Chapter 1: Introduction

1.1. Introduction

The routing and scheduling of vehicles and crews is critical in distribution management. Many existing transport logistics problems can be modelled as vehicle routing problems (VRPs). The objective of modelling is to establish a planned route to serve a set of customers at minimum cost. However, real-life applications are often more complex due to specific requirements that naturally exist for practical operations, such as limited capacity of vehicles, location facilities and priority delivery. Another complicating factor is the allocation of labour and other resources or equipment to carry out an operation over time. In order to make full use of the available resources, management must plan precisely what has to be done, when, how and by whom. With dozens of teams and hundreds of tasks, the planning process quickly becomes immense.

Recent research in this field not only includes significant breakthrough in the construction, analysis and implementation of solution procedures, but also in the computational efficiency involved in generating solutions. These advances can save millions of dollars and significant time by increasing productivity and facilitating management planning and financial control.

A good example of the routing and scheduling of vehicles and crews is ground maintenance scheduling at airports. Numerous tasks such as luggage handling, cleaning and in-flight catering services must be performed within the transit time of an aircraft. Typically, the synchronisation of teams to conduct maintenance services is required. Thus, a roster plan must consider constraints such as aircraft transit time, manpower allocation, required equipment, shift work hours and union regulations.

In VRP, a vehicle *route* is defined as a sequence of customer sites for delivery work and a vehicle *schedule* is defined as a sequence of deliveries to customer sites with an associated set of arrival and departure times. The vehicle must travel to the customers in a designated order within a specified timeframe. When visits to customer locations must be made at specific times, the problem that arises is an integrated vehicle routing and scheduling problem, known as a vehicle routing problem with time windows (VRPTW). Such problems often arise in practice and are representative of many real-world applications.

1.2. Vehicle Routing Problem with Time Windows (VRPTW)

In VRPTW, a fleet of vehicles is dispatched from the depot to serve a set of customers at different locations. Each customer must be visited exactly once within a predefined time window. The route must be constructed while taking into account travel distance and time, a vehicle's capacity and customers' hours of operation. It is well-known that VRPTWs are NP-hard even under very basic conditions (Lenstra and Rinnooy Kan, 1981; Savelsbergh, 1985).

According to a survey by Bräysy and Gendreau (2005), the main objective function of a VRPTW is to minimise the number of tours, travelling distance and duration. In recent years, VRPTWs with multiple objectives have received much attention (El-Sherbeny, 2010). There are also studies about the additional constraints of more complex VRPTW models, such as soft time windows (Ioachim et al., 1998), pick-up and delivery (Lin, 2008), split delivery (Frizzell and Giffin, 1995), multi depots (Dondo and Cerdá, 2009), heterogeneous fleet vehicles (Dondo and Cerdá, 2007) and multiple time windows (Doerner et al., 2008).

In real life, vehicle routing and scheduling problems can be very difficult to model mathematically due to the abundance of operational constraints. Even if the modelling problem can be overcome, providing optimal or even efficient approximations is extremely challenging. In most cases, decisions have to be made in a very short time and often data is only available close to deadlines. For example, aircraft maintenance and repair services at airports are expected to cope with emergency circumstances instantly and home delivery systems must decide whether to accept customer requests for on-time delivery (Campbell, 2005).

Due to the NP-hardness of VRPTWs, heuristic and metaheuristic solution techniques have become popular in order to cope with large problems. Many efficient heuristic and metaheuristic solutions have been proposed and successfully applied in both the commercial and public sectors (Bräysy et al., 2009). The two main reasons for the wide applicability of heuristic solutions are their flexibility in real-life applications and their ability to provide high quality solutions in a limited timeframe with minimal computational effort.

Most heuristic solutions are implemented using a two-stage approach, i.e., route construction and solution improvement (local searches). Solomon (1987) proposed a remarkable route construction insertion heuristic. The proposed solution is initialled by a "seed" customer, followed by inserting an "unrouted" customer that gives the maximum saving on the current partial route or minimum cost of insertion. The insertion process is

repeated until no more feasible customers are found. Next, a new "seed" customer is generated to sequentially construct a new route. In the paper, six different characteristic problem sets are developed to evaluate a few proposed solutions. A comprehensive review of solution improvement methods can be found in Bräysy and Gendreau (2005). Russell (1995) proposes a global route enhancement while constructing a partial route. The local solution improvement scheme is applied by interchanges partial route after every k customers have been routed. The proposed hybrid approach has been claimed to outperform the traditional two-stage algorithm.

Much of the recent literature discusses the use of metaheuristic techniques to solve VRPTWs. A metaheuristic approach provides a standard algorithm framework by embedding route construction and improvement heuristics to identify a high quality feasible solution in a solution space (Bräysy and Gendreau, 2005). Some of the well-known metaheuristic approaches are tabu search (Paraskevopoulos et al., 2008; Taillard et al., 1997), genetic algorithms (Potvin and Bengio 1996; Tan et al., 2001; Baker and Ayechew, 2003), simulated annealing (Chiang and Russell, 1996; Bent and Van Hentenryck, 2004), ant colony (Donati, 2008), evolution algorithm (Bräysy et al., 2004; Mester and Bräysy, 2005) and guided local search (Zhong and Cole, 2005).

In terms of exact solutions, there are three main approaches—dynamic programming (Kolen et al., 1987), Lagrangian relaxation (Kohl and Madsen, 1997; Kallehauge et al., 2006) and column generation (Desrochers et al., 1992; Kohl et al., 1999; Chabrier, 2006). In the last two decades, Desrochers et al. (1992) optimally solved some of Solomon's (1987) 100-customer test problems. Kohl et al. (1999) proposed an inequality cut, called a 2-path cut, encompassed by the branch and cut algorithm to optimally solve the first 150-customer test problem. However, neither study managed to solve all of Solomon's (1987) 50-customer size-reduced test problems. Recently, Feillet et al. (2004) suggested using the elementary shortest path problem with resource constraints instead of the non-elementary column generation (CG) model to solve VRPTWs. The computational result showed that the proposed model reduces the duality gap for branching, but there is no improvement in terms of problem size. In one experimental run, the model took almost an hour to solve an instance of size 25 customers. Based on previous studies, a VRPTW is unlikely to solve larger-scale problems optimally due to the exponential size of solution space.

1.3. Multi-trip Vehicle Routing Scheduling Problems with Time Windows (MTVRSPTW)

In many transportation and logistics settings, multi-trip travelling exists and it heavily affects various industries. The main reasons are due to the limited capacity of the vehicle, delivery of perishable goods where limited travelling time applied and also the working shift patterns. For example, school meal deliveries and in-flight catering services have to travel multi-trip because of limited truck capacity. Other applications can be referred to research done by Taillard et al. (1996), Brandão and Mercer (1997), Petch and Salhi (2003) and Salhi and Petch (2007)). Surprisingly, in the literature, multi-trip vehicle routing scheduling problems with time windows (MTVRSPTW) have received little attention.

A MTVRSPTW is a more complex version of a VRPTW, having a significantly larger number of constraints. Most papers about MTVRSPTWs involve real-life application-based simulations and the problems are solved using heuristics. For example, Brandao and Mercer (1997) study a biscuit distribution problem with less than 70 orders, and Cornillier et al. (2009) consider a petrol station replenishment problem. Both studies report an average of 20% improvement over manual solutions on minimising operation cost provided by various expert planners. Cornillier et al. (2009) state that solving the petrol station replenishment problem optimally, even for small cases with 15 stations, is computationally intractable. Furthermore, the proposed heuristic approach requires an average of 30 minutes to provide approximation solutions for a 50-station problem. Battara et al. (2009) propose an adaptive guidance approach to solve the distribution of goods to supermarkets and hypermarkets, with an objective function of minimising the size of required fleet. The sample size used in the study is 422 nodes on average. Battara et al. reported that an average of five minutes is required to obtain the best solution. In manpower scheduling for airline catering, Ho and Leung (2010) suggested a tabu search approach to achieve the minimum number of unassigned jobs. The proposed solution is compared with simulated annealing and the CPLEX search engine. As reported in the paper, CPLEX took as much as 22 hours to provide a feasible solution for a sample size of 101, after fixing one of the decision variables. There are also reports about different variants of MTVRSPTWs in waste management by Sahoo et al. (2004) and Kim et al. (2006).

In Azi et al. (2007), an exact two-phrase approach is proposed to solve a multi-trip single vehicle with time windows. Azi et al. (2010) suggested a branch-and-price algorithm to optimally solve a VRPTW problem with multiple uses of vehicles. The problem objective is to maximise the total revenue versus minimal the total travelled distance. The algorithm is

limited to small examples of up to 40 customers. Some of the instances with 25 customers could not be solved within 30 hours. In addition, the algorithm is sensitive to the time window's width and route time limit. Clearly, a heuristic solution is well justified for this problem, not only because of the problem's complexity but also because of the ease that it can be implemented in various industries where there is a fundamental need to finalize operational decisions in a matter of seconds for large sample sizes.

1.3.1. MTVRSPTW Formulation

The MTVRSPTW can be formally defined as follows. We consider a set of customers, $A = \{1, \cdots, n\}$, that needs to be served by a set of service teams. Each customer i has a processing time p_i, demand q_i and a time window, $[r_i, d_i]$, where r_i denotes the earliest time a service can take place and d_i denotes the latest time a service can be completed. These may be viewed as release times and due dates (deadlines), respectively.

