condition of the ticket.

Trip Planning

Multi-Day Trip-Planning Problem (MDTPP) can be modelled as a Generalized Vehicle Routing Problem (GVRP) (Kara, I., and Bektas, T., 2003), which is a general version of Traveling Salesman Problem (Lawler et al., 1976). GVRP solves the problem of delivering packages to customers and a delivery company, such as Fedex, tries to deliver all the packages in the depot to customers of various locations using multiple trucks. Goal of the delivery company is to minimize the distance travelled by the trucks while delivering all the packages to the customers. No customer should be visited more than once and trucks cannot have more packages than its load permits.

In comparison, MDTPP tries to minimize distance travelled while visiting all the selected tourist attractions given certain amount of days. MDTPP can be converted to GVRP easily; an MDTPP over multiple tourist attractions with time constraints can be converted into a GVRP over multiple customers with constraints on load of packages. The following sections will describe MDTPP's integer linear programming (ILP) formulation in more detail.

Formal Definition of Problem

MDTPP tries to build a trip plan that visits all the tourist attractions, which fits into given trip schedule, minimize the time spent by the traveller on the road, and at the same time, match the traveller's preference, such as how compact the traveller wants the trip. External parameters, such as tourist attractions to visit, trip schedule with start and end time, hotels the traveller plans to stay on each day of the trip, and list of nearby restaurants will be provided to MDTPP.

Table 3. Definition of symbols of Multi-Day Trip-Planning Problem

A : Set of tourist attractions the traveller plans to visit
H : Hotels the traveller plans to stay on each day of trip
L : Set of possible restraunts
Station : Union of **A**, **H**, and **L**
i, j : Station index
k : Day index

Table 3 defines symbols of MDTPP. A is the set of tourist attractions the traveller plans to visit during this trip. The number of attractions is properly managed to fit the traveller's trip schedule in the pre-processing step. Each tourist attraction has information of open hours and hours required to visit the tourist attraction. H is the set of hotels and contains the information of which hotel the traveller plans to stay on each date. If there is no hotel information provided, the system creates an imaginary hotel which is located 10 minutes away from any tourist attraction. L is the list of possible places that traveller may have lunch. For now, we assume that all restaurants are of equal quality and traveller goes to the closest restaurant at lunch time. In ILP formulation, letter i and j will be used as the index for stations and k as index for each date of trip.

Decision Variables

Table 4. Decision Variables

$\mathbf{x_{ijk}}$ [boolean] : Whether move from Station i to j on Day k
$\mathbf{u_{ik}}$ [float] : End of visit hour at Station i on Day k.
$\mathbf{r_{ijk}}$ [float] : Rest hours between Station i and j on Day k
$\mathbf{s_k}$ [float] : Start hour of trip on Day k

Table 4 describes decision variables of MDTPP. The goal of ILP is to determine decision variables properly to satisfy all the conditions and at the same time minimize optimization function. x_{ijk}, a boolean variable, represents whether the traveller will travel from station i to station j on day k, which can be used to reconstruct the traveller's trip path for each day. u_{ik}, a float variable, represents the time when the traveller exits station i, on day k. u_{ik} is used to regulate visit to each station to be within the station's open hours. r_{ijk} represents the rest the traveller takes while traveling from station i to station j on day k. r_{ijk} can be used to match the traveller's preference; for a traveller who prefers to travel fast, we may give penalty for large r_{ijk} value, and for travellers who prefers to travel slow, we may give penalty for r_{ijk} smaller than certain value. s_k represents the start hour of the trip on day k, which allows late start of a trip when the first attraction of the day is planned at late afternoon.

Parameters

Table 5. Input parameters

$\mathbf{H_k}$: Hotel of Day k
$\mathbf{q_i}$: Station OR cost of visiting Station i
$\mathbf{d_{ij}}$: Cost of traveling between Station i and j
$\mathbf{QS_k}$, $\mathbf{QE_k}$: Trip Start/End hour of Day k
$\mathbf{S_k}$: Set of tourist attractions visitable on Day k
$\mathbf{OPEN_{ik}}$, $\mathbf{CLOSE_{ik}}$: OPEN/CLOSE time of Attraction i on Day k

Table 5 describes input parameters to MDTPP ILP whose values are provided at the time of solving ILP. H_k is the hotel the traveller plans to stay on day k. If traveller stays in multiple hotels, ILP will prefer to visit tourist attractions near the hotel of the day to minimize the time on the road. Each of q_i and d_{ij} represents the time cost of visiting tourist attraction i and traveling between attraction i and j. QS_k and QEs are the trip start and end time on date k, which is from user setup. $OPEN_{ik}$ and $CLOSE_{ik}$ are open and close hours of tourist attraction i on day k. Open hours processing result and time dependent ticket information are intersected together to define $OPEN_{ik}$ and $CLOSE_{ik}$ parameters.

Equations

Table 6. Bounds of variables equations

$$\forall q_i \in A, k$$
$$\max(\mathrm{OPEN}_{ik} + q_i, QS_k) \leqq u_{ik} \leqq min(CLOSE_{ik}, QE_k) \ [1]$$
$$\forall q_i, q_j \notin H, \forall k, \quad r_{ijk} \geqq 0 \ [2]$$
$$\forall q_i \in H, q_j \notin H, k, \quad s_{jk} \geqq max(QS_k, 0) \ [3]$$

Equations are conditions that ILP must preserve by properly setting decision variables and the following sections describe optimization and condition equations of ILP formulation. Table 6 shows decision variables bounding equations. Equation 1 bounds attraction visit time variable u_{ik}, to be after the trip start time (QS_k) and attraction's open hour ($OPEN_{ik}$) and before the trip end time (QE_k) and attraction's close hour ($CLOSE_{ik}$). Equation 2 bounds resting variable r_{ijk} to be non-negative. Equation 3 bounds trip start time variables to be non-negative and be after the trip start time parameter (QS_k).

