Vehicle Routing with Multiple Trips and Time Windows

anonymous

Abstract

We consider a vehicle routing problem with multiple trips and time windows in continuous manufacturing environment in which a mobile robot transports materials from a central warehouse to multiple demanding places. Each demand has its own time window during which the robot should deliver material. The manufacturing is a continuous process with long horizon and the robot need to run multiple trips. In the literature three-index mixed integer programming (MIP) models are developed. However the three-index models are difficult to solve in reasonable time for real problems due to computational complexity of integer programming. We develop an innovative two-index MIP model to minimize traveling time. This model can significantly reduce computation time, compared with the three-index MIP model. For practical large-size problems, we also propose a method called Short Window method which calls the two-index MIP model. This method can save much cost, compared with the GA method in the literature. The computation time is also short and can satisfying requirement in real production environment.

1 Introduction

We consider the vehicle routing problem with multiple trips and time windows (VRPMTTW) in which vehicles (or mobile robots) can perform multiple trips to satisfy customers' demand and each customer has its own time window such that the vehicle needs to arrive in this window. The work is motivated by an application during our industrial engagement. The company has a few production lines and each line has one or more feeders for material feeding to the line. Mobile robots are used to transport material from a central raw material warehouse to the feeders. Each feeder has its capacity limit. Each feeding should be finished at its time window. (Dantzig and Ramser 1959) have also studied a similar problem. This type of problems will become increasingly important in the contexts of Smart Nations and Future of Factory. It is obvious that in the smart nations and future of factory, intelligent autonomous vehicles and mobile robots will be widely used. How to effectively schedule these mobile robots and vehicles is critical for implementing Smart Nation and intelligent manufacturing.

Copyright © 2018, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

In some vehicle routing problems, 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 window of feeders, otherwise, the production line may have to stop or generate wasted products due to missing some material. 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, sometimes vehicles need to satisfy customers' time windows, though they may not be so strict/critical as in production and healthcare areas.

In order to save cost, it is natural that a vehicle/mobile robot can perform multiple routes/trips only if it can satisfy the time window. So, vehicle routing problem with multiple trips and time windows (VRPMTTW) arises 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 one route. In the problem VRPMTTW, we need to explicitly consider the precedence of routes when a vehicle runs multiple routes, 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 VRP problems (just relaxing the constraints for multiple trip and time windows). Some researchers developed mixed integer programming (MIP) models which use three-index variables and have exponential-size of constraints. These models are very difficult to solve in reasonable time for real problems, hence some researchers developed heuristics such as Genetic Algorithm (GA) for it. We develop an innovative MIP model which uses two-index variables and has linear-size of constraints. This model have significant benefit in computation time, compared with three-index MIP model. For practical large size problems such as long planning horizon, we also propose a method called Short Window which splits the long horizon into short planning windows and then call the two-index MIP model for each window. The numerical results show this method is feasible for large-size problems and can save much cost, compared with the GA method in the literature.

The remainder of the 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 two-index mixed integer programming model. In Section 5, we propose a method to deal with scheduling over long planning. The numerical results are reported in Section 6. Finally, concluding remarks follow in Section 7.

2 Literature Review

Vehicle routing problem can be traced back to (Dantzig and Ramser 1959). Since then many researchers have studied it and a variety of variants and made many significant progresses in developing heuristics and exact algorithms. Some important variants are Vehicle Routing Problem 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 (Laporte 2009), (Toth et al. 2014) and (Braekers, Ramaekers, and Nieuwenhuyse 2016).

