Application of Large Neighborhood Search method in solving a dynamic dial-a-ride problem with money as an incentive

1, a R Adawiyah, 2, a Y Satria, 3, a H Burhan
1, 2, 3 Department of Mathematics, Faculty of Mathematics and Natural Sciences
Universitas Indonesia, Depok, 16424, Indonesia

Abstract. A Dynamic Dial-a-Ride Problem with Money as an Incentive (DARP-M) is a problem of finding optimal route to serve requests demand which uses taxi-sharing system with cost constraint. Taxi-sharing is a system where individual customer share vehicle with other customers, who has the same or similar origin, destination, and travel time. The optimal solution is the solution that can minimize the cost of each trip request. This study discusses DARP-M to optimize the use of taxi-sharing. The search for the DARP-M solution in this research uses the insertion heuristic method for construction of initial solution and the large neighborhood search method for the optimal solution determination. Then, the experiment uses three times periods, from which the experiment result shows the large neighborhood search method can minimize customers travel cost up to 27.40 % less than the cost of private rides.

1. Introduction
Ride-sharing is important for the economic development, because it is expected to reduce the number of private vehicle used, by optimizing the utilization of empty seats that are still available, by sharing a ride with other requests who have a similar time schedule or destination, and in the end it is expected to have impact in reducing congestion, pollution; and also fuel saving [1]. Ride-sharing can be applied not only to private vehicles, but can also be applied to taxis, which is then referred to as taxi-sharing. The reason for customers willing to share a ride is that if they could save money with that. To increase ride-sharing users which aims to save travel cost, we need to optimize the search for the matching between requests who want to share a ride. Refers to Santos & Xavier (2015) [2] they call this optimization problem by Dynamic Dial-a-Ride Problem with Money as an Incentive (DARP-M), and for the next we will mention this problem as DARP-M.

Note that taxi-sharing does not have a specific trip route and time restrictions either. Also, the price per km for taxi are fixed or same with all taxis. The goal of this study is to maximize the number of served request and minimizing the total value paid by all passengers. Based on research conducted by Santos & Xavier (2015) [2], this study discussed DARP-M to optimize the use of taxi-sharing. To solve this problem, we use insertion heuristic method for construction of initial solution and the large neighborhood search method for finding the optimal solution.

2. Literature Review
The Dial-a-Ride Problem (DARP) is an optimization problem that aims to find the optimal vehicle route to pickup and delivery requests between origins and destinations with time windows, and by
considering constraint like vehicle capacity and some other precedent constraints with minimum costs [3]. Example of the most common application of the DARP are door to door transportation services for elderly or disabled people [4].

The DARP generalizes a number of vehicle routing problems such as the Pickup and Delivery Vehicle Routing Problem (PDVRP) and the Vehicle Routing Problem with Time Windows (VRPTW) [3]. What makes the DARP different from most such routing problems is the focus on operational constraints that are related to transporting customers demand, so that high-quality services are needed. This is related to the time windows, maximum waiting time, and maximum ride time that we must able to minimize in order to reduce customer inconvenience. DARP services may operate according to a static or to a dynamic mode. In static mode, all requests are known beforehand while in the dynamic mode there are new requests coming during a travel time period.

3. Model Formulation of DARP-M

The DARP-M have a set $N$ with $n$ requests of passengers and a set $M$ with $m$ vehicles that are available for the taxi-sharing. Each request $i \in N$ has the origin point $i^+$, $i^-$ as a destination point, $e_i$ as the fastest departure time, $l_i$ as the latest arrival time, $p_i$ as the number of passengers and $s_i > 0$ as the maximum travel cost of passengers are willing to pay for a trip using taxi sharing system (in most of the cases $s_i$ is the cost of a private ride). Each vehicle $k \in M$ has the origin point $k^+$, $k^-$ as a destination point, $q_k$ as a maximum capacity of passengers inside the vehicle, and $e_k$ as the initial time to operate.

Let $V = \{i^+, i^- | i \in N\}$ be a set of all origin and destination points of all $n$ requests and $W = V \cup \{k^+ | k \in M\}$ be the set of all origin and destination points of all $n$ requests, including the origin point of all $m$ vehicle. Each point $w \in W$ has a service duration $d_w$, which is the time spend by passengers when to do a pickup and delivery operation and $p_w$ is the number of passengers in the vehicle. If $w$ is a destination point, $p_w$ will be negative. For origin point of vehicle $p_w = 0$, because we do not consider the driver as a passenger.