All the customers are served by a set of service teams, K, that use a fleet of homogenous vehicles with vehicle capacity Q. Each service team can only serve one customer at a time and may proceed to the next customer only after completing the service for the current customer. Due to economically significant, the objective function of the problem is to minimise the operation costs such as the total travelled distance, vehicle required, and the total manpower.

Each service team is required to serve a sequence of customers on multiple trips, subject to a trip time limit F constraint and vehicle capacity considerations. A complete trip must begin and end at the depot. The depot is denoted by $n+1$ and $n+2$. A route is defined as a working shift assignment for a loading team that consists of one or more trips, all completed within T time units.

The MTVRSPTW is unlikely to be solved optimally for reasonable examples, as illustrated by Azi et al. (2010) on minimum travelled distance.

1.4. Multi-trip Vehicle Routing and Scheduling Problem with Time Windows and meal break consideration (MTVRSPTW-MB)

This real life constraint, i.e., meal break allocation is an essential requirement in manpower required operations. In VRP, it also would cause the affect of multiple trips travelling decision when the drivers are required to return to depot or rest centre. However, there is limited research conducted on this problem.

Even though it is hard to find literatures on meal break allocation coupled with the vehicle routing consideration, it appears in many areas such as workforce planning, staffing scheduling or rostering. The comprehensive survey meal breaks in staff scheduling can be referred to Ernst et al. (2004) and Alfares (2004).

In most studies, VRP with meal break allocation are based on real-life application, such as pet food and flour delivery in Switzerland by Rochat and Semet (1994) and crew workforce planning by Dohn et al. (2009). Break time allocation has become a popular consideration in studies involving driver's working shift scheduling. This is due to the increasing legislation which requires drivers to be properly rested. For example, Prescott-Gagnon et al. (2010) conducted a study on European driver rules in VRPTW.

In term of multi-trips vehicle travelling, Brandão and Mercer (1997) proposed Tabu Search to solve a manpower scheduling problem with several additional constraints including meal break allocation. A real life waste collection vehicle routing problem also considers multiple disposal trips and driver's lunch break, such as studies done by Kim et al. (2006) and Benjamin and Beasley (2010). However, the multiple disposal trips do not start and end at depot as defined in previous section.

In addition, most of the previously studied variants of MTVRSPTW or MTVRSPTW-MB do not consider models as (i) there is more than one operational constraint which affects the decision of making multiple trips travelling, e.g.: travelling trip time limit and truck capacity, (ii) the unavailability of overtime for workers, (iii) the workers have to return to depot to have their lunch, (iv) workforce synchronisation in servicing customer.

1.4.1. Case Study

In-flight food loading operations for aircraft are modelled as MTVRSPTW-MB in order to examine the performance of our proposed solutions. In real life, several loading teams are required to deliver and upload packaged meals. The meals are transported by loading trucks travelling from a central kitchen to the aircraft that land at a nearby apron. Loading trucks can only serve a limited number of aircraft in a single trip; in-flight food has to be stored on aircraft within a certain time period after having left the kitchen in order to maintain freshness. Furthermore, each loading team is given a meal break during their shift. Due to safety reasons, loading teams are restricted to having their meal break in lounge areas. These operational constraints heavily affect vehicle allocation and travelling policies. The fundamental assumption, or constraint, is that all aircraft must be served within their transit time at the apron. Due to tight time windows and different aircraft types, some aircraft

require more than one loading team to offload and upload in-flight food on time. More information on this case study is described in Chapter 2.

In terms of modelling, we assume that each vehicle can perform several trips during a shift. There are a few transportation requirements because the teams must travel back and forth to the depot (the kitchen in our case) in order to provide service. Each trip is subject to a time limit constraint and limited truck capacity. In addition, a mandatory meal break must be scheduled within a predefined time interval. All customers (aircraft in our case) must be visited exactly once, any time within a predefined time window. A certain amount of time is required at the depot and customer sites to load and unload goods. For convenience, this period is included in the travelling times.

Since the problem is motivated by real-life applications, the primary goal is to minimise the total manpower while providing satisfactory service (meeting all deadlines) to all customers. In addition, the efficient synchronisation of loading teams has become an important and highly challenging task for airlines. Due to the larger-scale problems demand from industry, this research focuses on developing a computationally bounded heuristic that generates high quality solutions in a short time.

Yan et al. (2008) developed two stochastic-demand manpower supply plans based on a case study on a Taiwanese air cargo terminal. In the study, manpower supply was not considered when planning vehicle routing at the airport. Another manpower allocation real-life application at two European airports was conducted by Dohn et al. (2009). Skills and constraints limiting the number of available workers complicate the initial model of VRPTW with objective to maximise the number of assigned tasks. The problem is optimally solved through the application of a branch and price approach using the framework developed by CG. Ho and Leung (2010) solved an airline catering loading operation at Hong Kong International Airport using tabu search. The operation was modelled as an MTVRSPTW to achieve the minimum number of unassigned jobs. However, crew meal breaks were not considered in the study.

1.5. Purpose

The purpose of the thesis is threefold. The main purpose of this research is to develop a computationally bounded heuristic algorithm that can solve the MTVRSPTW-MB in the case study in seconds. Achieving this objective would benefit at least the following three parties:

a) *Management teams* save time and effort in planning and rostering operations. In addition, resources such as loading trucks and equipment are efficiently utilised. Finally, the solutions could provide a better rostering system for coping with peak and off-peak periods.
b) *Airlines* in risk reduction in emergency situations due to fast response times.
c) *Loading teams* by providing them a proper meal break which must be allocated within a given time window.

Furthermore, it is also important to evaluate the quality of current industries practises and the proposed heuristic algorithms. We consider several variants of the problem under different operational constraints. Our underlying different variants are given below:

- *Meal break allocation*

  Each loading team is given a meal break during their working shift within a predefined time window, which is normally not included in previous VRPTW research. This essential constraint is considered in every heuristic developed in this study in order to cope with current practises.

- *Fixing the number of trips*

  Each loading team is limited to travel exactly two trips for meal break allocation between trips. The model, which is two trips VRSPTW with meal break consideration, is described in Chapter 4. In this scenario, the trip travelling time limit and truck capacity constraints are relaxed.

- *Synchronisation of service teams*

  In this study, two versions of MTVRSPTW-MB are considered; taking into account loading teams' synchronisation and ignoring this requirement. When synchronisation of loading teams is applied, different teams are required to serve a particular aircraft during the same time period. Otherwise, loading teams are given flexibility to start the service any time within a given time window.

1.6. Thesis Structure

In Chapter 2, the in-flight catering delivery model, its constraints and objectives are explained in detail. A detailed mathematical model of MTVRSPTW is developed. All operational constraints, decision variables and parameters are explained in detail where the goal is to minimise the number of loading teams required.

A relaxed-MTVRSPTW is introduced in Chapter 3. The number of trips performed by each loading team is fixed to two in this model, where the CG framework is considered. The problem input size ranges from 10 to 30 aircraft. An optimal solution is generated for each instance using the CPLEX search engine. The purpose of this chapter is to illustrate the computational effort of generating optimal solution.

In Chapter 4, a pre-processing algorithm called Time Window Reductions (TWR) is developed with the goal of spreading the aircraft maintenance demand more evenly throughout the planning horizon. A simple example is used to illustrate the time window reduction iterative process in the chapter

Chapter 5 presents the Insertion Algorithm for the flexible shift rosters. The implementation of the heuristic is illustrated for the model containing the multi-trip travelling and meal break requirement.

In Chapter 6, a Two-stage Scheduling Heuristic is proposed in order to solve an MTVRSPTW problem which is used to model the in-flight catering delivery system. The proposed algorithm is described with a specific focus on implementation of loading teams' synchronisation. In addition, the demand pattern of in-flight maintenance is studied through the analysis of data from industry.

Then, we perform a numerical study of the heuristic solutions, the Insertion Algorithm and the two-stage scheduling procedure with a range of data in Chapter 7. The comparison of the different approaches and an extensive sensitivity analysis are carried out in order to demonstrate MTVRSPTW the effectiveness and robustness of the proposed heuristics. The numerical results and discussion are also reported at the end of the chapter.

Finally, Chapter 8 contains concluding remarks, our contribution and ideas for future research.

Chapter 2:
Problem Description: definitions and mathematical formulation

2.1. Introduction

Aircraft maintenance and in-flight catering have become crucial functions in the planning and scheduling of airline operations and services. Due to the high volume of scheduled flights and tight time windows, efficient scheduling is essential to ensure the availability of resources. Aircraft maintenance companies are normally required to meet certain deadlines that have been committed to customers. Failure to do so may result in a significant loss. Therefore, effective scheduling has become paramount for survival in this competitive environment.

In this research, manpower scheduling is studied, particularly on in-flight catering delivery. Three weeks of fieldwork was conducted at both Kuala Lumpur International Airport (KLIA) and Kuching International Airport (KIA) in Malaysia. Real data was primarily collected through observing the behaviour and duties of people located within teams and management. Particularly, we were located in the planning department in order to get an in-depth understanding of current rostering practices and standard procedures of maintenance activities and emergency planning. The weekly aircraft movement is fairly static, except for peak seasons. For example, a loading team might serve the same aircraft on a particular day weekly. A five-day data summary was studied and compared to the proposed heuristic solutions in the later chapters. In a similar application, Ho and Leung (2010) collected a week's data to analyse a Tabu Search solution technique for studying a crew scheduling problem. Dohn et al (2009) conducted a 24-hour day manpower allocation case study at two different airports. Among other applications which have been considered is a six-day real data study by Battara et al (2009) focusing on supermarket distribution operations, and a five-day data study by Ren et al (2010) on a healthcare routing problem.

