Metaheuristic Algorithms for the Many-to-one IRP with Dynamic Velocity

F Ramadhan1 and A Imran2

1, 2Department of Industrial Engineering, Institut Teknologi Nasional, Bandung 40124, Indonesia
*corresponding author: f.ramadhan@itenas.ac.id

Abstract. Inventory Routing Problem (IRP) is a combination of inventory management and transportation optimization problems that involve route selection, number of product pickups, and customer demands. In the many-to-one IRP model, the vehicle is sent from the depot and goes to pick up products from several suppliers to the assembly plant. Vehicle loads on this model will increase every time a product is taken from the supplier. It can decrease vehicle speed in the next travel process (dynamic velocity). There are still a few papers that discuss many-to-one IRP models with dynamic velocity. Therefore, this study aims to develop a new model called the many-to-one IRP model with dynamic velocity. A modified threshold accepting, variable neighborhood search, and record-to-record travel algorithm with the first improvement local search strategy is used to solve the IRP model. Small datasets many-to-one IRP from previous researches and experimental tests are used to test the algorithms. The results showed that the best-proposed algorithm is competitive when compared to the best-known solution in the previous studies (the average deviation is only 1.86%).

1. Introduction
Inventory routing problem (IRP) was first introduced by Bell et al. [1] as a model that combines inventory management and transportation optimization problems. IRP has an important role in optimizing the company's distribution strategy [2]. This model involves several aspects, such as route selection, number of product pickups, and customer demands [3]. IRP is categorized as NP-hard with a higher level of complexity than cases in the vehicle routing problem (VRP) [4].

There were several previous researches in IRP. Sindhuchao et al. [5] developed a multi-item IRP, so the cost of each product item varied. Archetti et al. [6] identified a multi-customer IRP, which the process is adapting to a VRP consisting of many customers. The combination of the number of customers and product types (including multi-period time) was developed by Moin et al. [7], called multi-product multi-period IRP. In addition to all these variations, there are IRP variations that focused on demand constraints, namely the stochastic IRP by Solyali et al. [8] and dynamic IRP by Bertazzi et al. [9]. In recent years, dynamic constraints have become the focus of development, such as transhipment [10] and perishable products [11].

Although many researchers have developed the IRP model, there is still little to discuss the many-to-one IRP model. Coelho et al. [12] conducted a systematic literature review related to the development of the IRP model over the past 30 years comprehensively. The research showed that no one discussed the many-to-one IRP model. In the many-to-one IRP model, the vehicle is sent from the depot and goes to pick up products from suppliers to the assembly plant [13], [14]. Vehicle loads on this model will always increase every time a product is taken from the supplier. It can cause the vehicle speed to decrease in the next travel process (dynamic velocity). There are a few papers that discuss many-to-one IRP models with dynamic velocity.
A model related to transportation routes that comprehensively addresses the dynamic velocity of vehicles is the travelling thief problem (TTP). TTP was first introduced by Bonyadi et al. [15]. This model combines the travelling salesman problem and the knapsack problem. Both models are combinatorial optimization problems which are popular and have been widely developed in theoretical and experimental researches [16]. TTP considers dynamic velocity as an essential aspect in the objective function of the problem, where vehicle speed can change depending on the product load in the vehicle [17]. When the vehicle is empty, the travelling speed can reach the maximum speed, but the minimum speed occurs when the vehicle capacity is full [18]. Therefore, the basic IRP model can be combined with the dynamic velocity in the TTP model, and this combination can make the IRP model evolve and can be implemented in real conditions.

There were several previous methods developed to tackle the IRP. Moin et al. [7] developed a hybrid genetic algorithm with a new set of operators and crossovers motion. Mladenovic et al. [13] put forward a two-phase variable neighborhood search (VNS) that produces the best total costs when compared to the optimal global solutions generated from CPLEX. Xiao and Rao [19] proposed a fuzzy genetic algorithm that produced good results to solve IRP. In the latest researches, the development of algorithm in the IRP area is still ongoing, such as the combination of mixed-integer programming and adaptive large neighborhood search by Ghami et al. [20], a modified threshold accepting (TA) by Ramadhan et al. [21], or the extension of VNS algorithm by Karakostas et al. [22]. Recent researches such as TA and VNS produce good and competitive output to complete the IRP model so that the two algorithms can be further developed to obtain better solutions.

