Mobile Robot Scheduling with Multiple Trips and Time Windows

Shudong Liu (✉), Huayu Wu, Shili Xiang, and Xiaoli Li

Institute for Infocomm Research, A*STAR
1 Fusionopolis Way, Singapore
{liush, wuhu, sxiang, xlli}@i2r.a-star.edu.sg

Abstract. We consider a vehicle routing problem with multiple trips and time windows (VRPMTTW) in which a mobile robot transports materials from a central warehouse to multiple demanding places. The robot needs to strictly satisfy the time windows at demanding places and it can run multiple trips. How to effectively scheduling the robot is a key problem in operations of Smart Nations and intelligent automated manufacturing. In the literature three-index mixed integer programming models are developed. However, these three-index models are difficult to solve in reasonable time for real problems due to computational complexity of integer programming. We propose an innovative two-index mixed integer programming model. The numerical results show our model can successfully obtain optimal solutions fast for cases where the existing literature has not found the optimal solution yet. To our best knowledge, it is the first two-index model for this type of problems.

Keywords: Vehicle routing problem · Combinatorial optimization · Time window · Multiple trip · Two-index model

1 Introduction

In this work we consider a vehicle routing problem with multiple trips and time windows (VRPMTTW) in which vehicles or mobile robots can perform multiple trips/routes to satisfy customers' demands. Each demand of each customer has its own time window such that a vehicle needs to arrive in this window. This type of problems will become increasingly important in the contexts such as Smart Nations and Factory of Future. It is obvious that in smart nations and factories in the future, intelligent autonomous vehicles and mobile robots will be widely used to transport people, goods and materials. Effective scheduling of these mobile robots and vehicles is critical for implementing Smart Nations and intelligent manufacturing.

With the fast development of technologies, e.g. in artificial intelligence and computational power and communication, intelligent mobile robots/vehicles are becoming more and more matured, and more and more nations and companies are adopting them. Among other countries, Singapore is carrying out trials for self-driving vehicle technologies and mobility concepts along the many kilometer test route; The Centre for Healthcare Assistive and Robotics Technology

(CHART) at Changi General Hospital in Singapore is working with academia, industry and research institutions to develop healthcare solutions leveraging on mobile robots and assistive technology. In industries, some companies in Singapore have already used mobile robots to transport materials in the factories. Amazon is also a pioneer in adopting mobile robots in its warehouse.

A common characteristic of both intelligent vehicles and mobile robots is vehicle routing. In some applications, the time window is very important. For example, in the case of mobile robots transporting material to feeders of production lines, the robots need to strict respect to the time windows of feeders, otherwise the production lines may have to stop or generate wasted products due to missing some materials. During our engagement with industries, we encountered a very similar situation as Dang et al. [6]: a robot transports materials to multiple feeders of production lines. Another example is the transportation of materials such as operational tools in hospitals where the time windows are also very important. In addition, in modern city logistics, vehicles often need to satisfy customers' time windows, though they may not be so strict as in production and healthcare situations.

In order to save cost, it is natural that a vehicle/mobile robot may perform multiple trips only if it can satisfy the requirement of time windows. So, vehicle routing problems with multiple trips and time windows (VRPMTTW) arise in many real-life contexts and will become more and more important. In the classical literature for vehicle routing problem with time window (VRPTW), it is in general assumed that each vehicle runs only one trip/route. In the problem VRPMTTW, we need to explicitly consider the precedence of trips when a vehicle runs multiple trips, which makes the problem more complicated.

In spite of the importance of VRPMTTW in practice, currently there is no much literature. This problem is NP-hard and more complex than the traditional vehicle routing problems. Some researchers [6] developed three-index mixed integer programming (MIP) models which have huge number of integer variables and exponential-size of constraints. They are very difficult to solve in reasonable time for real problems. Hence some researchers developed heuristics for it. We develop an innovative model which uses two-index integer variables and has linear-size of constraints. In this way, both the searching space and computational time are significantly reduced. To our best knowledge, it is the first two-index model for this type of problems in the literature.

The remainder of this paper is organized as follows. The literature review is given in Section 2. In Section 3, the problem is described and in Section 4 we present the mathematical programming model. The numerical results are reported in Section 5. Finally, concluding remarks follow in Section 6.