Most papers in the above literature assume one vehicle performs one route. However, in practice people often use one vehicle for multiple routes to save cost. Nevertheless, only a few consider the Vehicle Routing Problem with Multiple Trip (VRPMT). (Fleischmann 1990) first considered this problem. He modified the well-known saving algorithm and used a bin-packing heuristic to assign routes to the vehicles. Since then more people developed some heuristics using genetic algorithms, tabu search and so on (e.g. (ric D. Taillard, Laporte, and Gendreau 1996), (Brandão and Mercer 1998), (Salhi and Petch 2007), (Olivera and Viera 2007)). (Mingozzi, Roberti, and Toth 2013) 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, Gendreau, and Potvin 2007) first addressed this type of problem in 2007. They developed an exact method for the single vehicle case. The method consists of two phases: in the first phase, all non-dominated feasible routes are generated; in the second phase, some routes are selected and sequenced to form the vehicle workday. (Azi, Gendreau, and Potvin 2010) extended (Azi, Gendreau, and Potvin 2007) 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, Monaci, and Vigo 2009) 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. 2014) considered VRPMTTW 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. They developed a three-index integer programming model with exponential-size constraints. As this model is very time consuming, they developed a GA algorithm for it. (Nielsen et al. 2017) extended the work of (Dang et al. 2014). As mentioned in (Nielsen et al. 2017) , GA 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.

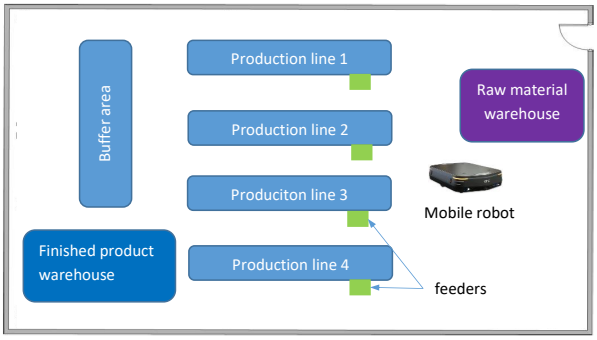

Figure 1: An example of layout of flexible manufacturing systems with robots

In this paper, we consider the same problem as (Dang et al. 2014) in which a single robot transports materials/parts from a warehouse to feeders. We propose an innovative two-index integer programming model. It is much easier to solve using general purpose solver as shown in Section 5. For larger size problems, we also propose a method called Short Window which can solve real problem in short time and save cost,compared with GA algorithms

3 Problem Description

In flexible manufacturing systems, there are multiple 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 (Dang et al. 2014).

From the figure, we can see that the mobile robot transports materials from warehouse to feeders. Each robot has a capacity limit. It is possible for a robot to carry materials for multiple feeders in one route (if the robot capacity allows it) and then come back to warehouse to start a new route. The capacity of robots can be measured by boxes as in (Dang et al. 2014) 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. This figure

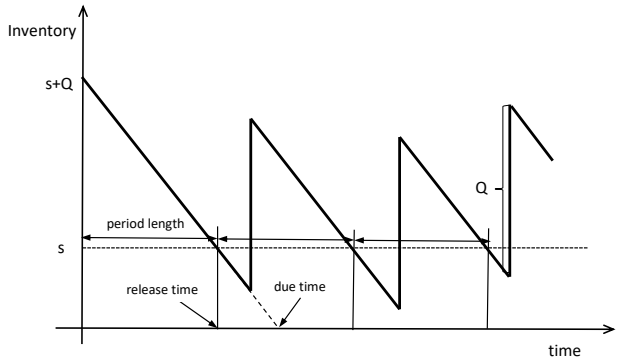

Figure 2: (s,Q) inventory policy for feeding materials at feeders

shows how the inventory (material/components) at a feeder changes with time. At the beginning, there is an initial inventory (it is at the maximum level in this figure). The inventory decreases with the time. When the inventory reach the 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. In general, each machine of production lines runs at its constant speed, i.e. the material consumption rate is a constant, so the inventory decrease at a constant rate in the figure. Due to 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 route (from departure from warehouse to coming back to warehouse) if the robot's capacity allows it. This should be more efficient than serving only one feeder in one route. After coming back to warehouse, the robots can load materials to serve later demands at feeders again and again. How to schedule robots to satisfy demands at feeders with multiple trips and hard time window is a critical problem to implement automated intelligent flexible manufacturing. As in (Dang et al. 2014), 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, the finished products are needed to send to product warehouse. At the end of production lines, there is a limited buffer area for products. An intelligent forklift (no driver) is used to transport products to warehouse. The forklift 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 mixed integer programming model for scheduling the mobile robot with the objective of minimizing total traveling distance.