Based on the previous researches, there are many metaheuristic methods to address the IRP model, but none of them uses the record-to-record travel algorithm (RRT). Dueck [23] first introduced a new optimization heuristic called the RRT algorithm. This algorithm can accept a worse solution than the current solution on each iteration. Uphill move local search is conducted to avoid being trapped in the local optimal solution. Li et al. [24] developed the RRT algorithm with a two-phase procedure. In the first phase, the least cost insertion algorithm is used to get a good initial solution. The second phase contains the addition of several parameter values are used for the uphill, downhill, and global iteration processes. Li et al. [24] compared several metaheuristic algorithms with RRT. The results of their research indicate that the total cost generated by the RRT algorithm is better than other metaheuristics [25], [26]. There are few papers that discuss IRP with dynamic vehicle speed on the TTP model.

Therefore, this study aims to develop a new model called the many-to-one IRP model with dynamic velocity. A modified RRT algorithm is used to solve the IRP model. Besides that, the first improvement local search strategy by Mladenovic et al. [27] is combined with the proposed algorithm to improve the IRP solutions. The first improvement strategy has been proven to produce better output with fast computational time compared to the basic best improvement strategy [27]. A modified TA and VNS algorithm also used to show the significant effect of the first improvement strategy rather than the basic metaheuristic approach.

2. Mathematical formulation

In this research, many-to-one IRP is defined as a set of period \( t \in \{1, ..., T\} \), supplier \( i \in S = \{1, ..., n\} \), and an assembly plant that has a demands \( d_{it} \) units of product \( i \). There is a capacitated homogenous fleet of vehicles \( z \in M = \{1, ..., m\} \). \( A = \{n + 1\} \) denotes the assembly plant, \( D = \{0\} \) for the depot, and \( x_{ijt} \) as a number of vehicle movements from node \( i \) to \( j \) in period \( t \). In each period, the vehicle goes from the depot to several suppliers to pick up the product. After completing picking products, the vehicle will go to the assembly plant. There are three types of costs in this model, namely, travel cost \( c_{ij} \), inventory cost \( h_i \), and fixed cost \( F \) for using one vehicle in one period. Inventory level product \( i \) in period \( t \) denote \( inv_{it} \). The basic many-to-one multi-product multi-period IRP objective functions are as follows:

\[
\text{minimize} \quad c \left( \sum_{j=1}^{m} \sum_{i \in A} c_{ij} x_{ijt} \right) + \sum_{i \in A} h_i \left( \sum_{t=1}^{T} \sum_{r=1}^{n} x_{it} \right) + \sum_{i \in A} h_i \left( \sum_{t=1}^{T} inv_{it} \right) + \left( F + c_{n+1} \right) \sum_{i \in A} \sum_{t=1}^{T} x_{it} \quad (1)
\]
Dynamic velocity as a modifier of travel cost is an important aspect of TTP case that was introduced by Bonyadi et al. [15]. The travel cost per unit distance in TTP is denoted \( R \) (similar with \( C \) in IRP) as a renting rate.

\[
R \left( \frac{c_{xt_n} \cdot x_{t_n}}{v_{\text{max}} - v_{\text{min}}} + \sum_{i=1}^{n-1} \frac{c_{xt_i} \cdot x_{t_i}}{v_{\text{max}} - v_{\text{min}}} \right)
\]  

In TTP, \( c_{xt_n} \) is the distance from the last node \( n \) to the first node, and \( c_{xt_{i+1}} \) is the distance from the current node to the next node. \( w_{x_n} \) is the total weight in the last node and \( w_{x_i} \) is the total weight in the node \( i \). The maximum vehicle weight is represented by \( W \). \( v_{\text{max}} \) and \( v_{\text{min}} \) denote the maximum and minimum vehicle velocity. The velocity formulation is as follows:

\[
v = \frac{v_{\text{max}} - v_{\text{min}}}{W}
\]  

With this dynamic travel speed (vehicle velocity) formulation, travel costs in the IRP objective function will be changed. The first node in the IRP is a depot, and the last node is the assembly plant. Therefore, the modified travel cost aspect in many-to-one multi-product multi-period IRP objective function are as follows:

\[
C \left( \sum_{i \in S} \sum_{t \in \tau} \frac{c_{xt_i} \cdot x_{t_i}}{v_{\text{max}} - v_{\text{min}}} + \sum_{j \in A} \sum_{t \in \tau} \frac{c_{jt} \cdot x_{jt}}{v_{\text{max}} - v_{\text{min}}} \right)
\]  

The constraint formulations related to vehicle movements, inventory level, and product picked-up quantities in the IRP with dynamic velocity are still the same as the basic many-to-one IRP, but the vehicle capacity will be changed with the weight capacity \( W \). In addition, \( r_{ijt} \) is product weight quantity transported from node \( i \) to \( j \) in period \( t \). \( a_{jt} \) is a total weight amount of product to be picked up at supplier \( i \) in period \( t \).