In our case study, a group of loading teams is required to deliver and upload packaged meals by using loading trucks from the ground kitchen to aircraft at the apron. All aircraft must be served within its time window considering not only the Standard Time of Arrival (STA) and Standard Time of Departure (STD) for the different aircraft, but also truck capacity. There are three types of flight patterns:

- Turnaround (*flights arrive and depart within the same day, requiring offloading meal carts and uploading new meals within a short time window.*)
- Inbound (*flights arrive on the day but will depart in following day, requiring offloading empty meal carts only.*)
- Outbound (*flights arrived in an earlier day and depart on the day,* r*equiring uploading new meals only*.)

The number of loading teams and aircraft service times varies according to flight patterns, plane type and size (wide, narrow or cargo), as shown in Table 2.1. Some wide body aircraft require more than one loading team to carry out the operation, as shown in the first column. Therefore, synchronisation of loading teams is considered in this case study. This means different loading teams must conduct the service to a particular aircraft at the same time. Service time in the last column includes subtasks such as (1) standby for passenger disembarkation at tarmac, (2) equipment exchange and (3) truck manoeuvring between the aircraft galleries. In this study, we assume that each loading team can provide service to all aircraft types.

Table 2.1: Flight standard servicing time classification

No. of loading teams required	**Aircraft Body Type**	**Flight Pattern**	**Offload time/team (minutes)**	**Upload time/team (minutes)**	**Service Time (minutes)**
4	Wide	Outbound	0	20	25
3		Turnaround	10	20	45
2		Inbound	10	0	20
1	Narrow	Outbound	0	15	15
		Turnaround	10	15	25
		Inbound	10	0	10
1	Cargo	Outbound	0	5	10
		Turnaround	5	5	10
		Inbound	5	0	0

The loading truck serves a limited number of aircraft in a single trip due to limited truck capacity. In addition, there is a limit on the exposure time of food since it must be

delivered and stored in the aircraft within a certain amount of time after being loaded in the kitchen.

In our case study, common labour policies are considered, such as the maximum working hours in a day and a mandatory hour meal break, which must be scheduled during a given time interval in the working shift. For example, a meal break is given between the third and fifth hours for an eight hour shift. Companies are doing their utmost to ensure worker welfare after the reputation-damaging incident at Apple's supplier, Foxconn. The workers there were required to work 11 daily hours with a 30-minute allocation for both lunch and dinner (Fiona, 2010). 14 suicide attempts were reported (Fiona, 2010) at Foxconn's Shenzhen plant in six months; twelve of those attempts resulted in death. Workers and corporate culture consultants claimed that the strict management style and inhumane work environment were the two main factors in the incidents.

This case study can be classified as deadline scheduling problem since each job must be executed before its deadline. We used deadlines rather than due-dates in this study due to the high penalty incurred by delayed jobs. The consequences of delay, such as missing the pre-assigned departure time slot, would affect the following connecting flights, scheduled crews, parking bays allocation, flights' take-off queue and so on. Thus, the catering company is willing to assign additional teams in order to avoid the possibility of not meeting customer deadlines. Since manpower is one of the major operation costs to the company, the primary objective function of this study is minimizing the number of loading teams, fulfilling all other operational constraints.

With dozens of teams and hundreds of tasks, the planning process quickly becomes immense and the use of operations research methods can save the companies both time and money. Therefore, the core puzzle of this research is to synchronise the loading teams, establishing start-times for the different flights and creating a roster for loading teams.

2.2. Problem Formulation

Due to the existence of operational constraints which affects the decision of making multiple trips travelling, such as truck capacity and food exposure time limits, in-flight catering delivery can be modelled as MTVRSPTW-MB.

In this case study, the aircraft play the role of customers in MTVRSPTW-MB formulation. A *trip* is defined as a cycle of travelling which begins at the kitchen, servicing one or more aircraft, then returned to the kitchen; meanwhile a *route* is a working shift

assignment for a loading team made up of one or more trips, with a meal break allocation in between. Examples of a route and trips are illustrated as Figure 2.1.

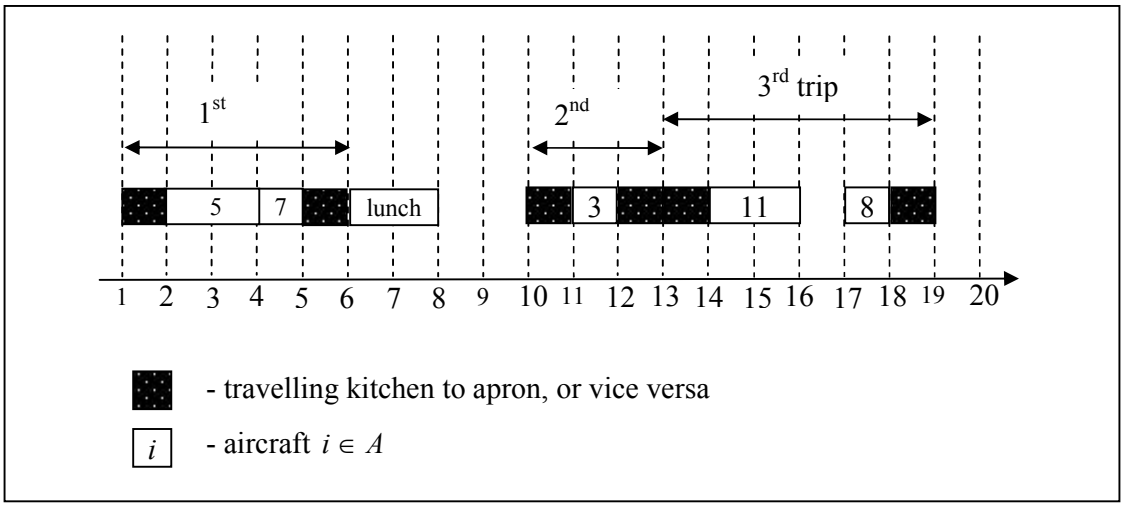


Figure 2.1: Graphical representation of a route

In-flight catering delivery can be mathematically defined as a directed graph, *G,* with a set of task nodes $J = A \cup H \cup L$ and identical travel distances are associated with every arc $(i, j) \in N$ where N is the arc set and $i, j \in J$. The nodes are defined as follow:

a) Aircraft service nodes, $A = \{1,2,\cdots,n\}$

Each aircraft node, $i \in A$ has

- a weight, which is servicing time of aircraft *i*, p_i
- a time window $[a_i, b_i]$, where a_i is the earliest starting time and b_i is the latest ending time
- a number of loading teams required to carry out the operation, r_i

b) Kitchen nodes are represented by $H = \{n+1, n+2, \cdots, n+2m-1, n+2m\}$, where *m* is the maximum number of trips allowed for each team.

Each kitchen node, $i \in H$

- has a weight, which is travelling time to and/from the kitchen, $p_{i \in H} = f$

- $\{n + 2b - 1\}$ or "odd h" represents the travelling time from the kitchen to the apron for uploading food to an aircraft, where $b = \{1, \cdots, m\}$
- $\{n + 2b\}$, or "even h" represents the travelling time from the apron back to the kitchen after offloading the meals, where $b = \{1, \cdots, m\}$

c) Lunch node, $L = \{n + 2m + 1\}$

- has a weight, which is the duration of the lunch break, $p_{n+2m+1} = P_{lunch}$
- has a time window, $[L_s, L_e]$ where L_s is the least working hours and L_e is the maximum working hours before lunch.

2.2.1. Notation

The following notation is used in our mathematical modelling.

2.2.1.1. Sets

A : set of aircraft = $\{1, \cdots, n\}$

B : set of the aircraft' split-tasks = $\{1, \cdots, s\}$

H : set of kitchen nodes = $\{n + 1, \cdots, n + 2m\}$

L : set of lunch nodes = $\{n + 2m + 1\}$

J : set of tasks to be assigned to loading team k, where $J = A \cup H \cup L$

K : set of loading teams = $\{1, \cdots, b\}$

U : set of travelling trips by each loading team = $\{1, ..., m\}$

Set B represents the set of aircraft' split-tasks that need catering services in a day, where each flight $i \in A$ requests a number of loading teams manpower, r_i. When the tasks are split according to the number of loading teams required, s is defined as total tasks in set B, $s = \sum_{i \in A}^{n} r_i$. In set U, the maximum travelling trips is limited to m trips. However, a route could end right after lunch or a few more trips after lunch, as long as the total trips before and after lunch is less than or equal to m.

Each "working" loading team is given a meal break but it is unnecessary to have a distinct lunch node for each loading team. This is because each node (except aircraft nodes) is

allowed to be assigned maximum once in each route/each loading team as in Eq. (2)-(3). In addition, Eq. (19) is defined to enforce obligatory meal break if a loading team *k* is "working". By defining one lunch node in set *L*, we could greatly reduce the size of variables in the model.

