אוניברסיטת בר-אילן
Bar-Ilan University

Route Optimization

Optimization

Advisors:

Prof. Gil Ariel

By:

Shon Otmazgin 305394975

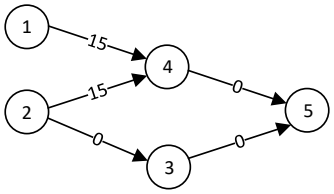
Sapir Rubin 301659751

GitHub Repository

אוניברסיטת בר-אילן
Bar-Ilan University

Abstract

This paper discusses a highly effective heuristic procedure for generating optimum and near-optimum solutions for a deadhead route problem with limit route duration which is a variety of the multi traveling salesman problem (MTSP). In practice, temporal aspect is a necessary constraint with respect to the routing problems. MTSP is a NP-hard problem, and computational time of exact algorithm increases exponentially as the number of trips increases. Hence, we present a beam search (BS) algorithm which is a heuristic based on breadth-first branch-and-bound without backtracking to solve the Deadhead route problem. BS filters out worse nodes by a local evaluation and only keep β (called beam width) nodes according to global evaluation at each level. In our BS, both one-step local evaluation and global evaluation are applied to estimate score of nodes by inserting unvisited nodes of a given initial solution. The computational results on test instances from the literature show that our beam search can obtain good solutions with effective computational times for Deadhead route problem with time windows.



Introduction

1. Glossary

Definition	Description
Stop id	Integer between 1 and 12. Where 0 is the depot stop id
Time	Minutes from midnight
Service trip	Consists of • id <unique int> • Origin <stop id> • Destination<stop id> • Departure<time> • Arrival <time>
Deadhead	• Trip duration between any two different stops – 15 minutes • Trip duration between a stop to itself – 0 minutes
Valid Vehicle	A list of possibly temporarily non decreasing and possibly geographically connected service trip ids. The vehicle must start and end at the depot.
Deadhead duration	Sum of all deadheads needed for vehicle temporal and geographic legality
Vehicle driving time	Sum of all trips (service and deadhead) durations
Restricted legal vehicle	Vehicle that its driving time is less than or equal to 480 minutes
Valid daily Schedule	List of restricted legal vehicles
Covered service trip	A service trip executed by exactly one Restricted legal vehicle in a Valid daily schedule
Daily schedule OpEx	Sum of all vehicle deadhead durations

2. Dataset

The dataset contains trips, with the following format as example:

Trip id	origin	destination	departure	arrival
1	2	9	305	365
27	1	1	720	840
28	1	1	840	900

3. Problem statement

Given a list of service trips (a trip between different stops during the day a vehicle must execute), we would like to find the most efficient valid daily schedule of vehicles, that must start and end their day in a single depot stop and that are restricted by 8 hours driving time and covered each trip exactly once.

$f_x(v)$ is the total deadhead duration of vehicle $v \in \mathbb{R}^n$. with respect to parameter vector x.

$$f_x(v) = 30 + \sum_{t=o}^{|v|-2} w_{v_t, v_{t+1}}$$

$$w_{i,j} = \begin{cases} 0, & if\ destination(i) = origin(j) \\ 15, & otherwise \end{cases}$$

$r(v)$ is the total driving time of vehicle $v \in \mathbb{R}^n$. with respect to parameter vector x.

$$r(v) = f_x(v) + \sum_{i=0}^{|v|-1} arrival(v_i) - departure(v_i)$$

we typically aim to minimize the daily schedule OpEx:

Given R Valid daily Schedule:

$$\phi = \sum_{k=1}^{|R|} f_x(R_k)$$

$$s.t.$$

$$r(R_k) \leq 480$$

אוניברסיטת בר-אילן
Bar-Ilan University

Beam Search for Deadhead Route Problem

Data representation

Definitions

1. N – set of all service trip IDs
2. C – set of all covered service trip IDs.

3. Pairwise Matrix - $S \in M_{|N|\times|N|}$ where

$$s_{i,j} = \begin{cases} True, & arrival(i) + w_{i,j} \leq departure(j) \\ False, & otherwise \end{cases}$$

4. Penalty Matrix - $P \in M_{|N|\times|N|}$ where

$$p_{i,j} = \begin{cases} w_{i,j}, & s_{i,j} = True \\ NaN, & otherwise \end{cases}$$

Note – Sorting trip IDs by departure leads S and P to be upper triangular matrices.

