Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 00 (2017) 000–000

Procedia Computer Science

www.elsevier.com/locate/procedia

4th Information Systems International Conference 2017, ISICO 2017, 6-8 November 2017, Bali, Indonesia

Advanced Traveler Information System: Itinerary Optimization As an Orienteering Problem Using Iterative Local Search-Hill Climbing Algorithm

Jockey Satria Wijaya, Wiwik Anggraeni, Ahmad Muklason*, Faizal Mahananto, Edwin Riksakomara, Arif Djunaidy

Department of Information Systems, Information Technology, Institut Teknologi Sepuluh Nopember
Jl. Raya ITS, Kampus ITS Sukolilo, Surabaya, 60111

Abstract

Traffic congestion is a problem that becomes government concern in many big cities like Surabaya. Some initiatives have been carried out by the government to decrease the intensity of traffic congestion, ranging from various policy like special lane for motorcycle, odd – even license plate, etc. However, it is yet to be successful to decrease the heavy traffic in many big cities in Indonesia. The right solution tends to encourage people to use public transport instead of private cars. Within the framework of Intelligent Transport System, this paper proposes an advanced traveller information system with itinerary optimisation by Angkot, a common public transport mode in Indonesia. The itinerary optimisation is modelled as an orienteering problem and is solved using iterative local search – hill climbing algorithm. The experiment results show that generally the proposed algorithm could solve the problem fast and effectively. In addition to propose the initial algorithm, the main contribution of this paper is a new dataset for orienteering problem that may encourage researcher to come up with more sophisticated algorithm. Eventually, the proposed system could encourage more people to use public transport.

Peer-review under responsibility of the Conference Program Chairs

Keywords: Orienteering Problem, Iterative Local Search, Intelligent Transport System

* Corresponding author. Tel.: +0-000-000-0000 ; fax: +0-000-000-0000 .
E-mail address: mukhlason@is.its.ac.id

Peer-review under responsibility of the Conference Program Chairs.

1. Introduction

Traffic congestion is a problem that becomes a problem in a capital and any other big cities like Jakarta and Surabaya. According to Stop-Start Index by Castrol-Magnatec published by Castrol, an oil company based in England, Jakarta's caught in the most traffic jams on Earth follows with Instanblu and Mexicto City. Meanwhile, Surabaya take 4th place as city with the worst traffic [1]. Many factors can cause traffic congestion in a big city, such as, dense population, unregulated traffic, bad road quality, etc. For the total population itself, Jakarta placed 1st as the most crowded city with 9,988,485 people, followed by Surabaya with 2,863,059 people [2].

Traffic congestion is a problem that becomes government attention for cities like Jakarta and Surabaya. It wasted someone's time, in addition to brings up huge carbon pollution. Urban areas whose streets are filled with vehicles produces exhaust gases including *Geenhouse Gases* (GHG) that can decrease the air quality [3]. In fact, a death can occur because of traffic congestion. According to a report from medical head in Brebes, there 18 people died on very crowded road. 12 of them died because of exhaustion, 5 of them died because of traffic accident, and 1 person died because of unknown reason [4]. One of the victims is a 14 years old who died because of carbon dioxide poisoning after the car she rode got caught on a traffic jam for more than 6 hours [4].

The government has made several efforts to decrease congestion, such as, building inner-city toll roads, regulate an odd and even plate numbers, restriction road for motorcycle, addition of road segment, etc. However it is still considered less helpful to reducing congestion in the big cities.

To anticipate and decrease the congestion, Surabaya's government has made additional roads, mode orderly traffic arrangements, and improve public transportation like building a Mass Rapid Transit (MRT), and plan to revitalize angkot [5]. Improvement of public transportation is an appropriate effort to decrease congestion, because one of the congestion factors is high volume of vehicles. Therefore, one of the efforts to decrease the congestion is establish and improve public transportation.

However, in the presence of public transportation only, it will not get society attention to use it rather than private vehicles. To get the attention, government must provide a comfortable, fast, cheap, in time and scheduled vehicle. Surabaya has Surabaya Intelligent Tranposrt System (SITS) an information system which aims to improve traffic smoothnes and security.

The implementation of Intelligent Transport System (ITS) in Surabaya is still at the stage of monitoring traffic congestion in real time using CCTV which already planted in the streets of Surabaya. This research aim to give input in the form of matemathical model which can be used to enhance SITS in the future. Based on that purpose, optimization modelling in this research is done with Orienteering Problem method.