2.2.1.2. Parameter

e = end time of planning period

C = truck capacity/aircraft

F = food exposure time limit

r_i = number of loading teams required by flight $i \in B$

p_i = servicing time of task $i \in J$

p_s = smallest servicing time of aircraft

L_s = minimum working hours before lunch

L_e = maximum working hours before lunch

$[a_i, b_i]$ = time window for aircraft $i \in A$

f = travelling time from kitchen to apron or vice versa

P_{lunch} = lunch break duration

T = maximum working hours for each loading team

m = maximum trips allowed for each loading team, $\left\lfloor \frac{T}{p_s + 2f} \right\rfloor$

b = upper bound on the number of loading teams, e.g. set $b = n$

M_1 = $2F$

M_2 = $2L_s$

2.2.1.3. Decision variables

In terms of scheduling, we define a binary/Boolean decision variable for the allocation of loading team $k \in K$ to a pair of consecutive tasks *i* and $j \in J$, as:

$$x_{i,j}^k = \begin{cases} 1, \text{ if loading team } k \text{ serves task } j \text{ followed by } i; \\ 0, \text{otherwise.} \end{cases}$$

It is also important to keep track of the start time to conduct each task. A decision variable w_i^k is defined as feasible service start time in order to schedule tasks $i \in J$ in proper

sequence so that loading team k can perform a task at a time. On the other hand, we assign aircraft servicing tasks to loading teams based on two binary decision variables, particularly on trip and team assignment as follow:

$$z_{i,q}^{k} = \begin{cases} 1, & \text{if } \exists\, i \in \text{A is assigned to loading team } k \in K \text{ at trip } q\text{th;} \\ 0, & \text{otherwise.} \end{cases}$$

$$y^{k} = \begin{cases} 1, & \text{if } \exists\, i \in A \text{ being assigned to loading team } k \in K; \\ 0, & \text{otherwise.} \end{cases}$$

2.3. Mathematical Modelling

Two models are considered to tackle two different assumptions. The first model is a general version of the MTVRSPTW-MB, based on our case study, without considering manpower requirement variable $r_i, i \in A$. This model is equivalent to the assumption that each aircraft requires exactly one loading team. The second model accommodates the constraint of loading teams synchronisation r_i for each aircraft i in set A.

The travel times are associated with the nodes in this study, as the model is developed based on our case study. In the case study, we consider only two main locations: apron (where aircraft are landed) and kitchen. The travel time between aircraft is minimal and identical that can be included in aircraft's processing time/servicing time. Then, the travel time between kitchen to aircraft is associated with kitchen node as its processing time.

2.3.1 General Model

In this model, we assume that each aircraft requires exactly a loading team to upload the in-flight food. Every loading team is capable of conducting the operation. The objective of this model as illustrated in Eq.(1) is to minimise the number of loading teams required to serve all the aircraft in set A.

$$\text{Minimise } \sum_{k \in K} y^{k} \tag{1}$$

s.t. all equations in sections (3.1.1), (3.1.2) and (3.2).

2.3.1.1 Sub-graph

In this section, all constraints that are related to an individual loading team are defined based on the sub-graph, as shown in Figure 2.2. Since MTVRSPTW-MB is not a network flow problem, Figures (2.2)-(2.9) are for illustrative purposes for the formulation of the model.

There are a few important entities in this sub-graph illustrating the relation between set *A*, *L* and *H* as follows:

a) Set *L* (lunch) has direct links only to set *H* (kitchen), but not to set *A* (aircraft). It ensures that the loading team can only have meal break after returning to the kitchen. This is also for safety and security reasons where crews are not allowed to stay at the apron except when conducting permitted aircraft maintenance operations.

b) Set *L* has an incoming flow from even $h \in H$ and an outgoing flow to odd $h \in H$. This is because the meal break is allocated exactly once during working shift. There is no outgoing flow when no more aircraft need to be served after the loading team's meal break.

c) In set *H*, each odd node $\{n+2b-1\}$works in a pair with $\{n+2b\}$, where $b=\{1,\cdots,m\}$ to complete a travelling cycle as a trip. In between the pairs, loading teams will provide catering services to one or more aircraft (set *A*) landed at the apron.

d) Set *A* has incoming flows from odd $h \in H$ and outgoing flows to even $h \in H$. The number of incoming flows or outgoing flows is equivalent to the number of trips in that particular route.

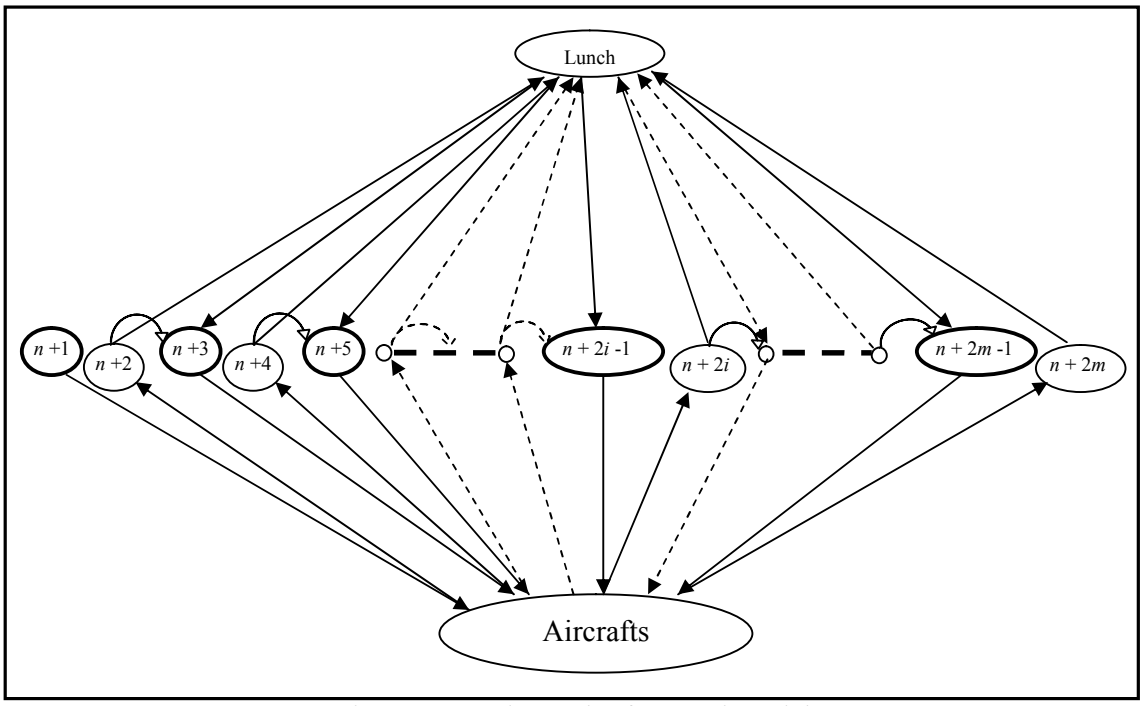


Figure 2.2: Sub-graph of general model

The operation constraints are described as following:

a) For each node $i \in J$ and $k \in K$,

$$\sum_{j=1}^{n+2m+1} x_{i,j}^{k} \le y^{k} \tag{2}$$

$$\sum_{j=1}^{n+2m+1} x_{j,i}^{k} \le y^{k} \tag{3}$$

$$x_{i,i}^{k} = 0 \tag{4}$$

Each node $i \in J$ is served only once by each loading team in Eq. (2) – (3). No self looping is allowed at any node, as in Eq. (4).

b) *Aircraft constraint*: For each node $i \in A$ and $k \in K$,

$$\sum_{q \in U} z_{i,q}^{k} \le y^{k} \tag{5}$$

$$\sum_{q \in U} z_{i,q}^{k} = \sum_{j=1}^{n+2m} x_{i,j}^{k} \tag{6}$$

$$\sum_{j=1}^{n+2m} x_{i,j}^{k} = \sum_{j=1}^{n+2m} x_{j,i}^{k} \tag{7}$$

Eq. (5) – (6) illustrate that each aircraft $i \in A$ is served at most once and only by a single loading team at a particular trip, q. In addition, each aircraft node $i \in A$ is restricted to obey flow balance laws, i.e. the amount of commodity flowing into it equals to the amount flowing out of it, as in Eq. (7). The flow balance law of each aircraft node $i \in A$ is graphically represented in Figure 2.3.

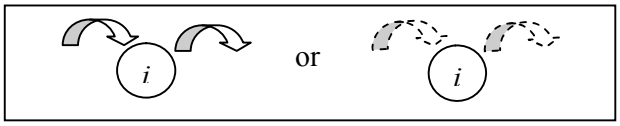


Figure 2.3: Flow balance of aircraft node $i \in A$

c) *Aircraft and Kitchen constraint*: For each node $k \in K$,

$$\sum_{j=1}^{m}\sum_{i=1}^{n} x_{n+2j,i}^{k} = 0 \tag{8}$$

$$\sum_{j=1}^{m}\sum_{i=1}^{n} x_{i,n+2j-1}^{k} = 0 \tag{9}$$

$$\sum_{j=n+1}^{n+2m+1} x_{n+2b-1,j}^{k} = 0, \qquad \forall b \in \{1,\cdots,m\} \tag{10}$$

$$\sum_{j=n+1}^{n+2m+1} x_{j,n+2b}^{k} = 0, \qquad \forall b \in \{1,\cdots,m\} \tag{11}$$

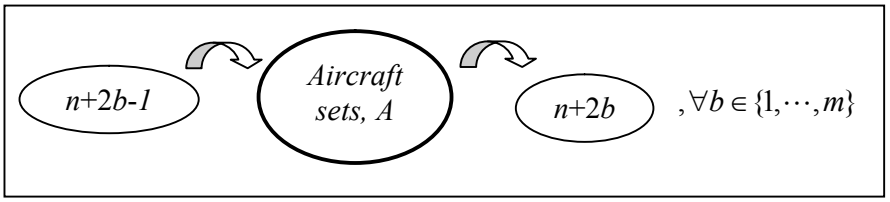


Figure 2.4: Network flow to/and from aircraft set, *A*

Figure 2.4 shows incoming/outgoing flow to/from aircraft sets, *A,* to form a trip. Eq. (8) prohibits the incoming flow from even $h \in H$ (travelling from apron to kitchen) to aircraft node $i \in A$. Meanwhile, flowing out from odd $h \in H$ (travelling from kitchen to apron) to aircraft node $i \in A$ is forbidden by Eq. (9).