5. All possible dead ends:

$$D = \{x \in N \backslash C \mid \forall_{j \in N\backslash C}\ S_{x,j} = False\}$$

6. L – set of l latest trips departure

Given vehicle v

7. Set of candidates for v:

$$T_v = \{\, t \mid S_{t,v_0} = True\}$$

8. Set of restricted candidates for v:

$$\Omega_v = \{\, t \in T_v \mid r(t+v) \leq 480\}$$

Example

Recall dataset example with trips IDs 1,27,28 as indices, getting the below Matrices:

$$N = \{1,27,28\};\ C = \emptyset;\ D = \{28\};\ l = 2 \Rightarrow L = \{27,28\}$$

$$S = \begin{pmatrix} False & True & True \\ False & False & True \\ False & False & False \end{pmatrix} \quad P = \begin{pmatrix} NaN & 15 & 15 \\ NaN & NaN & 0 \\ NaN & NaN & NaN \end{pmatrix}$$

$$\Omega_{[28]} = \{1,27\};\ \Omega_{[27]} = \{1\};\ \Omega_{[1]} = \emptyset$$

Thus, $[[1,27,28]]$ and $[[1,27],[28]]$ are two possible solutions with 3 and 5 deadheads corresponding.

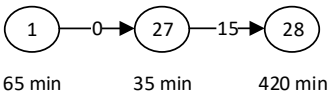
Beyond the black box

Initialization

For each $x \in D$, we must assign new vehicle since x must be the end of the route.
Hence without **Restricted legal vehicle** constraint the absolute minimum of vehicles in daily schedule is $|D|$ and the absolute minimum **Daily schedule OpEx** is $2|D| \cdot 15$.

Selecting x from D only problem

Assuming $N = \{1,27,28\}$, and



$$S = \begin{pmatrix} False & True & False \\ False & False & True \\ False & False & False \end{pmatrix} \quad P = \begin{pmatrix} NaN & 0 & NaN \\ NaN & NaN & 15 \\ NaN & NaN & NaN \end{pmatrix}$$

Hence $D = \{28\}$

Selecting $[28]$ will produce the schedule $[[1],[27,28]]$ with 5 deadheads, where the minimum is 4 deadheads with $[[1,27],[28]]$ schedule.

Solving that with initialize:

$$I = L \cup D$$

Thus, initialize vehicle with $x \in I$ is necessity for minimization.

Vehicle beam search

In this paper, we consider a beam search with scoring. However, unlike the conventional beam search applying a constructive heuristic for evaluation, the evaluation of our beam search is based on local evaluation $s(v)$ and global evaluation which try to minimize $|D|$ in each iteration.

The root of tree in the beam search is always an element from I. Then beam search first generates all restricted candidates at first level and evaluates those nodes by local evaluation $s(v)$. The evaluation estimates the deadhead penalty and ratio of how close one node to each other. The best β nodes are kept as beam nodes to generate restricted candidates in next level.

At the end, from β vehicles the one which will minimize $|D|$ will be taken.

Score function

$$s(v) = \begin{cases} \sum_{t=o}^{|v|-2} \frac{w_{v_t,v_{t+1}}}{15} - \frac{departure(v_t)}{departure(v_{t+1})}, & |v| > 1 \\ 0, & otherwise \end{cases}$$

Note: lower score minimize deadhead in v.

Solution

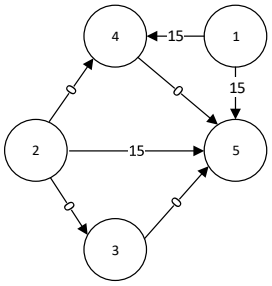
The solution is described by repeated insertions during the exploration of beam search. In the following subsections, the implementation details of the proposed beam search heuristic provided, and the procedure is described as follows:

Step 1. Determine the filter of latest departures (l) and beam width (β).
Step 2. Set R, Valid daily Schedule. $R = \emptyset$
Step 3. Obtain an initial vehicles candidate I and set $V = I$
Step 4. For each candidate $v \in V$ generate Ω_v.
Step 5. For each $t \in \Omega_v$ create new candidate $v' = t + v$
Step 6. Add v' to V and drop v from V.
Step 7. Evaluate each candidate $v \in V$ by a local evaluation $s(v)$.
Step 8. Select the best β vehicle candidates as beam nodes according to the score evaluation value of each vehicle.
Step 9. If there is vehicle $v \in V$ with $\Omega_v \neq \emptyset$ go to step 4.
Step 10. Select $v \in V$ minimize $|D|$.
Step 11. Add v to R.
Step 12. For $t \in v$ add t to C.
Step 13. If $N \backslash C \neq \emptyset$ go to step 3.
Step 14. R is valid schedule.