2 Literature Review

Vehicle routing problem was first studied by George Dantzig and John Ramser [7] in 1959. Since then many researchers have studied it and a variety of variants and made many significant progresses in developing heuristics and exact algo-

rithms. Some important variants are Vehicle Routing Rroblem with Backhaul (VRPB), Vehicle Routing Problem with Time window (VRPTW), Pickup and Delivery Problem (VRP) and Dial-a-ride Problem. There are a few excellent comprehensive literature review, refer to [9], [15] and [4].

Most papers in the above literature assume one vehicle performs only one route/trip. However, in practice people often use one vehicle for multiple trips to save cost. Nevertheless, only a few consider the Vehicle Routing Problem with Multiple Trip (VRPMT). Fleischmann [8] first considered this problem in 1990. He modified the well-known saving algorithm and used a bin-packing heuristic to assign routes to vehicles. Since then more people developed some heuristics using genetic algorithms, tabu search and so on (e.g. [14], [5], [13], [12]). Mingozzi et al. [10] developed an exact algorithm for VRPMT using set-partition like formulations, which heavily depends on the initial upper bound.

The literature for vehicle routing problem considering both multiple trip and time window (VRPMTTW) is scarce. Azi et al. [1] first addressed this type of problem in 2007. They developed an exact method for the single vehicle case. This method consists of two phases: in the first phase, all non-dominated feasible routes are generated; in the second phase, some trips are selected and sequenced to form the vehicle workday. In 2010 Azi et al. [2] extended their previous work [1] to multiple vehicles cases. Note that in both papers, the vehicle is not forced to visit all customers, but serve them as many as possible to maximize profit. In addition, there is a maximum duration in each route. These two aspects are different from the problem considered in this paper. Battarra et al. [3] developed a heuristic for it. They decomposed the problem into two easier problems, and developed two heuristics for them: the first heuristic is to create routes and the second is for a bin packing problem to connect routes.

Dang et al.[6] considered VRPMTTW in 2014 in which a single mobile robot was used for transporting parts from a warehouse to a few feeders of machines. The time windows are hard constraints, i.e. needing to respect them strictly. They developed a three-index integer programming model with exponential-size constraints. As solving this model is very time consuming as shown in their numerical study, they developed a genetic algorithm for it. Nielsen et al. [11] extended the work of [6]. As mentioned in [11] , genetic algorithms may not be able to find feasible solutions in such complex situations, hence they considered the soft time windows, i.e. allowing delay in delivery, and the objective was changed into weighted sum of traveling distance and delay.

In this paper, we consider the same problem as [6] in which a single robot transports materials/parts from a warehouse to feeders. We propose an innovative two-index integer programming model. This model has far less integer variables and constraints, compared with the three-index model in [6]. Hence it is much easier to solve using general purpose solvers as shown in Section 5.

3 Problem Description

During our industrial engagement, we encountered very similar problems as Dang et al. [6]. For simplicity, we describe the problem in [6].

In a flexible manufacturing system, there are a few production lines and each line has multiple machines. During production, machines consume materials/components and transform them into the final products. Later we just use material to represent both material and components when it is clear in the context. Some machines have its own feeders so that people or robots can feed material to it. Material is stored in a central warehouse. During the production, robots need to transport material from warehouse to the places near machines, and then put them into the feeders. Putting materials into feeders may be done by robots or by people, which depends on the capability of robots, but transporting material from warehouse to feeders will be done only by mobile robots. An example of layout of flexible manufacturing systems is shown in Figure 1, which is adapted from [6].

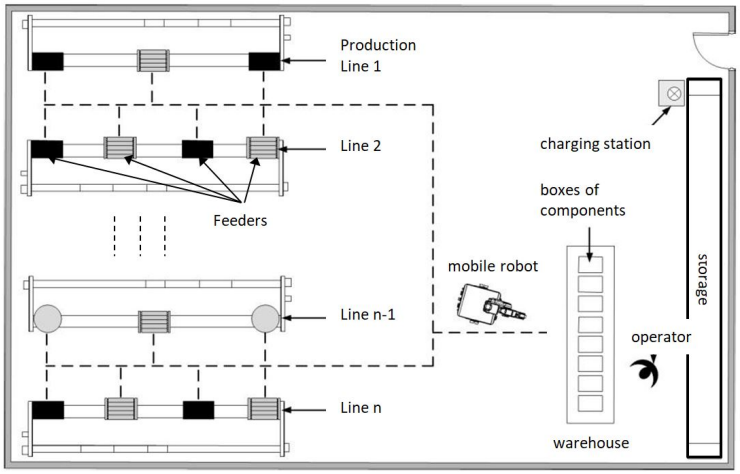

Fig. 1. Layout of a flexible manufacturing system with mobile robots

From the figure, we can see that the mobile robot transports materials from warehouse to feeders. Each robot has a carry capacity. It is possible for a robot to carry materials for multiple feeders in one trip (if the robot capacity allows it) and then come back to warehouse to start a new trip. The capacity of robots can be measured by boxes as in [6] where the boxes are called Small Load Carrier (SLC) and each SLC includes multiple components.