\[
\sum_{i \in S} x_{ijt} + a_{jt} = \sum_{i \in S} x_{ijt} + a_{ijt} \quad \forall j \in S, \forall t \in \tau
\]  

\[
in_{ijt} = ln_{ijt-1} + a_{ijt} - d_{ijt} \quad \forall i \in S, \forall t \in \tau
\]  

\[
\sum_{i \in S} r_{ijt} + a_{ijt} = \sum_{i \in S} r_{ijt} + a_{ijt} \quad \forall t \in \tau
\]  

\[
r_{ijt} \leq W, \quad \forall i \in S, \forall j \in S \cup A, i \neq j, \forall t \in \tau
\]  

Eq. (5) is a constraint related to product weight quantity transported. Eq. (6) is the total inventory equation related to total product weight and demand. Eq. (7) is the number of product weight picked-up quantities from several suppliers to the assembly plant. Eq. (8) is a constraint related to a maximum number of vehicle weight capacity. Additional constraint related to a number of vehicle departure and arrival are as follows:

\[
\sum_{j \in S} x_{ij} = \sum_{j \in S} x_{ij}, \quad i \in D, k \in A, \forall t \in \tau
\]  

The remaining constraint mathematical formulations are non-negativity and integer variable.
\[ x_{ijt} = 0, \; i \in D, j \cup A, \forall t \in \tau \]  \hspace{1cm} (16)

\[ x_{ijt} = 0, \; i \in A, j \cup S, \forall t \in \tau \]  \hspace{1cm} (17)

\[ x_{ijt} = 0, \; i \in S, j \cup D, \forall t \in \tau \]  \hspace{1cm} (18)

\[ r_{ijt} \geq 0, \; \forall i \in S, \forall j \in S \cup A, \forall t \in \tau \]  \hspace{1cm} (19)

\[ r_{ijt} = 0, \; \forall i \in S, \forall t \in \tau \]  \hspace{1cm} (20)

3. The proposed algorithms

The first modified metaheuristic approach is the threshold accepting with the first improvement strategy (TA-FIS). TA is a metaheuristic approach introduced by Dueck and Scheuer [28]. This algorithm has rules that set to reduce the threshold or the number of solutions accepted at each iteration. A better solution will be obtained by adding some local search processes. The proposed TA-FIS algorithm can be seen in Figure 1.

```
| Algorithm 1 TA-FIS method |
|---------------------------|
| 1: procedure TA-FIS        |
| 2: step (0) initialization: generate initial solution \( x \) using LCI (\( x^{*} \rightarrow x \)), and then set all threshold parameter values |
| 3: step (1) uphill move: do the local search with the first improvement strategy |
| 4:               \hspace{1cm} while \( t \leq t_{\text{max}} \) |
| 5:               \hspace{1cm} one-point-move, if the solution \( x \) better than \( x^{*} \), then \( x^{*} \rightarrow x \), go to the next local search; else \( t++ \) |
| 6:               \hspace{1cm} two-point-move, if the solution \( x \) better than \( x^{*} \), then \( x^{*} \rightarrow x \), go to step (2); else \( t++ \) |
| 7:               \hspace{1cm} step (2) downhill move: do the downhill move local search |
| 8:               \hspace{1cm} one-point-move \rightarrow two-point-move \rightarrow two-opt, if the solution \( x \) better than \( x^{*} \) |
| 9:               \hspace{1cm} then \( x^{*} \rightarrow x \), back to step (2); else go to step (3) |
| 10:              \hspace{1cm} step (3) updating parameter: updating the threshold acceptance |
| 11:             \hspace{1cm} if \( n \geq n_{\text{min}} \) then reduce threshold \( n \), back to step (1); else go to step (4) |
| 12:          \hspace{1cm} step (4) report: the best solution obtained, \( x^{*} \) |
```

Figure 1. TA-FIS metaheuristic algorithm.