On the other hand, outgoing flow from odd $h \in H$, n+2b-1, $\forall b \in \{1,\cdots,m\}$ is restricted to aircraft set, A by Eq. (10), as in Figure 2.4. Eq. (11) allows only the flow from aircraft sets, A to even $h \in H$, n+2b, $\forall b \in \{1,\cdots,m\}$.

d) *Kitchen constraint*: For each node $k \in K$,

In the sub-graph, kitchen node $n+1$ performs as a source that represents the first trip travelling from kitchen to apron. If outgoing flow does not exist at node $n+1$, both incoming flow and outgoing flow are prohibited at all other nodes $i \in J$, as illustrated in Figure 2.5. In this case, the loading team is not assigned any delivery work to aircraft.

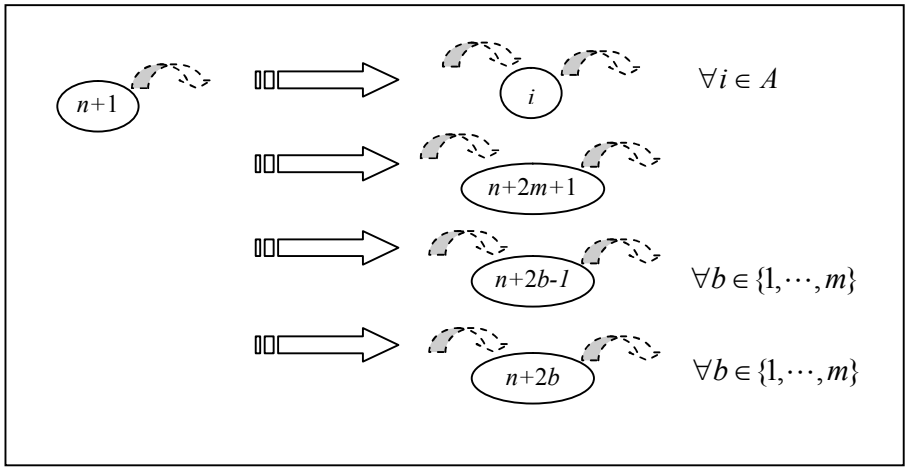


Figure 2.5: Network flow of unassigned loading team

$$\sum_{j=1}^{n+2m+1} x_{j,n+1}^{k} = 0 \qquad (12)$$

Since a source is supposed to have more outgoing flow, Eq. (12) restricts no incoming flow at node $n+1$. This is due to each node $i \in J$ being limited to having only one outgoing flow.

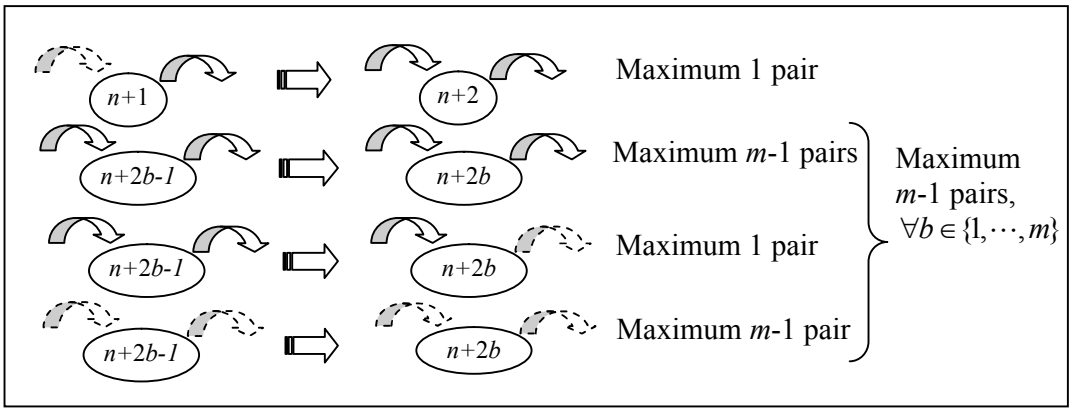


Figure 2.6: Flow balance of kitchen node $h \in H$

$$\sum_{j=1}^{n} x_{n+2b-1,j}^{k} = \sum_{j=1}^{n} x_{j,n+2b}^{k}, \quad \forall b \in \{1,\cdots,m\} \tag{13}$$

Once a loading team is assigned delivery work, the commodity flow restriction of the kitchen node $h \in H$ is shown, as Figure 2.6. According to the figure, kitchen nodes $h \in H$ must work in pairs in sequencing order, For example, n+1 is paired up with n+2, while n+3 pairs with n+4. This can be achieved by Eq. (13).

$$\sum_{j=n+2}^{n+2m+1} x_{j,n+2i-1}^{k} = \sum_{j=1}^{n} x_{n+2i-1,j}^{k}, \qquad \forall i \in \{2,\cdots,m\} \tag{14}$$

In Figure 2.6, each odd $h \in H$, except n+1 ought to follow the flow balance law as in Eq. (14).

e) *Kitchen and lunch constraint*: For each node $k \in K$,

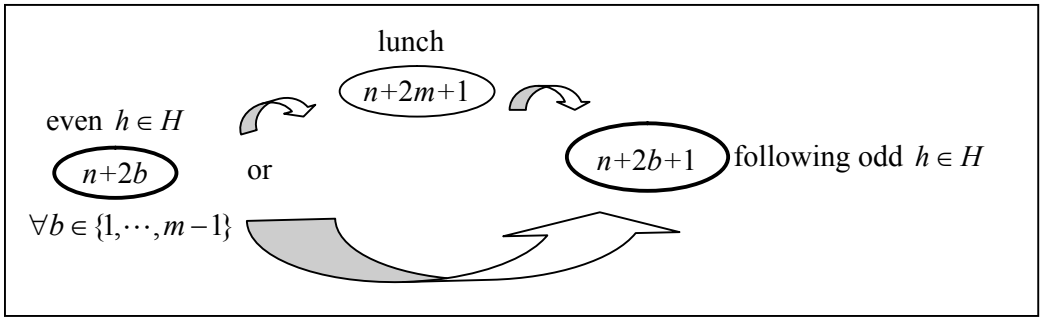


Figure 2.7: Network flow between lunch and kitchen nodes

$$\sum_{j=n+1}^{n+2m+1} x^k_{n+2i,j} \le \left(x^k_{n+2i,n+2i+1} + x^k_{n+2i,n+2m+1}\right), \qquad \forall i \in \{1,\cdots,m-1\} \tag{15}$$

$$\sum_{j=1}^{n+2m} x^k_{n+2m,j} = 0 \tag{16}$$

$$\sum_{j=n+1}^{n+2m+1} x^k_{j,n+2i+1} \le \left(x^k_{n+2i,n+2i+1} + x^k_{n+2m+1,n+2i+1}\right), \qquad \forall i \in \{1,\cdots,m-1\} \tag{17}$$

Figure 2.7 illustrates the network flow between kitchen nodes $h \in H$ and lunch node as follow:

- Allow outgoing flow from even $h \in H$ to follow either $h \in H$ or lunch node as in Eq. (15). For the special case of last even $h \in H$, the outgoing flow is restricted only to lunch node as in Eq. (16).
- Allow only incoming flow from either lunch node, i.e. n+2m+1 or even $h \in H$ to follow odd $h \in H$ as Eq. (17).