The times of feeding materials to feeders are controlled by a (s, Q) inventory policy. Each machine in general runs at a constant speed, i.e. the consumption rate of materials is constant. The (s, Q) policy is shown in Figure 2 which shows

how the inventory (material/components) at a feeder changes with time. At the beginning, there is an initial inventory. The inventory decreases with the time. When the inventory reaches the threshold level s, the system sends a signal to ask for replenishing inventory. The replenishing amount is a fixed value Q. The replenishment must be done before running out of stock. Hence for each replenishment, there is a time window. This time window is a hard constraint, because if replenishment starts too early, i.e. before the release time, the inventory will be above the maximal level $(s + Q)$ which may be beyond the physical space of feeders. If the replenishment is too late, i.e. after the due time, then the machine has to stop or generate bad products due to missing materials. As each machine of production lines runs at its constant speed, the inventory decrease at a constant rate. Due to the constant machine speed, the release times of two consequential replenishment has a fixed gap, i.e. the period length p. Note that different machines can have different speeds/material consumption rates and different feeders can have different s and Q.

A robot may carry materials for multiple feeders in one trip (from departure from warehouse to coming back to warehouse). This should be more efficient than serving only one feeder in one trip. After coming back to warehouse, the robot can load materials to serve later demands at feeders again. How to schedule robots to satisfy demands at feeders with multiple trips and hard time windows is a critical problem to implement automated intelligent flexible manufacturing. As in [6], we consider the single robot case for multiple feeders.

The problem addressed in this paper is quite typical in practice. There are other scenarios. For example, at the end of production lines in the same factory, the finished products are needed to send back to product warehouse. At the end of each production line, there is a limited buffer area for products. An intelligent forklift (no driver) is used to transport products to warehouse. The forklift also has a capacity limit and may serve multiple lines at the same time. The problem of scheduling of the forklift is almost the same as the problem addressed in this paper.

4 Mathematical Formulation

In this section we present a two-index mixed integer programming model for scheduling the mobile robot with the objective of minimizing total traveling distance/time.

4.1 Assumptions and Notations

Assume there are n_f feeders. Let $V = \{1, ..., n_f\}$ be the set of feeders, and $V^+ = \{0\} \cup V$, where 0 denote the warehouse. A single mobile robot of capacity Q_m transports materials. Each element (feeder or warehouse) in V^+ has its own location. A distance d_{ij} and traveling time t_{ij} is associated with each pair of locations (i, j) where $i, j \in V^+$. Each feeder requests material replenishment by its own (s, Q) policy. Assume material consumption rates, initial inventory,

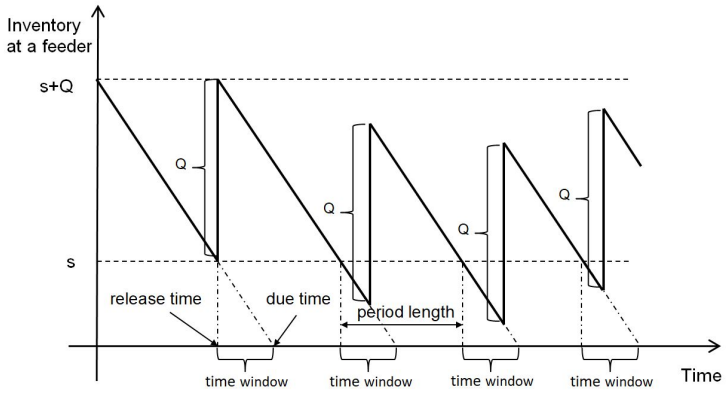

Fig. 2. (s,Q) inventory policy for feeding materials at feeders

parameters (s and Q) of inventory policies are known. Hence the time window (from release time to due time) of each request at each feeder is given. At each location, there is a working time. At the warehouse, it is the time of loading material up to robot, while at feeders it is the unloading time from robot to feeders. These working times are constants.