Based on Figure 1, there are four steps in TA-FIS algorithm. Step (0) is the initialization step. This step uses the LCI algorithm adapted by Ramadhan et al. [21] (including initialization for all threshold parameter values). The best parameter values (based on several experiments) are starting threshold value \( n = 15000 \), final threshold value \( n_{\text{min}} = 0 \), and the maximum number of uphill iterations \( t_{\text{max}} = 30 \). Step (1) is local search processes that focused on uphill movements. This movement is applied with choosing a node randomly and move to the other route randomly (intra or inter route). See Figure 2 for an example of uphill move processes. In step (1), first improvement local search strategy is applied to expand the solution space. Different from the best improvement strategy, first improvement local search strategy does not need to complete local search until the last iteration, which can be trapped in a local optimal solution. Therefore, in TA-FIS structure, if the current solution \( x \) better than \( x^{*} \), then \( x^{*} \rightarrow x \) and directly go to the next local search. Step (2) is a modified local search process that focused on downhill movements. This movement is used to move the node systematically by almost all possible changes. With these movements, the solution will be better, and it can approach the global optimal solution (see Figure 3). Step (4) is a reducing threshold value \( n \) until \( n_{\text{min}} \). Step (4) is the best solution result \( x^{*} \). The node movements (local searches) in this algorithm use one-point-move 1-0, 1-1 intra-inter route. For two-point-move, 2-0 and 2-2 inter-intra route is used to do the movements. The two-opt intra-inter route also used to get a better result in a downhill move. One-point-move, two-point-move, and two-opt local searches here are adapted from Ramadhan et al. [21].
The second modified metaheuristic approach is the variable neighborhood search with the first improvement strategy (VNS-FIS). VNS was first proposed by Mladenovic & Hansen [29] to solve the combinatorial optimization problem. This approach is based on systematic changes in the neighborhood using local searches. The proposed VNS-FIS algorithm can be seen in Figure 4. This algorithm is almost similar to TA-FIS. The main difference in VNS-FIS is the acceptance solution criteria. VNS-FIS can accept the worse solution than the current best solution as long as it is feasible. This approach can expand the solution space to get a better result in the next iteration.

**Algorithm 2 VNS-FIS method**

1: procedure VNS-FIS
2:  step (0) initialization: generate initial solution \( x \) using LCI \( (x^* \leftarrow x) \), and then set
3:  a maximum number of local and global iteration, \( l_{\text{max}} = 30 \); \( f = 0 \);
4:  \( l_{\text{max}} = 15000 \) (best parameter values based on several experiments)
5:  step (1) uphill move: do the local search with the first improvement strategy
6:  while \( l \leq l_{\text{max}} \)
7:    one-point-move, if the current solution \( x \) is feasible then update the solution
8:    if the current feasible solution \( x \) better than \( x^* \) then \( x^* \leftarrow x \), go to the next local
9:    search; else \( i++ \)
10:  while \( l \leq l_{\text{max}} \)
11:    two-point-move, if the current solution \( x \) is feasible then update the solution
12:    if the current feasible solution \( x \) better than \( x^* \) then \( x^* \leftarrow x \), go to step (2); else
13:      \( i++ \)
14:  step (2) downhill move: do the downhill move the local search
15:    one-point-move \( \rightarrow \) two-point-move \( \rightarrow \) two-opt, if the solution \( x \) better than \( x^* \)
16:    then \( x^* \leftarrow x \), back to step (2); else go to step (3)
17:  step (3) next iteration: updating the increment variable
18:    if \( f \leq l_{\text{max}} \) then \( f++ \), back to step (1); else go to step (4)
19:  step (4) report: the best solution obtained, \( x^* \)

**Figure 4. VNS-FIS metaheuristic algorithm**

The third modified metaheuristic approach is record-to-record travel with the first improvement strategy (RRT-FIS). The RRT algorithm has little in common with the VNS algorithm, where at each iteration it is permissible to accept worse solutions. However, RRT has a tolerance limit of \( \alpha \% \) unlike VNS, which allows acceptance as long as the solution obtained is feasible. The first RRT was introduced by Dueck [23] and developed by Li et al. [24]. The proposed RRT-FIS can be seen in Figure 5.
Algorithm 3 RRT-FIS method

1: procedure RRT-FIS
2:   step (0) initialization: generate initial solution $x$ using LCI ($x^* \leftarrow x$), and then set
3:       a maximum number of global iteration $M_{\text{max}} = 3000$, uphill iteration $K_{\text{max}} = 5$,
4:       downhill iteration $I_{\text{max}} = 30$, and record deviation $\alpha = 1\%$ (best parameter
5:       values based on several experiments)
6:   step (1) uphill move: do the local search with the first improvement strategy
7:       while $K \leq K_{\text{max}}$
8:           one-point-move, if the solution $x \leq (1 + \alpha)x^*$ then update the solution
9:           if the current solution $x$ better than $x^*$ then $x^* \leftarrow x$, go to the next local search;
10:          else $K++$
11:          set $K = 0$
12:       end
13:       while $I \leq I_{\text{max}}$
14:          one-point-move $\rightarrow$ two-point-move $\rightarrow$ two-opt, if the solution $x$ better than $x^*$
15:          then $x^* \leftarrow x$, back to step (2); else $I++$
16:       end
17:   step (2) downhill move: do the downhill move local search
18:       while $K \leq K_{\text{max}}$
19:          two-point-move, if the solution $x \leq (1 + \alpha)x^*$ then update the solution
20:          if the current solution $x$ better than $x^*$ then $x^* \leftarrow x$, go to step (2); else $K++$
21:          set $K = 0$
22:       end
23:   step (3) global iteration: updating the global iteration
24:       if $M \leq M_{\text{max}}$ then $M++$, back to step (1); else go to step (4)
25:   step (4) report: the best solution obtained, $x^*$