In terms of scheduling, the team would not return to "earlier" trip after lunch break because the "earlier" trip has already one incoming flow as in Eq. (2)-(3). On the other hand, the team also would not go to a much "later" trip because lunch node is restricted to the next following kitchen node as in Eq. (8)-(9) and Eq. (13)-(16). Regarding the service start time, it will be discussed in later section.

f) *Lunch constraint*: For each node $k \in K$,

$$\sum_{i=1}^{n} x^k_{i,n+2m+1} + x^k_{n+2m+1,i} = 0 \tag{18}$$

$$\sum_{i=1}^{m} x^k_{n+2i,n+2m+1} = y^k \tag{19}$$

As illustrated in Figure 2.7, the lunch node can only be allocated between even and following odd $h \in H$. Therefore, Eq. (18) prohibits the network link to/from aircraft set, A from/to lunch node. Eq. (19) assures lunch break is given to each loading team.

g) *Sink constraint*: For each node $k \in K$,

$$\left[\sum_{i=2}^{m}\sum_{j=1}^{n} x_{j,n+2i}^{k} - \sum_{i=2}^{m}\sum_{j=n+3}^{n+2m+1} x_{n+2i,j}^{k}\right] + \left[\sum_{j=n+2}^{n+2m} x_{j,n+2m+1}^{k} - \sum_{j=n+3}^{n+2m-1} x_{n+2m+1,j}^{k}\right] = y^{k} \tag{20}$$

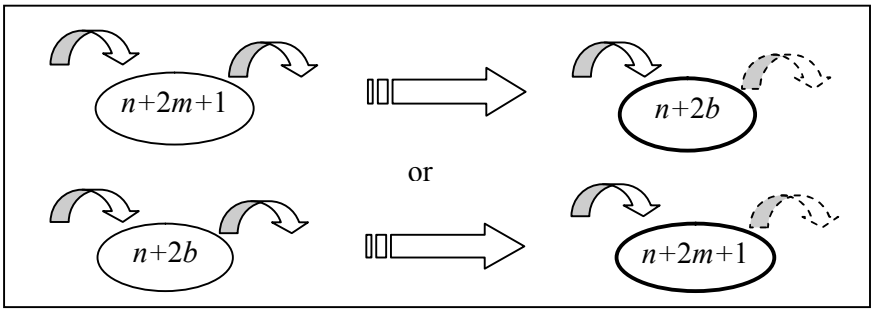


Figure 2.8: Network flow of sink node

Figure 2.8 demonstrates two possible types of sink nodes considered in this study. If the lunch break is allocated in between two consecutive trips, then the sink node is the last even $h \in H$ on the route. Otherwise, the lunch node, i.e. n+2m+1, will be the sink node in the case where there are no further catering operations assigned to the loading team after its meal break. This sink constraint is given by Eq. (20). In addition, Eq. (20) allows more incoming flow than outgoing flow from sink node, as illustrated in Figure 2.8.

h) *Time window constraint*: For each node $k \in K$,

$$w_i^k + p_i - w_j^k \le (1 - x_{i,j}^k)M_1, \quad \forall i, j \in J \tag{21}$$

$$a_i x_{i,j}^k \le w_i^k, \quad \forall i \in A, \forall j \in J \tag{22}$$

$$w_i^k \le x_{i,j}^k (b_i - p_i), \quad \forall i \in A, , \forall j \in J \tag{23}$$

Eq. (21) ensures the feasible start time of two consecutive service works. Time window restrictions of aircraft are satisfied by Eq. (22) and Eq. (23).

$$w_{n+i}^k \le e\sum_{j \in J} x_{j,n+i}^k, \quad \forall i \in \{2, \cdots 2m+1\} \tag{24}$$

$$w_{n+1}^k \le w_i^{\,k}, \quad \forall i \in J \tag{25}$$

$$w_i^k - w_{n+2q-1}^k - f \le (1 - z_{i,q}^k)M_1, \quad \forall q \in U, \forall i \in A \tag{26}$$

$$w_{n+2q}^k - w_i^k - p_i \le (1 - z_{i,q}^k)M_1, \quad \forall q \in U, \forall i \in A \tag{27}$$

$$\left(\sum_{j=1}^{n} x_{n+i+1,j}^{k} + \sum_{j=1}^{n} x_{j,n+i+1}^{k} - 1\right) 2M \le w_{n+i+1}^{k} - w_{n+i}^{k}, \qquad \forall i \in \{1,\cdots,2m-1\} \qquad (28)$$

Eq. (24) limits the service start time of kitchen nodes $h \in H$ and the lunch allocation falls within the planning horizon. Eq. (25) initiates the delivery work shift of a loading team by travelling from the kitchen to the apron. The service start time of each trip is satisfied by Eq. (26)-(27). Consecutively, Eq. (28) states that each kitchen node's service start time should increase according to its sequencing order.

$$w_{n+2m+1}^{k} - w_{n+1}^{k} \le y^{k} L_e \qquad (29)$$

$$L_s - \left(w_{n+2m+1}^{k} - w_{n+1}^{k}\right) \le \left(1 - y^{k}\right) M_2 \qquad (30)$$

$$w^{k}{}_{n+2i} - (w^{k}{}_{n+2i-1} + f) - F \le (1 - \sum_{j=1}^{n} x_{j,n+2i}^{k}) M_1, \qquad \forall i \in \{1,\cdots,m\} \qquad (31)$$

$$w_i^{k} \le w_{n+1}^{k} + T - p_i, \qquad \forall i \in J / \{n+1\} \qquad (32)$$

Eq. (29)-(30) assure that the lunch allocation is within its time window restriction. The food exposure time constraint and the maximum working shift requirement are fulfilled by Eq. (31) and (32), respectively.

i) *Truck capacity constraint*: For each node $k \in K$,

$$\sum_{i=1}^{n} z_{i,q}^{k} \le C y^{k}, \qquad \forall q \in U, \forall k \in K \qquad (33)$$

Eq. (33) limits in-flight food delivery of each trip, based on the truck capacity.

j) *Other constraints*: For each node $k \in K$,

$$w_i^{k} \ge 0, \forall i \in J \qquad (34)$$

$$y^{k} \in \{0,1\} \qquad (35)$$

$$x_{i,j}^{k} \in \{0,1\}, \quad \forall i, j \in J \qquad (36)$$

$$z_{i,q}^{k} \in \{0,1\}, \quad \forall i \in \{1,\cdots,n\}, \forall q \in U \qquad (37)$$

Eq. (34) – (37) define the binary, the integer, and non-negativity conditions on the decision variables.

2.3.1.2 Main-graph

$$\sum_{k\in K}\sum_{j=1}^{n+2m} x_{i,j}^{k} = 1, \qquad \forall i \in A \tag{38}$$

$$\sum_{k\in K}\sum_{q\in U} z_{i,q}^{k} = 1, \qquad \forall i \in A \tag{39}$$

Each loading team is given a set of kitchen nodes, H and lunch nodes, L, as shown in Figure 2.9. However, aircraft set, A, is shared by all the loading team. Eq. (38)-(39) assure that each aircraft is served by a loading team exactly once for a particular trip.

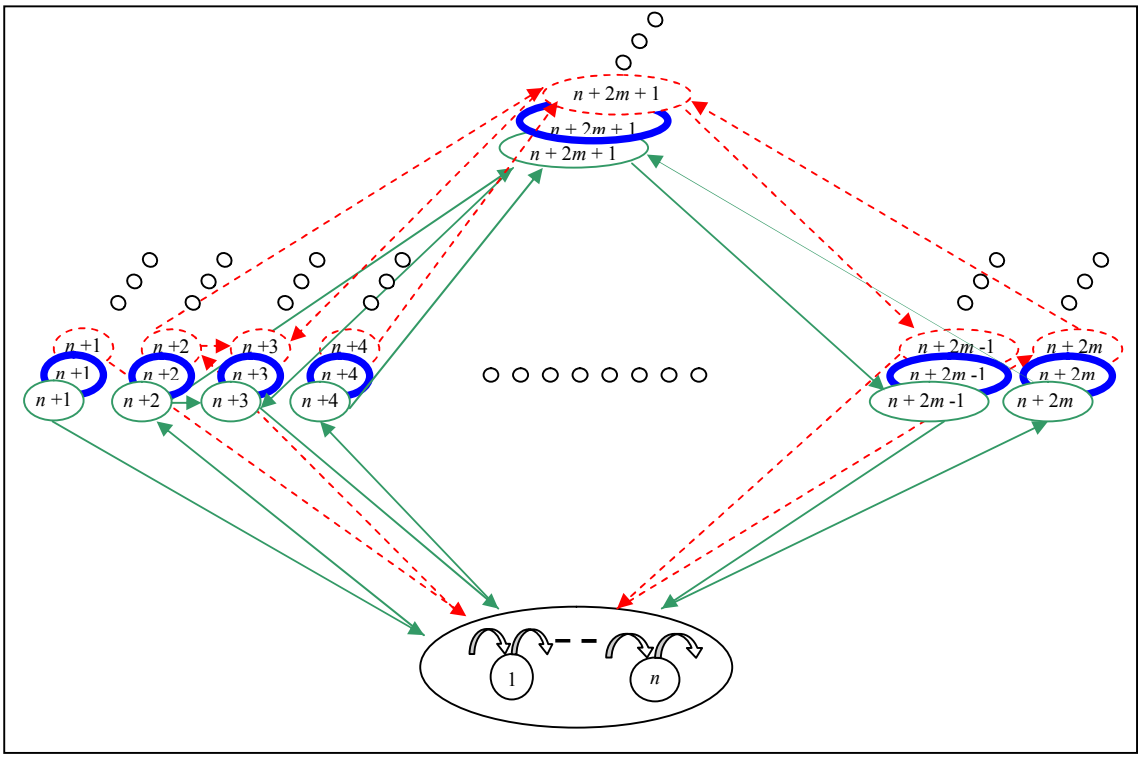


Figure 2.9: Main graph of Model A

2.3.2 Synchronisation of loading teams (Model B)

In this model, synchronisation of loading teams is considered. Thus, in order to serve the in-flight food to a particular aircraft, the loading teams assigned to the aircraft with the same start time and finishing time. Other operational constraints remain the same as in the general model in section 2.3.1. Model B can be mainly modelled by replacing some variances in the general model as follows:

Table 2.2: Differences between general model and Model B

General Model	Model B
Variable n	Variable s
Set A	Set B
Constraint (38)	Constraint (38')
Constraint (39)	Constraint (39')

where constraints (39') and (40') are defined as:

$$\sum_{k\in K}\sum_{j=1}^{n} x_{i,j}^{k} = r_i, \qquad \forall i \in B \tag{38'}$$

$$\sum_{k\in K}\sum_{q\in U} z_{i,q}^{k} = r_i, \qquad \forall i \in B \tag{39'}$$

Eq. (38')-(39') ensures the number of loading teams allocated are consistent with the manpower requirement of each flight, $i \in B$. The key constraint, synchronising the loading teams to have the identical service start time, w, is defined as Eq. (40).

$$w_i^k - w_i^{k'} \le \left(2 - x_{i,j}^{k} - x_{i,j'}^{k'}\right)e, \qquad ,\forall i \in A,\ \forall k \in K,\ \forall j, j' \in J,\ k \ne k' \tag{40}$$

2.4. Problem Complexity

The complexity of MTVRSPTW-MB has yet to be determined. This problem is more complex in nature in comparison to the MTVRSPTW model with an extra constraint for the meal break allocation.