Assume a single delivery/stop at a feeder can fulfill one replenishment of this feeder. When $Q_m = n$, the robot can carry materials to serve n feeders in a single trip. Assume at the beginning of planning horizon, the robot is at the warehouse.

Material replenishment at feeders and loading material at warehouse are regarded as different task types. Let $A_{type} = \{0, 1, ..., n_f\}$ denote the set of task types. Type $i \in V^+$ means the tasks at location i, e.g. type 0 is tasks of loading material to robot at the warehouse (plus possible unloading empty boxes from previous trip). Each task type can repeat many times. So, each task is identified by its type and the index in its own type. Following are more notations:

$A_{type} = \{0, 1, ..., n_f\}$, set of task types for production.

T, planning horizon.

$c_a, a \in A_{type} \backslash \{0\}$, material consumption rate at feeder a.

$N_A^a, a \in A_{type}$, number of times of repeating for task type a. $N_A^a, a \in A_{type} \backslash \{0\}$, is defined as $N_A^a = \lfloor T/c_a \rfloor$. N_A^0 is the number of tasks at warehouse, for which we will derive an upper and lower bound and will set a value for it.

$N_A = \sum_{a \in A_{type}} N_A^a$, total number of tasks at all places.

$N_A^- = N_A - N_A^0$, total number of tasks at feeders.

$I_A^a = \{1, ..., N_A^a\}, a \in A_{type}$, set of index of tasks of type a.

$I_A = \{1, ..., N_A\}$, set of index of tasks.

$(a, i), a \in A_{type}, i \in I_A^a$, the task which is the i_{th} task in the group of type a. So, each task is identified by two indexes: a and i.

$A^a = \{(a, i) | a \in A_{type}, i \in I_A^a\}$, set of tasks of type a.

$A = \underset{a}{\cup} A^a$, set of all tasks of all types.

$Pre = \{(a1, i1, a2, i2)\}$, $(a1, i1), (a2, i2) \in A$, set of precedence relation among tasks, which means task $(a2, i2)$ cannot start until task $(a1, i1)$ is finished.

For each task, there is a time window (from release time to due time) such that the task must be finished in this window. Given material consumption rates, inventory policies and initial state of inventories, we can calculate the time windows of tasks at each feeder. For each task, there is a working/processing time, e.g. loading and unloading time. Let

$TR_{a,i}$, release time of task (a, i).

$TD_{a,i}$, due time (deadline) of task (a, i).

TP_a, working time of task type a by the robot.

$TC_{a1,a2}$, $a1, a2 \in A_{type}$, traveling time from one place (after having finished task $a1$) to another place to start task $a2$.

M, a big number used for modeling, which is larger than any time (expressed in real number) in the system.

Q_m, maximum number of stops at feeders in a single trip, i.e. maximum number of feeders that the robot can carry material and serve in a trip due to its capacity.

We have following decision variables:

$x_{a1,i1}^{a2,i2} \in \{0, 1\}$, $(a1, i1), (a2, i2) \in A$. It is 1 if the robot is assigned to do the task $(a1, i1)$, followed by task $(a2, i2)$, 0 otherwise.

$s_{a,i}, (a, i) \in A$, starting time of task (a, i). $z_k^{a,i} \in \{0, 1\}$, $(a, i) \in A, k \in \{1, ..., N_A\}$. It is 1 if the task (a, i) is assigned to do as the k_{th} task in the all tasks, 0 otherwise.

Note that the above $x_{a1,i1}^{a2,i2}$ are two-index integer variables which shows the relation between one task (an index) to another task (an index). In the literature, people developed models using three-index variables $x_{a1,i1}^{a2,i2,k}$ where k is the index of trips. The number of integer variables in three-index formulation is many times of ours.

4.2 The Two-Index Mixed Integer Programming Model

As mentioned before, we need to set a value for number of tasks at warehouse, i.e. N_A^0. As there are N_A^- tasks at feeders, there are at most N_A^- trips to finish these tasks based on the assumption that a single delivery/stop at a feeder can fulfill one replenishment of this feeder, i.e. $Q_m \geq 1$. So, the number of tasks at warehouse $N_A^0 \leq N_A^- + 1$, where the last "1" is for returning back to warehouse after finishing all tasks at feeders. On the other hand, the robot also needs at least $\lceil N_A^-/Q_m \rceil$ trips by assuming each trip the robot can server Q_m feeders, hence $N_A^0 \geq \lceil N_A^-/Q_m \rceil + 1$. Again the last "1" is for returning back to warehouse at the end. There are two extreme cases. The real number of trips needed should be some value in the middle of these two extreme points. When we assign a smaller and feasible value to N_A^0, the computation time will be shorter because there will be smaller number of variables and constraints in the model. For safety, we set the maximal value, i.e. $N_A^0 = N_A^- + 1$, which means we add $N_A^- + 1$ tasks

of type 0 to tasks at feeders. So, the total number of tasks $N_A = 2N_A^- + 1$. The robot will execute these tasks following a sequence. We will assign these tasks onto a sequence of length N_A.