Next, each vehicle $k$ has two matrices that is $|V |\times |V |$ which shows travel time by $T_k$ and cost by $C_k$, between all pairs of points in $V \cup \{k^+ | k \in M\}$. The route $R_k = \{u_1, u_2, ..., u_z\}$ of a vehicle $k$ is a sequence of points that the vehicle passes. If a request $i$ is being served by vehicle $k$ then it must have $i^+$ and $i^-$ in route $R_k$. Note that $i^+$ must be served after $i^-$. For each point $u_j$ in a route $R_k$, $B^k_{u_j}$ is the time when the point $u_j$ will be served by vehicle $k$. And $B^k_{u_1} = e_k$ because the time at the first point is the same as the starting point of the vehicle $k$. And for $j > 1$, $B^k_{u_j} \geq B^k_{u_{j-1}} + t^k_{u_{j-1}, u_j} + d_{u_{j-1}}$ is the time when the point $u_j$ will be served by vehicle $k$ and must be bigger than the time at the previous point, this is related to time windows. With $t^k_{u_{j-1}, u_j}$ is travel time from point $u_{j-1}$ to point $u_j$ and $d_{u_{j-1}}$ is delay service duration at the point $u_{j-1}$. If $u_j = x^+$ then $B^k_{u_j} \geq e_x$ and if $u_j = x^-$ then $B^k_{u_j} \leq l_x$. Next for each point in route vehicle $k$ have the number of passengers $L^k_{u_j}$. For $j > 1$, $L^k_{u_j} = L^k_{u_{j-1}} + p_{u_j}$ and for $j = 1$ then $L^k_{u_1} = 0$ because we do not consider the driver as a passenger. Therefore must be $L^k_{u_j} \leq q_k$.

Then the last one, the cost of each requests trip must be calculated fairly. For shared trip, the cost of going from $u_j$ to $u_{j+1}$ in route $R_k$ must be divided equally to all requests in the vehicle $k$. So $S^k_{u_j, u_{j+1}}$ is the cost that each passenger of requests $i$ must be paid, if requests $i$ traveled to point $u_j$. Then we have that, $S^k_{u_j, u_{j+1}} = \frac{c^k_{u_j, u_{j+1}}p_i}{L^k_{u_j}}$, with $c^k_{u_{j+1}}$ is operational cost from $u_j$ to $u_{j+1}$ incurred by a vehicle $k$. So if we sum every cost of a request $i$, then it must be $S_i \leq s_i$.

The Dial-a-Ride Problem with Money as an Incentive, require the following conditions:

1. Precedence: The origin point of a request must be served first before the destination point.
2. Capacity: Each vehicle has a maximum capacity of passengers.
3. Time window: Each request and vehicle have a time interval that must be met.
4. Shared cost: Each request determines the maximum cost that can be paid for a shared ride.
5. No transfer: Each request can only be served by one vehicle.
6. Demand: All passengers of the same request must travel together in the same vehicle.

Moreover, we will explain the mathematical model for DARP. In DARP-M, the following four decision variables will be used:
- \( x_{i,u,v}^k = \begin{cases} 1, & \text{if the point } u \text{ to point } v \text{ is passed by vehicle } k \\ 0, & \text{otherwise} \end{cases} \)
- \( B_u^k \) is the time of the beginning of the service at point \( u \) in route of vehicle \( k \). If point \( u \) does not belong to route \( R_k \), this variable will be ignored.
- \( L_u^k \) is the number of passengers at point \( u \) inside vehicle \( k \). If point \( u \) does not belong to route \( R_k \), this variable will be ignored.
- \( S_{i,u}^k \) is the cost must be paid by request \( i \) when request \( i \) goes from point \( u \) to the next point in route \( R_k \). If request \( i \) is not served by vehicle \( k \) or if point \( u \) is not in route \( R_k \), this variable will be ignored.

Then, we need to optimize two criteria that will be the objective function of DARP-M.
1. Maximize the number of served requests.
2. Minimize the sum of the costs of all served requests.