Orienteering Problem (OP) is a combination of node selection and determine the shortest path from selected node. Therefor, OP can be seen as a combine method from Knapsack Problem (KP) and Travelling Salesman Problem (TSP). In TSP, we find the shortest path to visit all nodes, while in OP we don't need to visit all nodes before reaching the goal [6]. The research conducted by Vansteenwegen et al. (2011) states that OP or variant of OP can model a complex practical application, ex: tourism application which needs a fast and effective solution [6]. In this study, we think it's appropriate to use OP because of its characteristic which is finding an optimum route based on existing lane without the need to visit all nodes.

One of the method to find a solution for OP model is Iterative Local Search. With the Iterative Local Search method, we can find OP model solution fast and effectively [7]. With this research we hope to help Surabaya's government develop SITS, and also become additional reference related to OP and optimization in general.

2. Literature Review

2.1 Orienteering Problem

Orienteering Problem (OP) is a variant from *Travelling Salesman Problem* (TSP) using *score*. *Score* in OP is a profit or satisfaction obtained by visiting a node (city, place, location). The name Orienteering Problem originates from sport game of orienteering, which is commonly played in a jungle or forest. In OP, there are checkpoints (landmark). Every checkpoint has a score and can only be visited once. Individual competitors start at specified checkpoint, try to visit as many checkpoints as possible within given time frame. The objective is to maximize score within given time frame [8].

OP can be formulated as an integer programming model with $X_{ij}=1$ as decision variable if a visit to node j followed by node I, otherwise it become $X_{ij}=0$. Variable u_i is used for the constrain subtour elimination and allows

to determine visited node position. OP is formulated as follows: [8]:

$$Maximize \sum_{i=2}^{|N|-1} \sum_{j=2}^{|N|} S_i X_{ij} \quad (0)$$

Objective function (0) is for maximize total score collected. In addition, OP also has some constraints as follows:

$$\sum_{j=2}^{|N|} X_{1j} = \sum_{i=1}^{|N|-1} X_{i\,|N|} = 1 \quad (1)$$

Constraint (1) ensure the path starts in node 1 and ends in node $|N|$

$$\sum_{i=1}^{|N|-1} X_{ik} = \sum_{j=2}^{|N|} X_{kj} \leq 1; \forall k = 2, \dots, (|N| - 1) \quad (2)$$

Constraint (2) Ensure the connectivity of the path and guarantee that every node is visited at most once.

$$\sum_{i=1}^{|N|-1} \sum_{j=2}^{|N|} t_{ij} X_{ij} \leq T_{max} \quad (3)$$

Constraint (3) ensure the limited time budget

$$2 \leq u_i \leq |N|; \forall i = 2, \dots, |N| \quad (4)$$

$$u_i - u_j + 1 \leq (|N| - 1)(1 - X_{ij}); \forall i = 2, \dots, |N| \quad (5)$$

Constraints (4) and (5) prevents subtour, a condition where the path start and end in the same node.

2.2 Iterative Local Search

The objective of Iterative Local Search is to improve stochastic search by doing a wider sampling and use local search to make the solution optimal. There are several search methods in Iterative Local Search, such as:

a. *Deletion Method*: Except the starting and the ending nodes, beginning with the 2nd node on a tour, eliminates a node from the tour.
b. *Swap Method*: Swap a pair of nodes existing on a tour, except the starting and ending nodes.
c. *Insertion Method*: Insert unvisited nodes, if any, in any position between starting and ending nodes.

Code 1 is a pseudo code in general for the iterative local search [9]:

```
Procedure Iterative Local Search
s₀ = Generate Initial Solution
s* = LocalSearch(s₀)
Repeat
        s' = Perturbation (s*, history)
        s*' = LocalSearch (s')
        s* = AcceptanceCriterion (s*, s*',
        history)
        until termination condition met
end
```

Code 1 Pseudo Code Iterative Local Search

Iterative local search (ILS) will continue to change the solution until there is no better solution and the number of iterations reached the maximum limit. Local search can be done with a wide range of method like swap, insert, delet, replace, shake step, etc. In this study, we use only swap, insert, and delete to find the solution for OP model.

There are a many heuristic search technique in ILS, one of them is hill climbing search. With the hill climbing strategy, we always evaluate every new solution and choos the best one. The search stops when the algorithm can't fine a better solution. The strategy is named hill climbing because it is likened to a passionate mountain climber but do not know the direction to climb, alwas pick the steepest climb until they reach the peak.

3. METHODOLOGY

Here is the workflow for this research, as shown in Fig. 1.