Each trip starts from warehouse and ends at warehouse. The tasks from task $(0, i), i \in \{1, ..., N_A^0 - 1\}$ to task $(0, i+1)$ belong to the i_{th} trip. If there is no other tasks between $(0, i)$ and $(0, i+1)$, then this trip is empty, i.e. the robot does not server any feeders. Task $(0, i+1)$ is the end of i_{th}trip and also the beginning of $(i+1)_{th}$ trip. If task$(0, i)$ is executed as the m_{th}task in all tasks, and task $(0, i+1)$ is executed as the n_{th} task in all tasks, then $(n - m - 1)$ is the number of feeders served in the i_{th} trip. Due to robot's capacity, we have the constraint that $(n - m - 1) \leq Q_m$. In this way, we enforce the precedence relation between trips and the robot capacity limit.

We also need other constraints such as satisfying time windows. The objective is to minimize total traveling distance. The new compact two-index MIP model is as follows:

$$min \sum_{(a1,i1)\in A} \sum_{(a2,i2)\in A} TC_{a1,a2} \cdot x_{a1,i1}^{a2,i2} \tag{1}$$

$$s.t. \sum_{(a2,i2)\in A} x_{a1,i1}^{a2,i2} = 1, \ \forall (a1, i1) \in A \backslash \{(0, N_A^- + 1)\} \tag{2}$$

$$\sum_{(a2,i2)\in A} x_{a1,i1}^{a2,i2} = 0, \ \forall (a1, i1) \in \{(0, N_A^- + 1)\} \tag{3}$$

$$\sum_{(a1,i1)\in A} x_{a1,i1}^{a2,i2} = 1, \ \forall (a2, i2) \in A \backslash \{(0, 1)\} \tag{4}$$

$$\sum_{(a1,i1)\in A} x_{a1,i1}^{a2,i2} = 0, \ \forall (a2, i2) \in \{(0, 1)\} \tag{5}$$

$$x_{a1,i1}^{a1,i1} = 0, \ \forall (a1, i1) \in A \tag{6}$$

$$TR_{a,i} \leq s_{a,i}, \ \forall (a, i) \in A \tag{7}$$

$$s_{a,i} + TP_a \leq TD_{a,i}, \ \forall (a, i) \in A \tag{8}$$

$$s_{a1,i1} + TP_a + TC_{a1,a2} x_{a1,i1}^{a2,i2} - (1 - x_{a1,i1}^{a2,i2}) M \leq s_{a2,i2}, \\ \forall (a1, i1) \in A, (a2, i2) \in A \text{ such that } (a1, i1) \neq (a2, i2) \tag{9}$$

$$\sum_{k\in I_A} z_k^{a,i} = 1, \ \forall (a, i) \in A \tag{10}$$

$$\sum_{(a,i)\in A} z_k^{a,i} = 1, \ \forall k \in I_A \tag{11}$$

$$z_1^{0,1} = 1 \tag{12}$$

$$z_1^{0,i} = 0, \ \forall i \in \{2..N_A^0\} \tag{13}$$

$$z_{N_A}^{0,N_A^0} = 1 \tag{14}$$

$$z_{N_A}^{0,i} = 0, \ \forall i \in \{1..N_A^0 - 1\} \tag{15}$$

$$\sum_{k\in I_A}(k\cdot z_k^{a,i})-\sum_{k\in I_A}(k\cdot z_k^{a,i-1})\le Q+1,\ \forall(a,i)\in A^0\backslash\{(0,1)\} \tag{16}$$

$$\sum_{k\in I_A}(k\cdot z_k^{a,i-1})+1\le\sum_{k\in I_A}(k\cdot z_k^{a,i}),\ \forall(a,i)\in A\backslash\{(a,1)|a\in A_{type}\} \tag{17}$$