Most of the past studies using MTVRSPTW were solved by applying heuristic or metaheuristic techniques. The exact solution of MTVRSPTW using a two-phase approach was first proposed by Azi et al. (2007), but was limited to a single vehicle variant of the problem. The computational results in this study conclude that the solution space increases dramatically if the time window or the trip travelling time limit is not tight enough. In Azi et al. (2010), the researchers extended the study to multiple-vehicle variant. The Branch-and-price algorithm was developed and evaluated in Solomon's (1987) test problems. In the study, the algorithm is not capable of solving all the 25-customer instances of Solomon's test

problem, even though it could solve optimally some of 40-customer instances. The computational time increases exponentially in small instances.

To all appearances, MTVRSPTW with large instances is unlikely to be solved within reasonable computational time, the same for MTVRSPTW-MB. This is because even in a reduced case where every truck is able to serve all the desired customers in a single trip, the problem is a NP-hard problem.

Chapter 3: Two trips VRPTW with meal break consideration

3.1. Introduction

As mentioned in Chapter 1, the Multiple Trips Vehicle Routing and Scheduling Problem with Time Windows and meal break considerations (MTVRSPTW-MB) is difficult to solve efficiently. In this chapter, we illustrate the difficulty in obtaining optimal solutions by solving a relaxed version of the problem. We limited the maximum number of trips to two for each route as shown in Figure 3.1, thus allowing for the meal break to be allocated between trips. By doing this, some of the operational constraints of original model as illustrated in Chapter 2 needed to be relaxed. Those constraints are truck capacity, synchronisation of loading teams, and the trip's limited travelling time. The loading teams are restricted to having their meal breaks at the service centre for the case study. In the case study, apron is considered a high-restriction area due to safety and security purposes. Therefore, loading teams are not allowed to conduct any activities besides the aircraft servicing work.

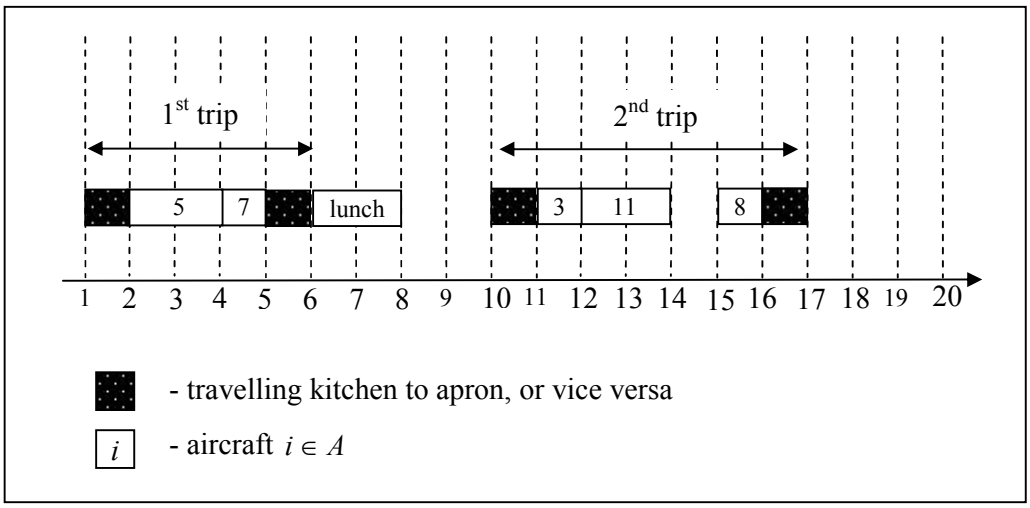


Figure 3.1. An example of two trips travelling in relaxed-MTVRSPTW-MB

3.2. Model Formulation and Description

As described in Chapter 2, for the MTVRSPTW we considered a set of customers, $A = \{1, \cdots, n\}$, which need to be served by a set of service teams, K. Each customer, *i,* has a processing time, p_i, and a time window, $[a_i, b_i]$. Each service team can only serve one

customer at a time and may proceed to the next customer after completing service for the current customer.

A mandatory meal break must be scheduled during a given time interval in the working shift. The teams are restricted to having their meal break at the service centre. Therefore, there are a total of two travelling trips for each working shift. Let l_{early} and l_{late} denote the minimum and maximum working hours in a shift before the compulsory meal break. A route is defined as a working shift assignment for a loading team, all completed within T time units.

We modelled the two trips VRPTW with meal break consideration (TTVRPTW-MB) as a graph, *G,* with a set of job nodes, $J' = \{1, \cdots, n, n+1, n+2, n+3\}$, where the meal break is denoted by $n+3$. The depot is denoted by $n+1$ and $n+2$. In this graph, identical travel distances are associated with every arc $(i, j) \in E$, where *E* is the arc set.

The following decision variables are used in TTVRPTW-MB:

- For each $i, j \in J'$ and $k \in K$,

$$x_{i,j}^k = \begin{cases} 1, \text{if service team } k \text{ serves customer } j \text{ followed by } i; \\ 0, \text{otherwise.} \end{cases}$$

- For each $k \in K$,

$$y^k = \begin{cases} 1, \text{if a route is assigned to service team } k; \\ 0, \text{otherwise.} \end{cases}$$

- For each $i \in J'$ and $k \in K$,

w_i^k = Service start time of customer *i* by service team *k*.

With *M* being an arbitrary large constant, the TTVRPTW-MB is formulated as follows:

$$\text{Min} \sum_{k \in K} y^k \qquad (1)$$

s.t.

$$\sum_{k \in K} \sum_{j \in J} x_{i,j}^k = 1, \qquad \forall i \in A \qquad (2)$$

$$\sum_{j \in J} x_{i,j}^k \le y^k, \qquad \forall i \in J', \forall k \in K \qquad (3)$$

$$\sum_{i \in J} x_{i,j}^k \le y^k, \qquad \forall j \in J', \forall k \in K \qquad (4)$$

$$x_{i,i}^{k} = 0, \qquad \forall i \in J', \forall k \in K \tag{5}$$

$$\sum_{j \in J} x_{j,i}^{k} = \sum_{j \in J} x_{i,j}^{k}, \qquad \forall i \in A \cup \{n+3\}, \forall k \in K \tag{6}$$

$$\sum_{i \in J} \left(x_{i,n+1}^{k} + x_{n+2,i}^{k} \right) = 0, \qquad \forall k \in K \tag{7}$$

$$\sum_{i \in A} x_{i,n+3}^{k} = y^{k}, \qquad \forall k \in K \tag{8}$$

$$w_i^k + p_i - w_j^k \le \left(1 - x_{i,j}^{k}\right) M, \qquad \forall i, j \in J', \forall k \in K \tag{9}$$

$$a_i x_{i,j}^{k} \le w_i^k, \qquad \forall i \in A, \forall j \in J', \forall k \in K \tag{10}$$

$$w_i^k \le x_{i,j}^{k} \left(b_i - p_i\right), \qquad \forall i \in A, \forall j \in J', \forall k \in K \tag{11}$$

$$w_{n+3}^k - w_{n+1}^k \le l_{late} y^k, \qquad \forall k \in K \tag{12}$$

$$l_{early} - \left(w_{n+3}^k - w_{n+1}^k\right) \le \left(1 - y^k\right) M, \qquad \forall k \in K \tag{13}$$

$$w_i^k \le w_{n+1}^k + T - p_i, \qquad \forall i \in J', \forall k \in K \tag{14}$$

$$w_i^k \ge 0, \qquad \forall i \in J', \forall k \in K \tag{15}$$

$$x_{i,j}^{k} \in \{0,1\}, \qquad \forall i, j \in J', \forall k \in K \tag{16}$$

$$y^k \in \{0,1\}, \qquad \forall k \in K \tag{17}$$

The solution of the above formulated mathematical program provides us with the minimal number of service teams required to serve all the customers, as defined by the objective (1). Each customer is served by any service team exactly once, as stated by constraint (2). Constraints (3) and (4) define the permitted flow for each customer to be assigned to "working" service teams. In other words, no customer is allowed to be assigned to excessive service teams due to the objective of minimising the number of service teams. Self-looping is prohibited by constraint (5). Constraint (6) assigns customers and meal breaks in a feasible sequence for each route. Constraint (7) ensures that each serving team departs from n+1 (depot) and returns to n+2 (depot) after providing service to customers. This can be done by prohibiting flow into *n*+1 and flow out from *n*+2, meanwhile maintaining network flow balance at other nodes as constraint (6). Constraint (8) ensures that a meal break is assigned for each service team during its shift. The main purpose of constraint (9) is to check the service start time of two consecutive tasks. If a node *j* is assigned after node *i* on a particular route, the service of node *j* (w_j^k) can only be conducted after the completion of

node i, i.e. $w_i^k + p_i$. Therefore, the service start time of node j has to be greater than the completion of node i. Constraints (10) and (11) define feasibility of the time schedule for all customers and the depot. Constraints (12) and (13) verify that the meal breaks are allocated during the allowable time intervals. Constraint (14) is the maximum working hour constraint for service teams. Constraints (15), (16) and (17) are the binary constraints on our decision variables.