Assumptions and Notations

Assume there are n_f feeders. Let $V = \{1, ..., n_f\}$ be the set of feeders, and let $V^+ = \{0\} \cup V$, where 0 denote the warehouse. A single mobile robot of capacity Q_m transports materials from the warehouse to the set of feeders. Each element/unit (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 the initial inventory, s and Q , and material consumption rate at each feeder 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 and it is the unloading time from robot at each feeder. These working times are constants.

Assume a single delivery/stop at a feeder can fulfill one replenishment of this feeder. If $Q_m = n$, the robot can carry materials to serve n feeders in a single route/trip (duration from departure from warehouse to returning back to warehouse). Assume at the beginning of planning, the robot is at the warehouse.

We regard the material replenishment at feeders and loading material at warehouse 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 route/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.

$p_a, a \in A_{type} \backslash \{0\}$, period length 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 $N_A^a = \lfloor T/p_a \rfloor$. 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 feeders. 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), and

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

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

$TC_{a1,a2}$, $a1, a2 \in A_{type}$, changeover time from task type $a1$ to task type $a2$, i.e. traveling time from one place (after having finished task $a1$) to another place to start task $a2$. M, a very 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 route of the robot, i.e. maximum number of feeders that the robot can carry material and serve in a route 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 routes. So the number of integer variables in three-index formulation is many times of ours.

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^- routes 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 $\lceil N_A^-/Q_m \rceil$ routes by assuming each route 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 routes 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 $(n-m-1) \leq Q_m$. So, in this way, we enforce the precedence relation between trips/routes and the robot capacity limits.

We also need other constraints such as stratifying time windows. The objective is to minimize the traveling distance. The new compact two-index mixed integer programming (MIP) model is as follows:

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

$$\sum_{k\in I_A} (k \cdot z_k^{a,i-1}) + 1 \leq \sum_{k\in I_A} (k \cdot z_k^{a,i}),$$

$$\forall (a,i) \in A \backslash \{(a,1) | a \in A_{type}\} \quad (17)$$

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

$$\sum_{k \in I_A} (k \cdot z_k^{a1,i1}) + 1 \leq \sum_{k \in I_A} (k \cdot z_k^{a2,i2})$$
$$+(1 - x_{a1,i1}^{a2,i2})M, \ \forall (a1,i1) \in A, (a2,i2) \in A$$
$$\text{s.t. } (a1,i1) \neq (a2,i2) \quad (19)$$

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

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

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

The objective function (1) is to minimize the weighted sum of total traveling time and the finishing time of tasks at feeders. Constraint (2) means every task expect 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 and 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. 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 during adding 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 by 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 (18) ensures that for tasks of the same type, the task with smaller index will start earlier than the tasks with larger indexes. 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 (20-22) define the variables.

5 Method for Scheduling over Long Planning Horizon

Thee above two-index MIP model has great advantage in terms of computation time as shown in Section 6, compared with the three-index MIP model. However, it is still not able to solve practical large size problems in reasonable time. When the planning horizon is long (e.g. 8 hours) , the number of integers will be very large. In this section we propose a method to schedule robots over long planning horizon.

We call this method as Short Window (SW) method. The main idea is to split the long horizon into multiple short planning windows, then for each planning window we call the above MIP model to get the schedules over the whole horizon. The scheduling in one planning window will affect the scheduling in next window. We want to finish all the tasks in a planning window ASAP so that the robots can start tasks for the next window ASAP. So, the objective in one planning window is the weighted sum of total travelling time and the make span (the finishing time - starting time). Let

T_s, the earliest start time for the robot to leave depot in current planning window. It is obtained from the results of scheduling in previous window, i.e. it is the time for the robot returning back to depot in the previous window.

T_e, the latest time for finishing all tasks except returning back to depot at the end. So, $T_e - T_s$ is the makespan.

W_T, the weight for the makespan in the objective.

So, the new MIP model for short planning windows is as follows:

$$min \quad \sum_{(a1,i1)} \sum_{(a2,i2)} TC_{a1,a2} \cdot x_{a1,i1}^{a2,i2} + W_T(T_e - T_s) \quad (23)$$

$$s.t. \quad \text{constraits } (2-22) \quad (24)$$

$$s_{0,1} \geq T_s \quad (25)$$

$$s_{a,i} + TP_a \leq T_e, \ \forall (a,i) \in A \backslash A^0 \quad (26)$$

Objective (23) is to minimize the weighted sum of traveling time and makespan. Constraint (25) says the the robot in current planning window cannot start work before a certain time T_s which is the returning time of robot in previous planning window. Constraint (26) expresses the finishing time of all tasks at feeders are less than or equal to T_e. T_e plus the traveling time to the depot will be the earliest time for tasks in next planning window, T_s.