$$s_{a,i-1}+TP_a\le s_{a,i},\ \forall(a,i)\in A\backslash\{(a,1)|a\in A_{type}\} \tag{18}$$

$$\sum_{k\in I_A}(k\cdot z_k^{a1,i1})+1\le\sum_{k\in I_A}(k\cdot z_k^{a2,i2})+(1-x_{a1,i1}^{a2,i2})M,$$
$$\forall(a1,i1)\in A,(a2,i2)\in A \text{ such that } (a1,i1)\ne(a2,i2) \tag{19}$$

$$x_{a1,i1}^{a2,i2}\in\{0,1\},\ \forall(a1,i1),(a2,i2)\in A \tag{20}$$

$$s_{a,i},\ \forall(a,i)\in A \tag{21}$$

$$z_k^{a,i}\in\{0,1\},\ \forall(a,i)\in A,k\in\{1,...,N_A\} \tag{22}$$

The objective function (1) is to minimize total traveling time of the robot. Constraint (2) means every task except the last task at warehouse (returning warehouse at the end) has one and only one immediate following task. Constraint (3) means the last task at warehouse has no following task. Constraint (4) ensures every task expect the first task at warehouse has one and only one immediate precedent task. Constraint (5) enforces the first task at warehouse has no immediate precedent task. Constraint (6) ensures no link from a task to itself. Constraint (7) ensures the starting time of each task is not earlier than its release time, and constraint (8) enforces the finishing time (starting time + working time) is before the due time. This is safer than only requiring the starting time is before the due time. In the latter it is possible the feeders has already no stock/materials when loading material from robot to feeders.

Constraint (9) ensures that if task $(a2,i2)$ is the immediately following task of $(a1,i1)$, then the starting time of task $(a2,i2)$ should be after the starting time of task $(a1,i1)$ plus its working time plus the traveling time from location $a1$ to location $a2$. All tasks are executed one by one in a sequence. Constraints (10-11) ensure that each task is assigned to an index number in the sequence and each index number is assigned from only one task. Constraints (12-15) enforce the first task at warehouse (loading material to robot) is the first task executed in the sequence of all tasks, and the last task at the warehouse (returning warehouse at the end of execution) is the last task in the sequence of all tasks. Constraint (16) ensures the capacity limit of robot. Constraint (17) makes sure that the index of execution for task (a,i) is after that of task $(a,i-1)$ at least by one, and the starting time of task (a,i) is later than that of task $(a,i-1)$ by the working time of task type a. Constraint (19) states that if task $(a2,i2)$ is the immediately following task of $(a1,i1)$, then the index of executing task $(a2,i2)$ in the sequence is after that of task $(a1,i1)$ by one. Constraints (20-21) define the variables.

5 Numerical Results

In this section we investigate the performance of the two-index MIP model, comparing with the three-index MIP model in [6].

We use the cases in [6] to run experiments. To make completeness, the input data are also described here. There are 4 feeders in the system. The traveling times are shown in Table 1. From the table, we can see that the traveling time from a to b may be different from b to a, i.e. it is not symmetric. The working times are: 90 seconds at warehouse and 42 seconds at each feeder.

Table 1. Traveling time of robot among feeders and warehouse (seconds)

From	To				
	Warehouse 0	Feeder 1	Feeder 2	Feeder 3	Feeder 4
0	0	34	37	34	40
1	39	0	17	34	50
2	35	17	0	35	49
3	34	33	35	0	47
4	36	47	48	46	0

The robot was initially designed to carry at most 3 boxes and each box contains a replenishment of a feeder. So it can serve at most three feeders in a single trip, i.e. $Q_m = 3$. In the experiment, a variety of values for Q_m are considered. For the cases D-1 and D-2 in [6], the planning horizon is 1 hour and there are 10 tasks. The time windows of tasks are calculated and shown in Table 2. From the table, we can see that each of Feeders 1 and 4 has 4 tasks while each of Feeders 2 and 3 has 1 task.