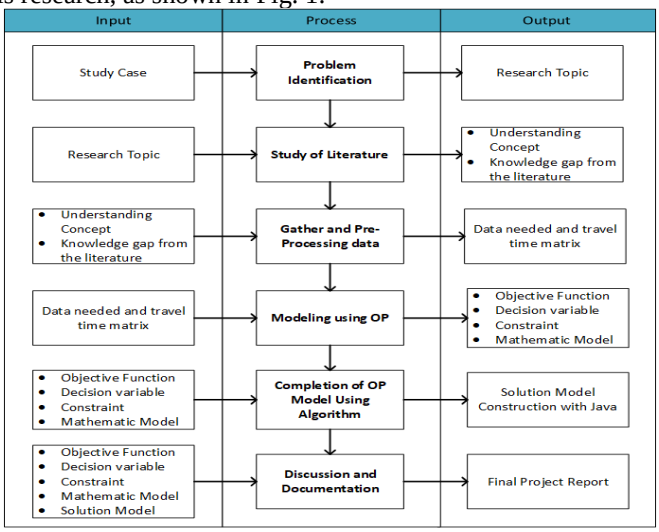

Fig. 1 Methodology

3.1 Problem Identification

We identify the problem by analyzing traffic condition in Indonesia especially Surabaya.

3.2 Study of Literature

We gather reference like research, books, and related documents. In this stage, we get to know the problem, objective, and constraint for this topic.

3.3 Gather and Pre-Processing Data

In this study, we need related data. Data obtained from Dinas Perhubungan Surabaya and Google Maps, such as:

- Line route for Angkot with 6 routes (F, JMK, JK, GS, RT, O)
- Number of vehicles on each route
- Travel time data of each stop

Pre-processing carried out by entering travel time data into matrix.

3.4 Modeling with Orienteering Problem

In this stage, we analyze the problem to determina objective function and constraints. From the data, we determine the score, node, arc to be able to create OP model. In this research, every road traversed by angkot acted as a node. Distnce between node ranges from 200 to 1000 meters, by reference from Departemen Perhubungan Darat [11]. For road that have distance for more than 1000 meters, will be broken so on that road has 2 or more nodes. For road that have distance for less than 200 meters, the location of that node is determine after that road merged with the next or previous road, so there are no node that have a distance between 200 meters and 1000 meters. Therefore, the total nodes in this study is 199 nodes. The following is a description for score, node, arc, and tMax:

1. Score: The number of angkot passing through the node
2. Node: The point that angkot passes
3. Arc: Travel time between node
4. tMax: The maximum travel time desired

3.5 Completion of OP model using algorithm

We create solution for the created model from the previous step. This stage is divided into severap sub-step:

1. Finding the shortest path between node
2. Determine the solution search parameter
3. Find the initial feasible solution
4. Roulette Wheel Selection
5. Iterative Local Search and retrieve the best solution

3.6 Discussion and Documentation

All the above steps will be compiled in a report as a form of documentation on this research.

4. IMPLEMENTATION

Forming Travel Time Matrix

The results of travel time data on every node obtained from Google Maps is still unreadable to Java, because it's still a table on excel. Therefore, data pre-processing needs to be done to be usable for optimation. Formed a matrix that shows travel time between one node to another node.

This matrix will be a table with 199 columns and 199 rows. It will represent every node position based on the network model. Every cell contains travel time that read from row to column. For the better understanding, Table 1 is a portion from the matrix.

Dijkstra's Algorithm Application

Table 1 still does not contain travel time value between one node to all the other nodes. It will be difficult for iterative local search algorithm to find the initial feasible solution from such data.

Table 1 Portion of Travel Time Matrix

	1	2	3	4	5	6	7	8
1	0	2	0	0	0	0	0	0
2	1	0	1	0	0	0	0	0
3	0	0	0	1	0	0	0	0
4	0	0	1	0	1	0	0	0
5	0	0	0	1	0	2	0	0
6	0	0	0	0	0	0	1	0
7	0	0	0	0	0	1	0	2
8	0	0	0	0	0	0	0	0

To find the initial feasible solution, every cell on Table 1 must be known. To find it, we can use Dijkstra's algorithm. Dijkstra's algorithm will find the shortest path from source node to all the other nodes within the graph.