Based on Figure 5, the main step of RRT-FIS similar to TA-FIS and VNS-FIS, but the detail processes for each main step is different. Different from VNS-FIS, RRT-FIS can accept a worse solution with threshold criteria. If the solution $x \leq (1 + \alpha)x^*$ then this current solution is acceptable. Besides being able to expand $\alpha\%$ solution space, RRT-FIS concept should get the result faster than VNS-FIS who search in unlimited solution space. Because the computational time is still faster, then this structure can be added more iteration in the downhill move. This step can do more downhill move to get better solution and near with global optimal solution. There are several parameter values in the RRT algorithm, such as a maximum number of global iterations, a number of uphill and downhill iterations, and record deviation. This research does several experiments to get the best parameter values that can produce good and competitive results.

4. Computational results and discussions

A 3.4GHz with 8GB RAM computer is used to run the C++ coded algorithm (Code::Block). First, the proposed algorithm will be implemented in the case of basic many-to-one IRP, because the datasets for IRP with dynamic velocity has not been found in previous researches. The small datasets many-to-one IRP from Moin et al. [7] is used to test the quality of three modified algorithms (TA-FIS, VNS-FIS, and RRT-FIS). The result and computational time (CPU time in seconds) by the proposed algorithms are compared with the best-known solution by Mjirda et al. [13]. The results are presented in Table 1 (the proposed algorithm results are the best solution from ten runs).

Based on Table 1, the proposed algorithms have good results. The average deviation from the best-proposed algorithm is only 1.86%. Therefore, this algorithm can be used for more complex inventory routing like IRP with dynamic velocity. Because the dataset for IRP with dynamic velocity has not been found in previous researches, so this paper creates a new dataset based on the many-to-one IRP datasets by Moin et al. [7]. Quantity and capacity in many-to-one IRP datasets become product weight and vehicle weight capacity. Different suppliers have different weight products, but products in the same supplier have the same weight. Depot, supplier, and assembly plant coordinates are still the same. In addition, this paper adapts $v_{\text{max}}$ and $v_{\text{min}}$ values from numerical examples in Polyakovskiy et al. [16] accompanied by several experiments to see the relationship between velocity, weight capacity, and total cost. The results of comparisons and experiments between proposed algorithms for IRP with dynamic speed can be seen in Table 2.
Based on Table 2, RRT-FIS is slightly better than VNS-FIS (only 5% different in the average deviation), but both algorithms have good results. They can be used to solve IRP with dynamic velocity compared to TA-FIS. Almost all datasets completed by RRT-FIS and VNS-FIS can get the best solution with fast computational time. The difference in velocity and weight capacity for each dataset shows that longer velocity ranges and larger weight capacity can make the total cost smaller. There must be more experiments to see the correlation between parameter values and all costs in the IRP.

Based on the computational results, two modified proposed algorithms (RRT-FIS and VNS-FIS) can be used as an approach to solve the IRP model. RRT-FIS shows domination in computational times (less than 60 seconds) to get the best solution, and this algorithm can be used as a priority approach to solving the IRP. Although TA-FIS algorithm is not better than two other algorithms, TA-FIS can be

### Table 1. Comparison of the proposed algorithm solutions.

| Datasets  | BKS* | TA-FIS | VNS-FIS | RRT-FIS |
|-----------|------|--------|---------|---------|
|           | Total cost | Total cost | CPU (s) | Total cost | Total cost | CPU (s) |
| S12T5     | 1961.71 | 2096.75 | 97 | 1983.42 | 61 | 1961.71 | 16 |
| S12T10    | 4002.85 | 4002.85 | 103 | 4002.85 | 68 | 4002.85 | 22 |
| S12T14    | 5635.77 | 5635.77 | 98 | 5635.77 | 59 |
| S20T5     | 2861.26 | 3121.12 | 100 | 3012.41 | 72 | 3002.39 | 23 |
| S20T10    | 5944.72 | 6242.24 | 114 | 6242.24 | 93 | 6242.24 | 38 |