Table 2. Time windows of tasks in cases D-1 and D-2 (1 hour planning horizon)

Task index	Feeder 1	Feeder 2	Feeder 3	Feeder 4
1	[562.5, 1125.0]	[1650.0, 3000.0]	[1650.0, 3000.0]	[562.5, 1125.0]
2	[1125.0, 1687.5]	—	—	[1125.0, 1687.5]
3	[1687.5, 2250.0]	—	—	[1687.5, 2250.0]
4	[2250.0, 2812.5]	—	—	[2250.0, 2812.5]

The MIP models are solved using solver CPLEX. Experiments are run on a notebook with Intel Core i7-6500U CPU @2.50GHz and 16G RAM. The results for cases D-1 and D-2 and a comparison with [6] are shown in Tables 3 and 4 and Figure 3. Note that [6] run the experiments on a PC with Intel Core i5 CPU @2.67GHz and 4GB RAM.

Table 3 shows the objective values and computation times under different ways: two-index model of this paper, three-index MIP model and Genetic Algorithm (GA) in [6]. Note that for both cases D-1 and D-2, we found the optimal

Table 3. Comparison of results (cases D-1 and D-2)

Case	Q_m	3-Index MIP (Dang et al)		GA (Dang et al)		2-Index MIP	
		objective	time(s)	objective	time(s)	objective	time(s)
D-1	2	488	21589	504	1	452	1152
D-2	3	384	8377	396	1	384	199

solutions fast (in minutes) while the three-index model cannot obtain optimal solution after 6 hours for Case D-1. For Case D-1, the computation time 1152 of our method is the time of finally proving that 452 is optimal objective. In fact the solver obtained the optimal solution much earlier (in 82 seconds). The solution progress for D-1 is shown in Figure 3.

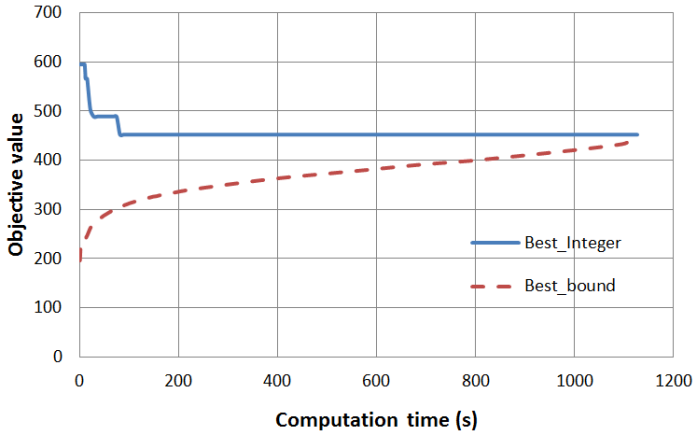

Fig. 3. Solution progress with time (Case D-1)

From Figure 3, we can see that the objective value of integer solutions decreases with time and it reaches 452 at about 82 seconds while the lower bound of the solutions increases with time, and finally at time 1152 (second) the lower bound is equal to the integer feasible solution, proving 452 is the optimal objective. For Case D-2, it is similar: the solver finds optimal solutions much earlier than proving optimality.

The effectiveness of two-index model can be explained by number of integer variables and constraints. In three-index models, Case D-1 has 4040 variables and the number of constraints increase exponentially with number of tasks. While in two-index model, there are only 903 integer variables and the number of constraints increase linearly with number of tasks. Increasing by only one binary integer variable, the searching space will be double. So our two-index model can

significantly reduce the searching space and hence find optimal solutions fast. In addition, the usage of computer memory of our model is small. The searching tree needs less than 100M.

As there are 10 tasks at feeders, there are at most 10 trips. In the optimal solutions there are some empty trips, i.e. the robot departs the warehouse and return back without serving any feeders. The non-empty serving trips for Case D-1 are as follows: Trip 1: (0,4) → (1,1) → (0,5); Trip 2: (0,5) → (4,1) → (4,2) → (0,6); Trip 3: (0,6) → (1,2) → (1,3) → (0,7); Trip 4: (0,7) → (4,3) → (4,4) → (0,8); Trip 5: (0,9) → (1,4) → (2,1) → (0,10); Trip 6: (0,10) → (3,1) → (0,11). So there are 6 non-empty trips in which 4 trips each serve 2 feeders and 2 trips each serve only one feeder.

As for Case D-2 (can serve 3 feeders in one trip), the non-empty trips are as follows : Trip 1: (0,5) → (1,1) → (4,1) → (4,2) → (0,6); Trip 2: (0,6) → (1,2) → (1,3) → (3,1) → (0,7); Trip 3: (0,9) → (4,3) → (4,4) → (0,10); Trip 4: (0,10) → (1,4) → (2,1) → (0,11). So, there are only 4 non-empty trips in which 2 of them each serve 3 feeders and others each serve 2 feeders.