The mathematical formulation of DARP-M can be found in Santos & Xavier (2015) [2], as follow:

\[
\text{max} \sum_{i \in N} \sum_{k \in M} \sum_{v \in V} \left( x_{i,v}^k S_{i,v} - \frac{c_{i,v}^k}{s_i} \right) 
\]

Subject to:

\[
\sum_{k \in M} \sum_{v \in V} x_{i,v}^k \leq 1, \quad \forall i \in N \tag{2}
\]

\[
\sum_{v \in V} x_{i,v}^k = 1, \quad \forall k \in M \tag{3}
\]

\[
\sum_{v \in V \cup \{k^+\}} x_{i,v}^k - \sum_{v \in V} x_{i,v}^k = 0, \quad \forall u \in V, \forall k \in M \tag{4}
\]

\[
\sum_{v \in V \cup \{k^+\}} x_{i,v}^k - \sum_{v \in V} x_{i,v}^k = 0, \quad \forall i \in N, \forall k \in M \tag{5}
\]

\[
x_{i,u,v}^k \in \{0,1\}, \quad \forall k \in M, (u,v) \in V \cup \{k^+\} V \cup \{k^\} \tag{6}
\]

\[
(x_{i,u,v}^k = 1) \rightarrow B_u^k \geq B_{i,u}^k + t_{i,u,v} + d_{i,v}, \quad \forall k \in M, \forall u, v \in V \cup \{k^+,k^-\} \tag{7}
\]

\[
e_i \leq B_{i,v}^k \leq B_{i,v}^k \leq l_i, \quad \forall i \in N, \forall k \in M \tag{8}
\]

\[
e_k \leq B_{i,v}^k, \quad \forall k \in M \tag{9}
\]

\[
(x_{i,u,v}^k = 1) \rightarrow L_u^k = L_{i,u}^k + p_{i,v}, \quad \forall k \in M, \forall u, v \in V \cup \{k^+,k^-\} \tag{10}
\]

\[
0 \leq L_{i,u}^k \leq q_k, \quad \forall k \in M, \forall u \in V \tag{11}
\]
\[ L_{k}^{k} = L_{k}^{k-} = 0, \quad \forall k \in M \]  \hspace{2cm} (13)

\[ (x_{u,v}^{k} = 1) \land (B_{l}^{k} \leq B_{u}^{k} < B_{l}^{k}) \rightarrow \frac{C_{u,v}^{k} p_{l}}{L_{u}^{k}}, \] \hspace{1cm} (14)

\[ \sum_{k \in M} \sum_{v \in V} S_{k,v}^{i} \leq s_{i}, \quad \forall i \in N \] \hspace{2cm} (15)

\[ S_{k,v}^{i} \geq 0, \quad \forall i \in N, \forall k \in M, \forall v \in V \] \hspace{2cm} (16)

The objective function is to maximize the number of served requests and minimizes the total of the costs of all served requests. In Santos & Xavier (2015) [2] they rewrite it using two auxiliary variables. Let \( y_{i} \) be a binary variable which will be one, if and only if, \( i \) is served by vehicle \( k \) or zero, otherwise. Let \( S_{i} \) be the total cost that each passenger of requests \( i \) must be paid. The objective function is: \( \max \sum_{i \in N} (y_{i} - \lambda \frac{S_{i}}{s_{i}}) \). Note that \( y_{i} = \sum_{k \in M} \sum_{v \in V} x_{u,v}^{k} \) and \( S_{i} = \sum_{k \in M} \sum_{v \in V} s_{i,v}^{k} \) and also note that \( s_{i} \geq S_{i} \). The smaller \( \frac{s_{i}}{s_{j}} \) is, the bigger the value of saving money.

Constraint (2) each request \( i \) at most is only served by one vehicle \( k \). Constraints (3) and (4) each vehicle \( k \) must have origin and destination point in its route. Constraint (5) is a flow conservation. Constraint (6) guarantees that the origin and destination point of request \( i \) will be served by the same vehicle \( k \), so it is not allowed to change vehicle \( k \) when the request \( i \) is being served. Constraint (7) variable \( x_{u,v}^{k} \) is a decision variable that is worth binary integers. Constraint (8) the time at point \( v \), if vehicle \( k \) goes from point \( u \) to point \( v \). Constraints (9) and (10) every request \( i \) and vehicle \( k \) has a time windows that must be met. Constraint (11) and (12) is about the number of passengers in vehicle \( k \) must be in the maximum capacity of the vehicle \( k \). Constrains (13) guarantees the number of passengers in vehicle \( k \) at origin and destination point vehicle \( k \) is zero, because we do not consider the driver as a passenger. Constraint (14) the cost that must be paid by the passengers in each part of the route. Constraint (15) the total of each request \( i \) is not greater than the maximum allowed that can be paid by request for a shared ride. Constraint (16) for each route that is passed by vehicle \( k \), the travel costs must be paid by request \( i \).