Average deviation**: 4.98% 2.28% 1.86%

*BKS* = best known solution by Mjirda et al. [13]
**Average deviation = (proposed algorithm – BKS) / BKS x 100%

### Table 2. Comparison and experimental results for IRP with dynamic velocity.

| Datasets  | v/W  | Best Results** | TA-FIS | VNS-FIS | RRT-FIS |
|-----------|------|----------------|--------|---------|---------|
|           | Total cost | Total cost | CPU (s) | Total cost | Total cost | CPU (s) |
| S12T5     | (0.10, 0.40) | 3622.83 | 28.36 | 3622.83 | 22.42 | 3622.83 | 11.03 |
|           | 5     | 3622.83 | 28.36 | 3622.83 | 22.42 | 3622.83 | 11.03 |
|           | 10    | 1995.54 | 32.32 | 2014.30 | 26.86 | 1995.54 | 13.43 |
|           | 15    | 1645.29 | 31.88 | 1645.29 | 29.50 | 1649.54 | 14.50 |
|           | 5     | 3630.99 | 29.47 | 3630.99 | 22.21 | 3630.99 | 12.92 |
|           | m-S12T5* | 3630.99 | 29.47 | 3630.99 | 22.21 | 3630.99 | 12.92 |
|           | (0.10, 0.70) | 2034.51 | 30.58 | 2034.51 | 26.52 | 2056.08 | 13.22 |
|           | 15    | 1646.68 | 31.83 | 1646.68 | 29.20 | 1646.68 | 14.50 |
|           | 5     | 3630.99 | 29.55 | 3630.99 | 22.35 | 3630.99 | 11.17 |
|           | m-S12T10* | 3630.99 | 29.55 | 3630.99 | 22.35 | 3630.99 | 11.17 |
|           | (0.10, 1.00) | 2030.59 | 30.54 | 2030.59 | 29.57 | 2030.59 | 13.46 |
|           | 15    | 1692.65 | 31.79 | 1692.65 | 29.13 | 1692.65 | 14.39 |
|           | 5     | 7659.12 | 57.68 | 7659.12 | 42.23 | 7659.12 | 20.55 |
|           | m-S12T10* | 7659.12 | 57.68 | 7659.12 | 42.23 | 7659.12 | 20.55 |
|           | (0.10, 0.40) | 4174.64 | 60.60 | 4174.64 | 52.00 | 4199.24 | 25.51 |
|           | 15    | 3332.80 | 62.63 | 3343.40 | 56.36 | 3332.80 | 28.35 |
|           | 5     | 7641.30 | 57.30 | 7857.04 | 42.08 | 7857.04 | 20.11 |
|           | m-S12T10* | 7641.30 | 57.30 | 7857.04 | 42.08 | 7857.04 | 20.11 |
|           | (0.10, 1.00) | 4295.12 | 59.79 | 4435.29 | 53.11 | 4295.12 | 25.18 |
|           | 15    | 3331.69 | 62.83 | 3331.69 | 56.66 | 3331.69 | 23.04 |
|           | 5     | 7524.21 | 57.79 | 7818.41 | 55.00 | 7818.41 | 21.13 |
|           | m-S12T14* | 7524.21 | 57.79 | 7818.41 | 55.00 | 7818.41 | 21.13 |
|           | (0.10, 0.40) | 4631.88 | 59.41 | 4631.88 | 54.71 | 4631.88 | 21.11 |
|           | 15    | 3427.16 | 62.65 | 3759.40 | 57.73 | 3427.16 | 24.21 |
|           | 5     | 10776.00 | 79.20 | 10776.00 | 76.12 | 10981.21 | 40.02 |
|           | m-S12T14* | 10776.00 | 79.20 | 10776.00 | 76.12 | 10981.21 | 40.02 |
|           | (0.10, 0.40) | 6162.86 | 83.80 | 6230.31 | 80.09 | 6162.86 | 41.67 |
|           | 15    | 5237.21 | 85.55 | 5237.21 | 44.02 |
|           | 5     | 11021.08 | 89.11 | 11021.08 | 86.31 | 11021.08 | 54.22 |
|           | m-S12T14* | 11021.08 | 89.11 | 11021.08 | 86.31 | 11021.08 | 54.22 |
|           | (0.10, 1.00) | 6663.18 | 85.55 | 6663.18 | 55.19 |
|           | 15    | 4921.41 | 89.31 | 5001.48 | 56.93 |
|           | 5     | 11143.90 | 85.05 | 11143.90 | 56.93 |
|           | m-S12T14* | 11143.90 | 85.05 | 11143.90 | 56.93 |
|           | (0.10, 1.00) | 6663.18 | 85.55 | 6663.18 | 55.19 |
|           | 15    | 4921.41 | 89.31 | 5001.48 | 56.93 |