6 Numerical Results

In this section, we investigate the performance of two-index MIP model, by comparing with the three-index MIP model in (Dang et al. 2014). We also check the feasibility and performance of the SW method for long planning horizon, comparing with the GA method in (Dang et al. 2014).

Comparison between two-index MIP with three-index MIP

We use the cases in (Dang et al. 2014) to compare the models. There are 4 feeders in the system. The traveling time are shown in Table 1. From the table, we can see that the

Table 1: Traveling time of robot (second)

From	To				
	0	1	2	3	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

Feeder	1	2	3	4
Max level (part)	250	2000	2000	250
Min level (part)	125	900	900	125
Part feed rate(sec/part)	4.5	1.5	1.5	4.5

Table 2: Parameters of feeders

traveling time from location a to location b may be different from b to a, i.e. it is not symmetric. The working time at warehouse is 90s, and it is 42s at each feeder.

There are two cases: D-1 in which $Q_m = 2$ and D-2 in which $Q_m = 3$. For both cases D-1 and D-2, the planning horizon is about 1 hour and there are 10 tasks. Feeders 1 and 4 have 4 tasks and Feeders 2 and 3 have only one task. The parameters of inventory control policies at feeders and material consumption rates are shown in Table 2. From this table, we calculate the time window (releasing time to due time) for each task.

The MIP models are solved using CPLEX solver. 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 (Dang et al. 2014) are shown in Tables 3 and Figures 3 and 4. Note that Dang et al. (Dang et al. 2014) run the experiments on a PC with Intel Core i5 @2.67GHz CPU and 4 GB RAM.

Table 3 shows the objective values and computation times under different ways: two-index model of this paper, three-index MIP model and GA algorithm in (Dang et al. 2014). Note that for both cases D-1 and D-2, we found the optimal solutions in short time while the three-index model can not obtain optimal solution after 6 hours for case D-1. For case D-1, the computation time 1152s of our method is the time of finally proving that 452 is optimal objective value in the worse case (with N_A^0 at the largest possible value, i.e. number of tasks plus 1, which is 11 in this case). When we reduce N_A^0, the computation time will further reduce to seconds. As for objective difference, for case D-1, the 3-index model is 8.0% higher than 2-index model, and the GA heuristic is 11.5% higher than 2-index model. For case D-2, the 3-index model also found the optimal solution. The GA heuristic is 3.1% higher than 2-index model.

Table 3: Comparison of different models and methods

Case	Q_m	3-Index MIP		GA		2-Index MIP	
		obj	time	obj	time	obj	time
D-1	2	488	21589	504	1	452	1~1152
D-2	3	384	8377	396	1	384	1~199

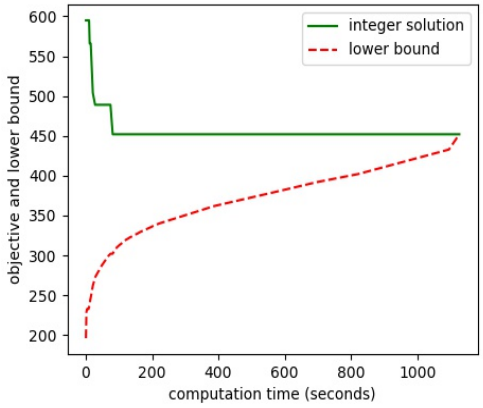

Figure 3: Solution progress (case D-1, $N_A^0 = 11$)