Initial Feasible Solution

To find the initial feasible solution, maximum travel time and iteration needs to be declared.

```
1.  int tMax = 60;
2.  int maxLoop = 1000000;
3.  int loop = 1;
```

Code 1 Declare tMax and maxLoop

Code shows that finding solution with 1,000,000 iteration. The maximum travel time between strating node and ending node is 60 minutes. After we declare maximum travel time and iteration, next thing we do is search for the solution.

```
1.  do {
2.  int Rn = new Random().nextInt(jumlNode - 2) +
    1;
3.  ArrayList<Integer> listRl = new ArrayList();
4.  ArrayList<Integer> listRl1 = new ArrayList();
5.  for (int i = 1; i < jumlNode + 1; i++) {
6.  if (i == nodeAwal || i == nodeAkhir) {
7.  continue; }
8.  listRl1.add(i);  }
9.  Collections.shuffle(listRl1);
10. listRl.add(nodeAwal);
11. for (int i = 0 ; i < listRl1.size(); i++) {
12. listRl.add(listRl1.get(i)); }
13. listRl.add(nodeAkhir);
```

Code 2 ArrayList to store all route

Array listRl works to store every node in a form of arraylist, while listRl1 works to store a randomized-position nodes. Then, we create array named subListRs which contains all of listRl array. We also need to create method to count the total travel time and total skor for every solution produce by the algorithm. Then we create a function that store feasible solutions into array named feasibleSol:

```
1.  for (int i = 0; i < subListRs.size(); i++) {
2.  if (kalkulasiJarak(subListRs) < tMax &&
    !feasibleSol.contains(subListRs)) {
3.  feasibleSol.add(subListRs);  }   }
4.  loop++;
5.  } while (loop <= maxLoop);
```

Code 3 Store the Initial Feasible Solution

Code will search for a solution from arraylist subListRs with travel time less that tMax and store it to arraylist feasibleSol. The code will not store a solution that already store in feasibleSol before. This way, solution that have less travel time than tMax will be considered as initial feasible solution.

Roulette Wheel Selection

Because of the result from previous step are too many, we need to select some solution which later optimize with iterative local search. Selection is done using roulette wheel selection because of its characteristic that doesn't always select the best solution, and not randomly.

Iterative Local Search

Based on the selection process, all the results of the selection will be optimized using iterative local search. The local search used in this study is swap method, insertion method, and deletion method. The local search is done repeatedly until maximum iteration is reached. If the result after local search has smaller travel time, or bigger score, then the new solution replaces the previous one.

Retrieve the Best Solution

Retrieval of the best solution is done by checking every optimized solution and select solution with the highest score.

5. RESULT AND DISCUSSION

Roulette Wheel Selection Result

In this experiment, we try to change the total size for roulette wheel selection (RWS). The program will run 5 times for every RWS size. From 5 runs on every RWS size, we select a solution with the highest score as an optimal solution. In the first experiment we select 10 solutions. In the second experiment, we select 30 solutions, and the third experiment we select 50 solutions. Every experiment on this section, we use node 1 as a starting node, and node 91 as ending node with tMax is 60 minutes. Program will search for initial feasible solution 1,000,000 times, and do the local search 2,000,000 times. Table 2 is a comparison table for every RWS size.

Table 2 Size RWS Comparison

Size RWS	Route	Time	Score
10	[1, 2, 3, 4, 5, 6, 9, 36, 50, 51, 52, 53, 35, 34, 22, 23, 7, 8, 10, 11, 12, 13, 14, 15, 91]	58	4626
30	[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 199, 103, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 91]	58	5421
50	[1, 2, 3, 4, 5, 6, 7, 8, 54, 182, 183, 184, 185, 186, 187, 190, 191, 192, 193, 194, 195, 196, 106, 10, 11, 12, 13, 14, 99, 146, 169, 83, 84, 85, 91]	56	5433

As we can see in Table 2, the larger the size of roulette wheel, then the larger the chance to get the highest score. Therefore, we can conclude that in this experiment the size of the roulette wheel influence the search. The larger the size, then the chance to get a better solution is higher

Comparison of Every Iterative Local Search Method

In this experiment, we try to change the method use in every experiment, we do it 4 times. Whereas in every experiment, the program will use only one method at a time, and every method in the last experiment. This experiment uses node 1 as the initial node and node 91 as the final node with tMax for 40 minutes. The number of solutions that will pass the roulette wheel selection is 30 solutions. The initial feasible solution search was done as many as 1,000,000 iterations, and local searches performed as many as 2,000,000 iterations. Table 3 is a comparison table of the 4 trials:

Table 3 Local Search Method Comparison

Method	Route	Time	Score
Swap	[1, 13, 14, 16, 91]	39	1219
Insertion	[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 99, 97, 98, 146, 169, 87, 91]	37	4220
Deletion	[1, 22, 184, 169, 91]	33	963
All in one	[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 99, 146, 169, 83, 91]	35	4145

Based on Table 3, the optimal solution search from node 1 to node 91 is suitable to use the insertion method, or a combination of the three iterative local search methods. Thus, in this experiment it can be said that the use of swap, insertion, and deletion method in iterative local search is also very influential in finding the optimal solution. Merging these three local search methods yields a good solution when compared to just using one of the local search methods, especially swap and delete.