Example

Dataset

Trip id	origin	destination	departure	arrival
1	2	6	200	250
2	1	8	200	250
3	8	9	240	450
4	8	9	280	450
5	9	10	500	700



- Setting $\beta = 2$; $l = 1$
- $N = \{1,2,3,4,5\}$; $C = \emptyset$;
- $S = \begin{pmatrix} False & False & False & True & True \\ False & False & True & True & True \\ False & False & False & False & True \\ False & False & False & False & True \\ False & False & False & False & False \end{pmatrix}$ $P = \begin{pmatrix} NaN & NaN & NaN & 15 & 15 \\ NaN & NaN & 0 & 0 & 15 \\ NaN & NaN & NaN & NaN & 0 \\ NaN & NaN & NaN & NaN & 0 \\ NaN & NaN & NaN & NaN & NaN \end{pmatrix}$

Iterations

- $D = \{5\};\ L = \{5\};\ \Rightarrow\ I = \{5\};\ V = \{[5]\}$
- $\Omega_{[5]} = \{1,2,3,4\}$
- Evaluation
  - $s([1,5]) = 1 - \frac{200}{500}$
  - $s([2,5]) = 1 - \frac{200}{500}$
  - $s([3,5]) = -\frac{240}{500}$
  - $s([4,5]) = -\frac{280}{500}$

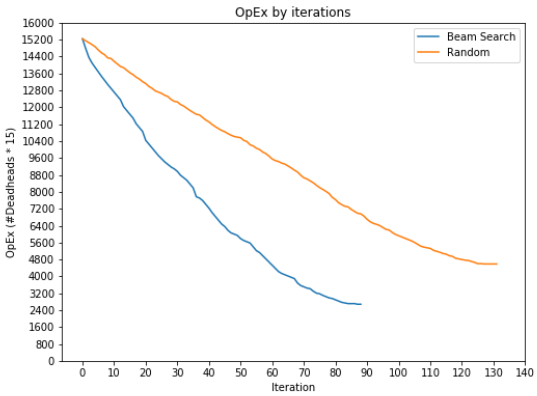
- Best 2 candidates are $[3,5]$ and $[4,5]$, in addition $\Omega_{[3,5]} = \{2\}$ and $\Omega_{[4,5]} = \{1,2\}$ therefore returning to step 4
- Evaluation
  - $s([2,3,5]) = -\frac{240}{500} - \frac{200}{240}$
  - $s([1,4,5]) = -\frac{280}{500} + 1 - \frac{250}{280}$
  - $s([2,4,5]) = -\frac{280}{500} - \frac{250}{280}$
- Best 2 candidates are $[2,3,5]$ and $[2,4,5]$ with $\Omega_{[2,3,5]} = \Omega_{[2,4,5]} = \emptyset$
  - $v_1 = [2,3,5] \Rightarrow D = \{1\}$
  - $v_2 = [2,4,5] \Rightarrow D = \{1,3\}$
- $v_1 = [2,3,5]$ is the chosen vehicle by global evaluation
- $C = \{2,3,5\}$; $N \backslash C = \{1,4\} \neq \emptyset$. therefore, will return to step 3.
  - $S = \begin{pmatrix} \text{False} & \text{True} \\ \text{False} & \text{False} \end{pmatrix}$
  - $P = \begin{pmatrix} \text{NaN} & 15 \\ \text{NaN} & \text{NaN} \end{pmatrix}$
  - $D = \{4\};\ L = \{4\};\ \Rightarrow\ I = \{4\}$
- $\Omega_{[4]} = \{1\}$
- Evaluation
  - $s([1,4]) = 1 - \frac{250}{280}$
- The only candidate will be taken
  - $v_3 = [1,4] \Rightarrow D = \emptyset$
- $v_3 = [1,4]$ is the chosen vehicle
- $N \backslash C = \emptyset$.
- The algorithm finished with the optimal solution of Valid daily Schedule $= \{[2,3,5],[1,4]\}$ and Daily schedule OpEx = $5 \cdot 15$