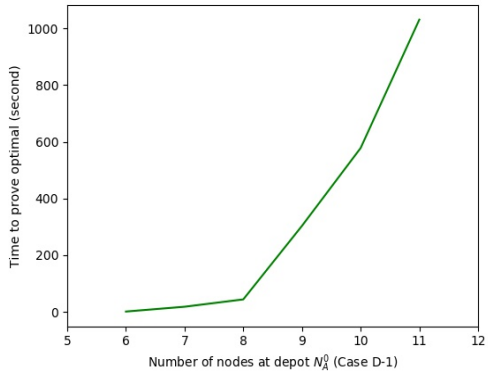

Figure 4: Computation time changing N_A^0 (case D-1)

The solution progress for case D-1 with $N_A^0 = 11$ is shown in Figure 3. From this figure, we can see that the objective value of integer solutions decreases with time and it reaches 452 at about 82 seconds. And the lower bound of the solutions increases with time, and finally at time 1152s, the lower bound is equal to the integer feasible solution, proving 452 is the optimal objective value. So, in fact, the model obtains the real optimal solution at 82s.

When we change the parameter N_A^0, the computation time will change. The computation time (finding optimal solutions and proving the optimality) changing N_A^0 for case D-1 is shown in Figure 4. From the figure, when we reduce N_A^0, the computation time will reduce. From the model, we can see that when we reduce N_A^0, the number of integer varibles will reduce, hence the computation time will decrease. It is intuitive. For case D-3, there is a same trend.

In summary, the two-index MIP model has great benefit in terms of computation time, compared with three-index MIP model: the three-index model cannot obtain optimal solutions after 6 hours, the two-index models can find it

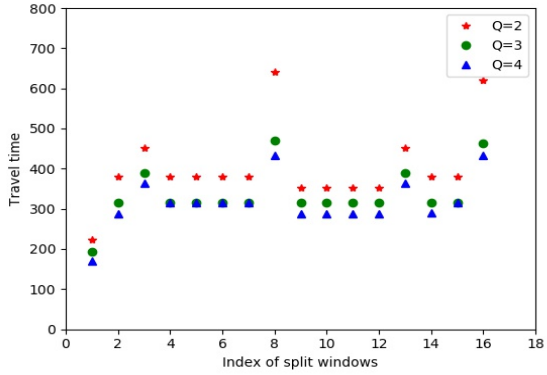

Figure 5: Traveling time in planning windows (T=8h)

in less than 2 mins. The effectiveness of two-index model can be explained using number of variables and constraints. In three-index model, 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 variables and the number of constraints increase linearly with number of tasks. In addition, the usage of computer memory of our model is small. The searching tree needs less than 100M.

Comparison between short window method and GA heuristic

For long planning horizon problems which are too large for general MIP models, we compare previous short window method with GA heuristic in (Dang et al. 2014).The results are shown in Figure 5 and Table 4.

In the short window method, we chose planning window length about half of hour. The planning horizon is $T = 8$ hour. So there are 16 windows. But the number of tasks in each planning window are not the same. Windows 8th and 16th have 12 tasks, respectively. most of other windows have only 8 tasks. Each planning window calls the previous two-index MIP model. We set the time limit for the solver is 2 minutes. The total traveling time in each planning window is shown in Figure 5. From the figure, we can see that the traveling time in windows 8th and 16th are significantly higher than those of other windows. This is due to these two windows have higher number of tasks. It is also due to the computation time limit of 2 mins. As it has 12 tasks, the solver needs more than 2 mins to get optimal or near-optimal solutions. In window 1st, there are only 4 tasks, so the traveling time is low.

The total traveling time of different methods over the long horizon are shown in Table 4. From the table, GA method has a higher traveling time of 3.2-4.5% than the Short Window method which calls the 2-index MIP model. The computation time in some planning windows is less than 1 min because the solver CPLEX found the optimal solution in tens of seconds. Note that in GA method for case of T=8h, each run needs about 20 seconds to get a solution and need to run

Table 4: Comparison of methods for long horizon(T=8h)

Q_m	T	Objective		Difference
		GA	Short Window	
2	8	6738	6447	4.5%
3	8	5667	5373	5.5%
4	8	5223	5063	3.2%

many times. These results show the short window method (plus 2-index MIP model) can be used to solve really large size problems.