4. **Solution Methodology**

4.1. **Insertion Heuristic**

This section describes insertion heuristic method for construction of initial solution for finding feasible route, starting with selecting the first request to enter the route and then entering other request who have not entered the route repeatedly until all customers enter the route [5]. Furthermore, every possible route will be checked whether it is a feasible solution or not. This feasible solution is obtained if each request on the route meets the time windows, vehicle capacity, and shared cost. If there are more than one route that is a feasible solution, the route with the smallest insertion cost will selected.

Here is the formula for calculating requests insertion costs [6]:

\[ c_{1}(i,u,j) = \alpha_{1} c_{11}(i,u,j) + \alpha_{2} c_{12}(i,u,j) + \alpha_{3} c_{13}(i,u,j) \] \hspace{2cm} (17)

\[ \alpha_{1} + \alpha_{2} + \alpha_{3} = 1, \alpha_{1} \geq 0, \alpha_{2} \geq 0, \alpha_{3} \geq 0 \] \hspace{2cm} (18)

\[ c_{2}(i,u,j) = c_{1}(i,u,j) \] \hspace{2cm} (19)

Value \( c_{11} = D_{iu} + D_{uj} - \mu D_{ij}, \mu \geq 0 \) where \( D_{iu} \) is distance from point \( i \) to point \( j \), this value shows the addition of distance if request \( u \) is inserted between point \( i \) and point \( j \). \( c_{12} = T_{ju} - T_{j} \) where \( T_{ju} \) is the new time for service to begin at point \( j \). And \( c_{13} = b_{u} - T_{u} \). This formula aims to get
maximum profit by placing requests who will be inserted into an existing route rather than placing it on a new route.

4.2. Large Neighbourhood Search (LNS)

This method is used to minimize travel costs. The initial solution used in the LNS method is the final solution of the insertion heuristic method. This method starts with creating a sub-environment by using the destroy and repair method. The destroy method to destructs y% of route part of the current solution and choose x number of requests to be removed, note that after remove requests, the resulting route is a feasible. The repair method to rebuilds the destroyed solution by reinserting the requests that was removed by destroy method, provided that after reinserting the requests, the resulting route is a feasible solution. And then the results of the sub-environment will be evaluated to the evaluation function [7].

The evaluation function in this study, based on lexicographic ordering with the main objective to minimize travel costs [8].

\[ f(\sigma) = \sum_{r \in \sigma} p(r) \]  

(20)

This evaluation function is used to determine whether the selected sub-environment is a solution or not. The selected sub-environment solution can be accepted as a new solution if the solution is better than the previous solution based on lexicographic ordering.

5. Experimental Results

In this study, the parameters that exist in each method use are:

1. Objective function: We used \( \lambda = 0.99 \) to balance the second criteria of the objective function, because it will always be better to serve a request, even if there is no sharing which refers to Santos & Xavier (2015) [2].
2. Insertion heuristic: At this method we used variables \( \mu = 1, \alpha_1 = 0.5, \alpha_2 = 0.5, \) and \( \alpha_3 = 0 \) to calculate insertion costs refers to Solomon (1987) [6].
3. Large neighborhood search: At this method we used two variables \( x = 1 \) is the number of requests to be removed and reinserted on the route and \( I = 50,000 \) is the number of iteration for the solution search.

The implementation program uses programming language C with the software Code::Blocks version 17.12 and compiler TDM-GCC-64 on a hardware with 3.1 GHz Intel Core i5 processor specifications, memory 8 GB 2133 MHz LPDDR3, and macOS Sierra Version 10.12.6 Operating System.

Experimental data in this study using data dummy consisting of 512 nodes representing the location, 1457 edges representing the street connecting these locations. The graph will be used to created 3 time periods instances for static DARP-M. The time periods are in a certain time interval, there is a number of requests and taxis are different. Taxi service time in 3 periods from 07.00 until 10.00 which are converted to 0 until 180, with the duration of each period is 60 minutes. In each period, there is 30, 80, 150 taxis and 70, 170, 310 requests, and with the assumption that the capacity of the all taxis is for 4 passengers. The number of requests each period is independent and the number of taxis that will operate each period is dependent, that is related to the number of taxis in the previous period. Next, if the taxi is still delivering requests from the previous period, then the taxi is allowed to take new requests in the next period provided that the taxi must complete the request in the previous period first, then be allowed to take requests in the new period if the taxi is already empty. Because in the experiment it could be that taxis to delivery requests that have a time windows exceeds the periods time limit, this is why the problem DARP-M formed taxi-sharing is semi-dynamic. Taxi speed is constant 2 minutes for one km and the service duration each point is 2 minutes, which is the time taken to do pick up or delivery operation that is spend by passengers. Operational cost of taxi when there is a request in the vehicle is Rp.3,000 for one km.
Note that, on the results of experiment each period, not all requests can be served by a taxi, so there is a request which will be ignored and a number of taxis that did not served requests. The following information is about the number of taxis operating and requests that are served.