Experiment

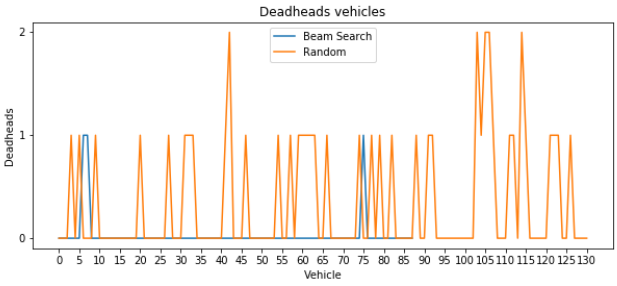
The proposed beam search was coded in Python 3.7.4 and run on a PC with Intel Core i5 8350U CPU @ 1.7 GHz, RAM 16GB under Windows 10 operations system. The beam search was tested with a dataset proposed by ©Optibus. The Dataset contain 508 service trips with 12 stations and trip's duration are gaussian distributed. Find here full dataset. The proposed beam search algorithm compered against random selection (random selection from Ω_v). The Schedule OpEx achieved is **2685** with **179** deadheads using Beam Search and **4590** with **306** deadheads using random selection. Find here full Experiment notebook.

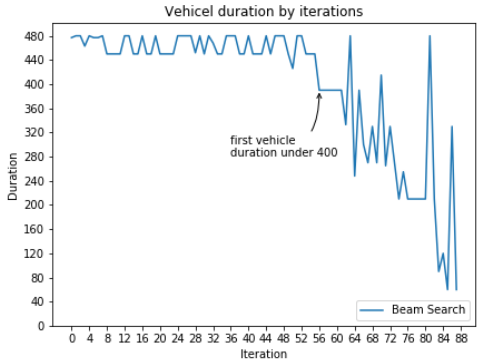
OpEx by iterations *– starting with Vehicle for each trip in our dataset. In each iteration the beam search and the random selection are merging vehicles.*

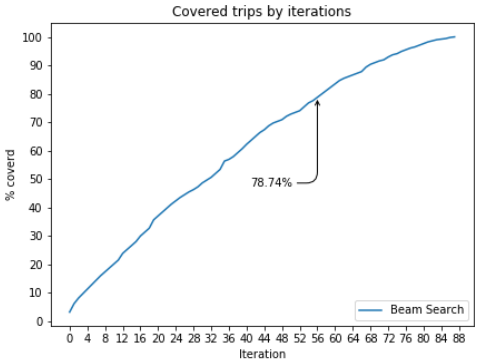
We can see the both curves are monotonically non-increasing and we achieved linear convergence with much better slope with beam search.



Deadheads vehicles *– how many deadheads were added by each vehicle. Only Vehicles created at iterations 6,7 and 75 added deadheads (3), total of 88 vehicles created 179 deadheads in beam search while pulse graph described by random selection.*







Covered trips by iteration *– showing how many trips are covered in each iteration. We can observe linear curve.*

Vehicle duration by iteration *– the max driving time for vehicle defined as 480 minutes. We can see that from iteration 0-56 we maximize the vehicles driving time and we covered around 80% of the trips which are the optimal solution. It seems like in the last 20% of the trips we have pulse vehicles creation which may can be optimize by running the solution again on the last 20%.*

Conclusion

We have proposed a beam search heuristic for the deadheads routing problem in this paper. Both local evaluation and global evaluation based on insertion are applied in this paper, in which an initial solution is needed. To reduce computational complexity, score functions for checking feasibility are used in local evaluation step, where the feasibility of solutions can be checked with computational effort of $O(1)$. Regardless of the feasibility of the initial solution, our beam search can either fix the infeasible solution or improve the solution. Our beam search is tested on full ©Optibus dataset and on random 100 trips obtained from the Dataset with the same result (notebooks). The results showed that 80% vehicles can achieve **optimal** solution in small computational effort, and the gap of remainder vehicles is the driving time and not the deadheads. The results show that our beam search can obtain good solutions with effective computational times.