From the above, when we increase the capacity of robot from 2 (Case D-1) to 3 (Case D-2), the optimal total traveling time decreases from 452 to 384, and the number of trips decreases from 6 to 4. It is intuitive.

It is desirable to compare the models for more cases. However, [6] provided results of MIP model for only two cases, as it cannot solve larger size problems. Nevertheless, the results of above two cases have already shown the benefit of the two-index MIP model in some degree (found optimal solution in mins VS not found optimal solutions in 6 hours), which confirms the theoretical analysis for the reason why the two-index model can significantly save computational time.

6 Conclusions

We have considered the vehicle routing problem with multiple trips and time windows and proposed an innovative two-index MIP model. To our best knowledge, it is the first two-index MIP model for this type of problem. This model has significant advantage in terms of computation time due to its concise modeling which significantly reduced number of integer variables and constraints, compared with the three-index model. Numeral results show this two-index model can obtain optimal solutions fast for cases while the three-index model cannot solve in many hours.

Due to the wide applications of this type of problem in smart nations and intelligent automated manufacturing, this type of problem is worth to further study. The two-index model still cannot solve larger size problems in short time. It is interesting to develop high quality heuristic for them by combining this two-index model with other techniques such as rolling horizon. In addition, the models considering stochastic characteristics in the systems are also very useful in real applications. Finally, it is natural to extend to cases of a fleet of robots. Though it is feasible to classify feeders/demands/customers into groups and each group is served by a single mobile robot using the method in this paper, con-

sidering multiple robots simultaneously still have much benefits in cost saving, improving service level and increasing system robustness.

References

1. Azi, N., Gendreau, M., Potvin, J.Y.: An exact algorithm for a single-vehicle routing problem with time windows and multiple routes. European Journal of Operational Research 178(3), 755 – 766 (2007)
2. Azi, N., Gendreau, M., Potvin, J.Y.: An exact algorithm for a vehicle routing problem with time windows and multiple use of vehicles. European Journal of Operational Research 202(3), 756 – 763 (2010)
3. Battarra, M., Monaci, M., Vigo, D.: An adaptive guidance approach for the heuristic solution of a minimum multiple trip vehicle routing problem. Comput. Oper. Res. 36(11), 3041–3050 (Nov 2009)
4. Braekers, K., Ramaekers, K., Nieuwenhuyse, I.V.: The vehicle routing problem: State of the art classification and review. Computers & Industrial Engineering 99, 300 – 313 (2016)
5. Brandão, J.C.S., Mercer, A.: The multi-trip vehicle routing problem. Journal of the Operational Research Society 49(8), 799–805 (1998)
6. Dang, Q.V., Nielsen, I., Steger-Jensen, K., Madsen, O.: Scheduling a single mobile robot for part-feeding tasks of production lines. Journal of Intelligent Manufacturing 25(6), 1271–1287 (2014)
7. Dantzig, G.B., Ramser, J.H.: The truck dispatching problem. Management Science 6(1), 80–91 (1959)
8. Fleischmann, B.: The vehicle routing problem with multiple use of vehicles (1990), facbereichWirtschaftswissenschafte Universitat Hamburg
9. Laporte, G.: Fifty years of vehicle routing. Transportation Science 43(4), 408–416 (2009)
10. Mingozzi, A., Roberti, R., Toth, P.: An exact algorithm for the multitrip vehicle routing problem. INFORMS Journal on Computing 25(2), 193–207 (2013)
11. Nielsen, I., Dang, Q.V., Bocewicz, G., Banaszak, Z.: A methodology for implementation of mobile robot in adaptive manufacturing environments. Journal of Intelligent Manufacturing 28(5), 1171–1188 (2017)
12. Olivera, A., Viera, O.: Adaptive memory programming for the vehicle routing problem with multiple trips. Computers & Operations Research 34(1), 28 – 47 (2007)
13. Salhi, S., Petch, R.J.: A ga based heuristic for the vehicle routing problem with multiple trips. Journal of Mathematical Modelling and Algorithms 6(4), 591–613 (2007)
14. Taillard, E.D., Laporte, G., Gendreau, M.: Vehicle routeing with multiple use of vehicles. The Journal of the Operational Research Society 47(8), 1065–1070 (1996)
15. Toth, P., Vigo, D., Toth, P., Vigo, D.: Vehicle Routing: Problems, Methods, and Applications, Second Edition. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA (2014)