Average deviation**: 3.39% 0.89% 0.39%

*m-S12T = modified S12T; m-S12T10 = modified S12T10; m-S12T14 = modified S12T14
**Best Results = best results between TA-FIS, VNS-FIS, or RRT-FIS for each dataset

doi:10.1088/1742-6596/1501/1/012009
developed further to get the best result and can be an alternative approach to solve many-to-one IRP with dynamic velocity model. An example of different best solution route from basic many-to-one IRP and IRP with dynamic velocity can be seen in Figure 6. The 3D graph illustration based on the relationships between $t_{\text{max}}$, $W$, and the total cost can be seen in Figure 7.

Figure 6. An example of different best solution route.

Figure 7. The 3D graph illustration based on three experimental parameters.

5. Conclusions
This paper develops the many-to-one IRP to a many-to-one IRP with dynamic velocity model. Dynamic velocity in the travelling thief problem model is adapted to produce an IRP model that is similar to real conditions. To complete this model, three modifications of the metaheuristic approach (threshold accepting, variable neighborhood search, and record-to-record travel algorithms) were developed. In addition to the process of modifying the three approaches, the first improvement local search strategy algorithm is integrated to get better solutions. This research shows that the three modified algorithms produce a good solution compared to previous researches (the best average deviation of the proposed algorithm is only 1.86%) with fast computational time to solve IRP with dynamic velocity model (less than 60 seconds). The result of this research proves that the algorithms can be used to solve new, more complex IRP models such as IRP with dynamic velocity.

For further development of this research, several local search procedures that have not yet been implemented can be added, such as OR-opt, cross tail movements, or perturbation based local search on improving existing solutions. Determination of parameter values related to the number of global iterations or uphill and downhill processes can be done further to get the best parameter values.

6. References
[1] W. J. Bell et al., “Improving the distribution of industrial gases with an on-line computerized routing and scheduling optimizer,” Interfaces (Providence), vol. 13, no. 6, pp. 4–23, 1983.
[2] R. Ribeiro and H. Lourenço, “Inventory-routing model, for a multi-period problem with stochastic and deterministic demand,” 2003.

[3] O. Solyali and H. Süral, “A branch-and-cut algorithm using a strong formulation and an a priori tour-based heuristic for an inventory-routing Problem,” *Transp. Sci.*, vol. 45, no. 3, pp. 335–345, 2011.

[4] M. J. Geiger and M. Sevaux, "On the use of reference points for the bi-objective inventory routing problem," in *Metaheuristics International Conference*, 2011, pp. 1–9.

[5] S. Sindluchao, H. E. Romeijn, E. Akçali, and R. Boondiskulchok, “An integrated inventory-routing system for multi-item joint replenishment with limited vehicle capacity,” *J. Glob. Optim.*, vol. 32, no. 1, pp. 93–118, 2005.

[6] C. Archetti, L. Bertazzi, G. Laporte, and M. Speranza, “A branch-and-cut algorithm for a vendor managed inventory routing problem,” *Transp. Sci.*, vol. 41, no. 3, pp. 382–391, 2007.

[7] N. H. Moin, S. Salhi, and N. A. B. Aziz, “An efficient hybrid genetic algorithm for the multi-product multi-period inventory routing problem,” *Int. J. Prod. Econ.*, vol. 133, no. 1, pp. 334–343, 2011.

[8] O. Solyali, J. F. Cordeau, and G. Laporte, “Robust inventory routing under demand uncertainty,” *Transp. Sci.*, vol. 46, no. 3, pp. 327–340, 2012.

[9] L. Bertazzi, A. Bosco, F. Guerriero, and D. Laganà, “A stochastic inventory routing problem with stock-out,” *Transp. Res. Part C, Emerg. Technol.*, vol. 27, no. 1, pp. 89–107, 2013.

[10] L. C. Coelho, J. F. Cordeau, and G. Laporte, “The inventory-routing problem with transhipment,” *Comput. Oper. Res.*, vol. 39, no. 11, pp. 2537–2548, 2012.