Table 1. Number of Operated Taxis and Number of Served Requests.

| Period | Operated Taxi | Served requests |
|--------|---------------|-----------------|
| 1      | 30            | 55              |
| 2      | 80            | 158             |
| 3      | 147           | 309             |

Next is the result of the use insertion heuristic method for construction of initial solution and the large neighborhood search method for finding the optimal solution in experiment of 3 times periods.

Table 2. Experimental Results in 3 Time Periods

| Period | Cost Private Ride | Insertion Heuristic | LNS |
|--------|-------------------|---------------------|-----|
|        |                   | Obj          | TC  | Sav %  | Obj    | TC  | Sav %  |
| 1      | 1318850.86        | 54.01        | 1065098.22 | 19.24   | 60.30 | 992363.49 | 24.75   |
| 2      | 3905864.36        | 161.49       | 3216717.41 | 17.64   | 186.16 | 2835510.09 | 27.40   |
| 3      | 7643070.86        | 319.74       | 6366624.08 | 16.70   | 382.04 | 5602004.83 | 26.70   |

Table 2 shows results in 3 time periods. For each period, there is information about cost of private ride, the objective function value (Obj), total cost shared ride (TC), and the money saved made by requests (Sav%). The result in each period is the best solution obtained by experimenting 5 times in the same data. Insertion heuristic method for construction of initial solution has a solution that can minimize travel cost until 19 % and the large neighborhood search (LNS) method can minimize passengers travel cost significantly until 27.40 % in period 2. Therefore, the results of 3 periods shows that the large neighborhood search (LNS) method can solve DARP-M problems in taxi-sharing optimization with good result which can minimize travel cost to be cheaper than private rides cost. And also with the taxi-sharing system, it can reduce the number of taxi fleets operating compared to if all requests must be served on a private rides.

6. Conclusion
This study discusses the problem of taxi-sharing optimization in the form of DARP-M using insertion heuristic method for the initial solution and the large neighbourhood search (LNS) method to finding optimal solution. From the experiment result, we can conclude that the insertion heuristic method can build initial solution that are good enough for minimize travel cost and LNS method manage to produce a better solution than initial solution, with requests paying up to 27.40 % less than they would pay on private rides.

Acknowledgement
This research was supported by Hibah PITTA 2018 funded by DRPM Universitas Indonesia No.4230/UN2.R3.1/PPM.00.01/2018.

References
[1] Agatz N, Savelsbergh M, Erera A and Wang X 2012 Optimization for dynamic ride-sharing: A review European Journal of Operational Research 223(2) 295-303
[2] Santos D O and Xavier E C 2015 Taxi and ride sharing: A dynamic dial-a-ride problem with money as an incentive Expert Systems with Application 42 6728-6737
[3] Cordeau J-F and Laporte G 2007 The dial-a-ride-problem: Models and algorithms Annals of Operations Research 153 29-46
[4] Toth P and Vigo D 1997 Heruristic algorithms for the handicapped persons transportation
problem \textit{Transportation Science} \textbf{31} 60-71

[5] Rosenkrantz D J, Stearns R E and Lewis P M 1997 An analysis of several heuristics for the traveling salesman problem \textit{SIAM Journal of Computing} \textbf{6} 563-581

[6] Solomon M M 1987 Algorithms for the vehicle routing and scheduling problem with time windows constraints \textit{Operations Research} \textbf{35}(2) 254-265

[7] Pisinger D and Ropke S 2010 Large neighborhood search In M Gendreau \textit{Handbook of metaheuristics} \textbf{2} 399-420 Spinger

[8] Bent R and Hentenryck P V 2004 A two-stage hybrid local search for the vehicle routing problem with time windows \textit{Transportation Science} \textbf{38} 515-530