6. CONCLUSION

From the findings, we can conclude that iterative local search optimization methods can produce good, quick, and optimal solutions of models that have been made. However, there are some issues that should be tackled as the future works: First, in this study, optimization is only in the scope of 6 routes F, JMK, RT, JK, O, and GS. Another route can be added to expand the coverage of angkot optimization in Surabaya City. Second, this research uses a random algorithm to find an initial feasible solution. It is advisable to use better algorithms such as Ant Colony Algorithm, Memetic Algorithm, and other optimization algorithms that can be used to search the solution so that the resulting solution can be more optimized. Third, optimization of solutions in this study only use local search

methods such as swap, insertion and deletion. The next development can be added other local search methods such as shake step, backward insertion, forward insertion and other local search methods. Lastly, we use number of angkot passing at one point as a score for OP. Representation of scores can be improved by considering several other factors such as how often the point is passed, the level of congestion, or how popular the point is in the city of Surabaya, so as to represent each point better.

REFERENCES

[1] Jakarta Globe, "Jakarta: World's Worst for Traffic Gridlock," Jakarta Globe, [Online]. Available: http://jakartaglobe.id/news/jakarta-worlds-worst-traffic-gridlock/. [Accessed 1 February 2017].

[2] Kementrian Dalam Negeri, "Permendagri No. 39 Tahun 2015," Kementrian Dalam Negeri, [Online]. Available: http://www.kemendagri.go.id/pages/profil-daerah/provinsi/detail/31/dki-jakarta. [Accessed 1 February 2017].

[3] M. Grote, I. Williams, J. Preston and S. Kemp, "Including Congestion Efects in Urban Road Traffic CO2 Emissions Modelling: Do Local Government Authorities Have the Right Options?," *Transportation Research Part D: Transport and Environtment,* vol. 43, pp. 95-106, 2016.

[4] S. Webb, "'Horror' Traffic Jam in Indonesia Lasted 35 Hours - and 18 Have Died on Same Stretch of Road in A Week," Mirror News, 8 July 2016. [Online]. Available: http://www.mirror.co.uk/news/world-news/horror-traffic-jam-indonesia-lasted-8377678. [Accessed 2 February 2017].

[5] M. Zulfikar and H. E. P. Gultom, "Ini Cara Tri Rismaharini Antisipasi Kemacetan di Surabaya," Tribun News, 29 November 2013. [Online]. Available: http://www.tribunnews.com/regional/2013/11/29/ini-cara-tri-rismaharini-antisipasi-kemacetan-di-surabaya. [Accessed 27 February 2017].

[6] P. Vansteenwegen, W. Souffriau and D. V. Oudheusden, "The Orienteering Problem: A Survey," *European Journal of Operational Research,* vol. 209, no. 1, pp. 1-10, 2011.

[7] P. Vansteenwegen, W. Souffriau, G. V. Berghe and D. V. Oudheusden, "Iterated Local Search for the Team Orienteering Problem with Time Windows," *Computers & Operations Research,* no. 36, pp. 3281 - 3290, 2009.

[8] A. Gunawan, H. Chuin Lau and P. Vansteenwegen, "Orienteering Problem: A Survey of Recent Variants, Solution Approaches and Applications," *European Journal of Operational Research,* vol. 255, no. 2, pp. 315-332, 2016.

[9] H. Lourenco, O. Martin and T. Stutzle, "Iterated Local Search," *Handbook of Metaheuristc,* pp. 321-353, 2003.

[10] Computer Science and Engineering UNT, "Dijkstra's Algorithm," [Online]. Available: http://www.cse.unt.edu/~tarau/teaching/AnAlgo/Dijkstra%27s%20algorithm.pdf. [Accessed 10 June 2017].

[11] Departemen Perhubungan Direktur Jenderal Perhubungan Darat, "Pedoman Teknis Perekayasanaan Tempat Perhentian Kendaraan Penumpang Umum," 1998. [Online]. Available: https://www.scribd.com/doc/74879620/PEDOMAN-TEKNIS-PEREKAYASANAAN-TEMPAT-PERHENTIAN-KENDARAAN-PENUMPANG-UMUM-HALTE. [Accessed 21 April 2017].

[12] J. Brownliee, "Iterated Local Search," Clever Algorithm, [Online]. Available: http://www.cleveralgorithms.com/nature-inspired/stochastic/iterated_local_search.html. [Accessed 20 June 2017].