[11] B. S. Onnogo, J. Panadero, C. G. Corlu, and A. A. Juan, “Agri-food supply chains with stochastic demands: A multi-period inventory routing problem with perishable products,” *Simul. Model. Pract. Theory*, vol. 97, p. 101970, 2019.

[12] L. C. Coelho, J. F. Cordeau, and G. Laporte, “Thirty years of inventory routing,” *Transp. Sci.*, vol. 48, no. 1, pp. 1–19, 2013.

[13] A. Mjirda, B. Jarboui, R. Macedo, and S. Hanafi, “A variable neighborhood search for the multi-product inventory routing problem,” *Electron. Notes Discret. Math.*, vol. 39, pp. 91–98, 2012.

[14] F. Ramadhan, A. Imran, and A. F. Rizana, “A record-to-record travel algorithm for multi-product and multi-period inventory routing problem,” in *2018 5th International Conference on Industrial Engineering and Applications, ICIEA 2018*, 2018, pp. 274–278.

[15] M. R. Bonyadi, Z. Michalewicz, and L. Barone, “The travelling thief problem: The first step in the transition from theoretical problems to realistic problems,” in *2013 IEEE Congress on Evolutionary Computation, CEC 2013*, 2013, no. July 2014, pp. 1037–1044.

[16] S. Polyakovskiy, M. R. Bonyadi, M. Wagner, Z. Michalewicz, and F. Neumann, “A comprehensive benchmark set and heuristics for the travelling thief problem,” *GECCO 2014 - Proc. 2014 Genet. Evol. Comput. Conf.*, pp. 477–484, 2014.

[17] J. Wu, M. Wagner, S. Polyakovskiy, and F. Neumann, “Exact approaches for the travelling thief problem,” in *SEAL*, 2017, pp. 110–112.

[18] J. Wu, M. Wagner, S. Polyakovskiy, and F. Neumann, “Evolutionary computation plus dynamic programming for the bi-objective travelling thief problem,” in *Genetic and Evolutionary Computation Conference*, 2018, pp. 777–784.

[19] N. Xiao and Y. L. Rao, “Multi-product multi-period inventory routing optimization with time window constraints,” *Int. J. Simul. Model.*, vol. 15, no. 2, pp. 352–364, 2016.

[20] Y. Ghiami, E. Demir, T. Van Woensel, M. Christiansen, and G. Laporte, “A deteriorating inventory routing problem for an inland liquefied natural gas distribution network,” *Transp. Res. Part B*, vol. 126, pp. 45–67, 2019.

[21] F. Ramadhan, A. Imran, and A. F. Rizana, “Threshold based algorithms for the multi-product multi-period inventory routing problem,” *J. Eng. Sci. Technol.*, vol. 14, no. 5, pp. 2703–2715, 2019.

[22] P. Karakostas, A. Sifaleras, and M. C. Georgiadi, “A general variable neighborhood search-based solution approach for the location-inventory-routing problem with distribution outsourcing,” *Comput. Chem. Eng.*, vol. 126, pp. 263–279, 2019.
[23] G. Dueck, “New optimization heuristics: The great deluge algorithm and record-to-record travel,” 
J. Com, vol. 104, pp. 86–92, 1993.
[24] F. Li, B. Golden, and E. Wasil, “A record-to-record travel algorithm for solving the heterogeneous fleet vehicle routing problem,” Comput. Oper. Res., vol. 34, no. 9, pp. 2734–2742, 2007.
[25] C. D. Tarantilis, C. T. Kiranoudis, and V. S. Vassiliadis, “A list based threshold accepting metaheuristic for the heterogeneous fixed fleet vehicle routing problem,” J. Oper. Res. Soc., vol. 54, pp. 65–71, 2003.
[26] C. D. Tarantilis, C. T. Kiranoudis, and V. S. Vassiliadis, “A threshold accepting metaheuristic for the heterogeneous fixed fleet vehicle routing problem,” Eur. J. Oper. Res., vol. 152, no. 1, pp. 148–158, 2004.
[27] N. Mladenović, A. Alkandari, J. Pei, R. Todosijević, and P. M. Pardalos, “Less is more approach: Basic variable neighborhood search for the obnoxious p-median problem,” Int. Trans. Oper. Res., vol. 00, pp. 1–14, 2019.
[28] G. Dueck and T. Scheurer, “Threshold accepting: A general purpose optimization algorithm appearing superior to simulated annealing,” J. Comput. Phys., vol. 90, pp. 161–175, 1990.
[29] N. Mladenović and P. Hansen, “Variable neighborhood search,” Comput. Oper. Res., vol. 24, no. 11, pp. 1097–1100, 1997.