Table 7. Station visit control equations

$$\forall q_i \in A, \quad \sum_{j,k} x_{ijk} = 1 \text{ and } \sum_{j,k} x_{jik} = 1 \ [4]$$
$$\forall k \ \exists q_0 = H_k, \quad \sum_{j \neq 0} x_{0jk} \leqq 1 \text{ and } \sum_{j \neq 0} x_{j0k} \leqq 1 \ [5]$$
$$\forall q_i \in L, k, \text{ if plan to have lunch on Day k}$$
$$\sum_{i,j \neq i} x_{ijk} = 1 \text{ and } \sum_{i,j \neq i} x_{jik} = 1 \ [6]$$
$$\forall i \neq 0, k, \quad \sum_{j \neq i} x_{ijk} - \sum_{j \neq i} x_{jik} = 0 \ [7]$$

Equation 4, 5, 6, and 7 regulates the number of visits to each station. Equation 4 conditions each attraction to be visited once and only once throughout the whole trip. Equation 5 shows that a traveller may leave the hotel for trip (x_{0jk}=1, x_{j0k}=1) or stay in hotel for the day (x_{0jk}=0, x_{j0k}=0) in case of a trip with a few tourist attractions. Equation 6 ensures that the traveller visits one of the listed restaurants on each day the traveller needs to buy lunch and equation 7 ensures that once the traveller enters a station, the traveller also leaves the station.

Table 8. Visit time control equations

$$\forall i \in A, \exists q_0 = H_k, \quad u_{ik} + d_{i0}x_{i0} \leqq QE_k \ [8]$$
$$\forall i \in A, \exists q_0 = H_k \quad u_{ik} + (-q_i - d_{0i} - s_k)x_{0ik} \geqq 0 \quad [9]$$
$$\forall q_i, q_j \notin H, i \neq j, \forall k,$$
$$u_{ik} - u_{jk} + (QE_k + d_{ij} + q_j + r_{ijk})x_{ijk} \leqq QE_k \ [10]$$
$$u_{jk} - u_{ik} + (QE_k - d_{ij} - q_j - r_{ijk})x_{ijk} \leqq QE_k \ [11]$$

Time required to visit each attraction is accumulated to u_{ik} through equations in Table 8. Equations regulate the total visit time to be less than trip end hour (QE_k). Station visit control equations in Table 7 and visit time control equations in Table 8 together guarantees there to be at most 1 trip per day and the trip to be a simple cycle starting from and ending at the hotel.

Table 9. Optimization equation

$$z = min\{\sum_{i,j,k}(d_{ij} + q_i)x_{ijk} + \sum_{i,j,k} r_{ijk}\} \quad [12]$$

Equation 12 in Table 9 is the optimization equation. The algorithm tries to find the values of x_{ijk} and r_{ijk} that minimizes the optimization equation while satisfying all the inequalities. As mentioned in decision variables section, x_{ijk} values are used to generate list of visiting attractions on each date. Combined with u_{ik} values and ticket selection results, the system can generate traveller's complete trip plan including the information of recommended visit time of each tourist attraction, time and place of lunch, time dependent tickets to buy, proper departure time in the morning, and etc.

ILP can easily be extended to add functionalities. For instance, s_{jk} can be added to optimization function to promote early or late departure from hotel, or optimization equation of r_{ijk} can be modified to match traveller's preference between compact and loose trip. User preference to visit certain station on specific time can be satisfied by setting $OPEN_{ik}$ and $CLOSE_{ik}$ properly.

Experiment

We tested our system on 90 tourist attractions in New York City to see whether we can get results in feasible amount of time. We manually input information of each tourist attractions, such as open hours, expected visit duration, and location of each tourist attraction. For experiment, a laptop machine of 1.30GHz CPU, 2GB memory, 150GB HDD with Ubuntu LTS 12.4 is used.

In most cases, system could generate trip plan in less than 1 minute, and even with a fairly complex trip plan with more than 30 tourist attractions, system generated trip plans in less than 5 minutes.

Conclusion

Our system could build a trip plan over multiple tourist attractions that minimize traveller's time and cost while preserving open hours conditions of each tourist attraction by converting trip planning problem into integer linear programming problem. Current algorithm can easily be extended to match various preferences of travellers. The system could generate a trip plan covering 30 tourist attractions in feasible amount of time. In the future, we plan to extend our system to build a trip plan over multiple cities.

References

Hochmair, H., and Rinner, C. (2005). Investigating the need for eliminatory constraints in the user interface of bicycle route planners. *Proceedings from the COSIT '05, Ellicottville, NY, USA.*

Kang, M. (2013). Integer Programming Formulation of Finding Cheapest Ticket Combination over Multiple Tourist Attractions. *Information and Communication Technologies in Tourism 2013 (Proceedings of the International Conference in Innsbruck, Austria, January 23-25, 2013).* Berlin - Heidelberg: Springer

Kara, I., and Bektas, T. (2003). Integer Linear Programming Formulation of the Generalized Vehicle Routing Problem. *EURO/INFORMS Joint International Meeting, Istanbul, July 06-10, Turkiye.*

Kurata, Y. (2011) CT-Planner2: More Flexible and Interactive Assistance for Day Tour Planning. *ENTER 2011, Information and Communication Technologies in Tourism 2011, 25-37, Innsbruck, Austria, Jan, 2011.*

Lawler EL, Lenstra JK, Rinnooy~Kan AHG, Shmoys DB (eds.) (1985). The Traveling Sales-man Problem. *Wiley, New York.*