From the above results we can see that the two-index MIP model can find the optimal solutions in short time (less than 3 mins) for the two cases in (Dang et al. 2014) while (Dang et al. 2014) spent 6 hours and still cannot find optimal solutions for one case. It is desirable to compare the two-index model with the three-index model for more cases. However, Dang et al.(Dang et al. 2014) provided results of three-index model only for two cases, as it cannot solve larger size problems. Nevertheless, the results of above two cases already show the benefit of the two-index model in great degree, which confirms the theoretical analysis for the reason why the two-index model can have great benefit in saving of computational time.

7 Conclusions

We have considered the vehicle routing problem with multiple trips and time windows in which a mobile robot transports materials among different places. The time windows of tasks must be applied strictly and the robot can travel many trips. We developed an innovative two-index MIP model to minimize traveling time. This model can significantly reduce computation time, compared with the three-index MIP model in the literature. While the three-index MIP model cannot obtain optimal solutions after 6 hours, the two-index model can obtain it in less than 2 mins. We also proposed a method called Short Window method for large-size real production problems. This method can save much cost/traveling time, compared with the GA method in the literature.

There are a few interesting directions for future research. First, we may refine the Short Window method. For this method, we may split the long planning horizon based on number of tasks, not current way based on planning window length. If splitting horizion based on window length, we may allocate computation time more wisely, for example, for planning windows with more tasks, allocate more computation time so that the solver can find near-optimal solution. Second, we may analyze the pattern of trips, and then develop more efficient method to split long horizon. Finally, we may consider the cases of a fleet of robots. Sometimes we may need a fleet of robots to work together to finish the tasks.

References

Azi, N.; Gendreau, M.; and Potvin, J.-Y. 2007. An exact algorithm for a single-vehicle routing problem with time win-

dows and multiple routes. *European Journal of Operational Research* 178(3):755 – 766.

Azi, N.; Gendreau, M.; and Potvin, J.-Y. 2010. 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.

Battarra, M.; Monaci, M.; and Vigo, D. 2009. An adaptive guidance approach for the heuristic solution of a minimum multiple trip vehicle routing problem. *Comput. Oper. Res.* 36(11):3041–3050.

Braekers, K.; Ramaekers, K.; and Nieuwenhuyse, I. V. 2016. The vehicle routing problem: State of the art classification and review. *Computers & Industrial Engineering* 99:300 – 313.

Brandão, J. C. S., and Mercer, A. 1998. The multi-trip vehicle routing problem. *Journal of the Operational Research Society* 49(8):799–805.

Dang, Q.-V.; Nielsen, I.; Steger-Jensen, K.; and Madsen, O. 2014. Scheduling a single mobile robot for part-feeding tasks of production lines. *Journal of Intelligent Manufacturing* 25(6):1271–1287.

Dantzig, G. B., and Ramser, J. H. 1959. The truck dispatching problem. *Management Science* 6(1):80–91.

Fleischmann, B. 1990. The vehicle routing problem with multiple use of vehicles. FacbereichWirtschaftswissenschafte Universitat Hamburg.

Laporte, G. 2009. Fifty years of vehicle routing. *Transportation Science* 43(4):408–416.

Mingozzi, A.; Roberti, R.; and Toth, P. 2013. An exact algorithm for the multitrip vehicle routing problem. *INFORMS Journal on Computing* 25(2):193–207.

Nielsen, I.; Dang, Q.-V.; Bocewicz, G.; and Banaszak, Z. 2017. A methodology for implementation of mobile robot in adaptive manufacturing environments. *Journal of Intelligent Manufacturing* 28(5):1171–1188.

Olivera, A., and Viera, O. 2007. Adaptive memory programming for the vehicle routing problem with multiple trips. *Computers & Operations Research* 34(1):28 – 47.

ric D. Taillard; Laporte, G.; and Gendreau, M. 1996. Vehicle routeing with multiple use of vehicles. *The Journal of the Operational Research Society* 47(8):1065–1070.

Salhi, S., and Petch, R. J. 2007. A ga based heuristic for the vehicle routing problem with multiple trips. *Journal of Mathematical Modelling and Algorithms* 6(4):591–613.

Toth, P.; Vigo, D.; Toth, P.; and Vigo, D. 2014. *Vehicle Routing: Problems, Methods, and Applications, Second Edition*. Philadelphia, PA, USA: Society for Industrial and Applied Mathematics.