3.3. Column Generation

Such a problem as TTVRPTW-MB introduces too many symmetries, such as too many identical route patterns with few different start times of customers generated on a same route. As a consequence, a heavy computational effort is needed to produce a significantly better solution.

Fortunately, for any given customer order, a limited number of route patterns will be sufficient, so many possible patterns can be disregarded, and the application can focus on finding the relevant ones.

Therefore, we reformulated the previous model as Y_r binary decision variable to indicate if route $r \in R$ is chosen, where R is the set of feasible routes. For a feasible route $r \in R$, define:

$$d_i^r = \begin{cases} 1, & \text{if node } i \text{ is visited on route } r; \\ 0, & \text{otherwise.} \end{cases}$$

We can then formulate the TTVRPTW-MB as a set covering problem as follows:

(Master Problem) $\quad \text{Min} \sum_{r \in R} Y_r \qquad (19)$

s.t.

$$\sum_{r \in R} \sum_{i \in A} d_i^r Y_r \geq 1 \qquad (20)$$

$$Y_r \in \{0, 1\} \qquad (21)$$

The objective (19) is to minimize the number of routes chosen, where a route is assigned to an individual service team. Constraint (20) ensures that each customer is visited

at least once. Constraint (21) is the binary constraint on the decision variable of the master model.

Solving this model with all route patterns from the beginning is practically impossible. In fact, even with only 10 customers with wide time windows and a servicing time $\frac{1}{10}$ of the working shift time, there would exist roughly 10^{10} kinds of route patterns, and hence that many decision variables. Such a formulation might not even fit in memory on a reasonably large computer. Moreover, most of those patterns would obviously not be interesting as a solution. These considerations make Column Generation (CG) an interesting approach for this problem. Furthermore, the computational effort required to solve the problem directly by branch and bound proves to be too large. Thus, the problem becomes computationally intractable even for small problem sizes, further motivating us to consider CG.

In solving TTVRPTW-MB by CG, one should start with a subproblem. Choose one route pattern (which could be as simple as one customer per route) subject to the operational constraints, such as time window and truck capacity. This procedure is likely to work in the sense that it produces a feasible solution to the problem. However, it will not necessarily produce a satisfactory answer and will probably use too many manpower resources.

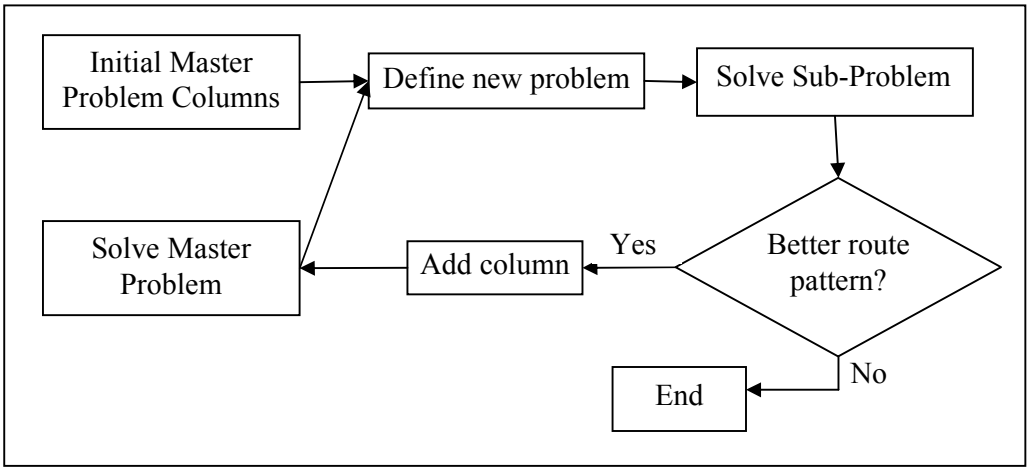


Figure 3.2. Flowchart for a general CG

To get closer to a satisfactory solution, the application can then generate other columns. That is, other decision variables (other Y_r) will be chosen and added to the model. Those decision variables are chosen on the basis of their favourable reduced cost through the

solution of a subproblem. This subproblem is defined to identify the coefficients of a new column of the master problem. A simplified flowchart of CG is shown in Figure 3.2.

The following decision variables are used in subproblem of TTVRPTW-MB. For each $i, j \in J'$:

w_i = Service start time of job i.

$$x_{i,j} = \begin{cases} 1, \text{if job } j \text{ is served and scheduled followed by } i; \\ 0, \text{otherwise.} \end{cases}$$

Both decision variables above have the same function to that described in Chapter 2, but leave the consideration of the assignment of routes to service teams as the master problem. This is due to the main purpose of subproblem in generating various route patterns (columns) to add in set R. Then, the master problem could choose the minimum number of routes (each route is equal as a service teams in our case) combination from set R. For each $i \in A$, another binary decision variable is defined to indicate if customer i is assigned on the route as:

$$d_i = \begin{cases} 1, \text{if customer } i \text{ is served on the route;} \\ 0, \text{otherwise.} \end{cases}$$

The subproblem is defined as follows:

$$\text{(Subproblem) Min } 1 - \left(\sum_i \pi_i d_i \right) \tag{22}$$

s.t.

$$d_i \le \sum_{j \in J} x_{i,j}, \qquad \forall i \in A \tag{23}$$

$$\sum_{i \in A} x_{n+1,i} \le 1 \tag{24}$$

$$\sum_{i \in J \subset n+1} x_{i,n+2} \le 1 \tag{25}$$

$$\sum_{i \in J} x_{i,j} \le 1, \qquad \forall j \in J \tag{26}$$

$$\sum_{j \in J} x_{i,j} \le 1, \qquad \forall i \in J \tag{27}$$

$$\sum_{i \in A} x_{i,n+3} \le 1 \tag{28}$$

$$w_{n+1} + l_{early} \le w_{n+3} \tag{29}$$

$$w_{n+3} \leq w_{n+1} + l_{late} \quad (30)$$

The subproblem is the problem to find a feasible route with minimum cost in R as defined by the objective (22). The vector π of dual variables is associated with the demand fulfilment constraint (20). When the reduced cost is positive, no negative reduced cost column exists and the algorithm terminates. Each customer is served, at most, once on a route, as stated by constraint (23). Constraints (24) and (25) ensure that each route starts from n+1 and ends at n+2. Constraints (26) and (27) define permitted inflow and outflow, at most, once for each node i in set J'. An obligatory meal break allocation is ensured by constraints (28) – (30) during the allowable time window. Furthermore, the subproblem is also subject to constraints (5)-(7), (9)-(11), and (14)-(17) by replacing decision variables w_i^k and $x_{i,j}^k$ with w_i and $x_{i,j}$, respectively.

3.4. Performance Analysis

The TTVRPTW-MB described in Chapter 3 was solved by using a generic mixed integer programming solver, ILOG OPL Development Studio IDE 6.0, with the ILOG CPLEX 11.1 solver.

Due to the complexity of the TTVRSPTW-MB, the problem input size was limited to 30 customers. All the data was simulated using a normal distribution to generate task processing times and time window length. Note that all customers may have distinct time windows but the number of different tasks is bounded. This is to reflect real-life practice where there are several airplane types, each consisting of a different set of tasks.

Table 3.1. Performance Solution of CPLEX

Number of customers	CPLEX	
	Number of teams	CPU time
10	2 teams	6 sec
15	3 teams	2,608 sec
20	6 teams	57,615 sec
25	7 teams	100,845 sec
30	9 teams	86,400 sec

Table 3.1 provides the number of teams and the time required to obtain solutions for different size problems. The TTVRPTW-MB solved by CPLEX is simulated for the case where W (the largest ratio of time window width to its processing time) is set to 5. Five different data sets ranging from 10 to 30 customers were solved and compared.

CPLEX proves to be very sensitive to the data set and requires exponentially increasing computation time as the number of tasks increases (over 15 hours are required to solve instances of 15 to 20 customers). Furthermore, the CPLEX algorithm (branch and bound in the CG framework) is a significantly complex solution to cope with even small problem instances.

Next, the shorter time window data was generated (when W = 1.5 or 2) in order to evaluate the sensitivity of time window's ratio in solving the TTVRPTW-MB. As demonstrated in Table 3.2, the computational time drops significantly with a decrease in the value of W. It is clear that the task of obtaining optimal solutions for the TTVRPTW-MB using CPLEX is unachievable, even with the use of very efficient computers. Thus, industries which dealt with hundreds and thousands of customers are required to develop alternative approaches. Even though CPLEX requires less time to generate the solution for smaller W values, real-life application will be poorly modelled if W is too close to 1.

Table 3.2. Time Window Sensitivity Analysis

Number of customers	CPLEX	W	CPU time (sec)
20	6 teams	5	57,615
		2	1,474
25	7 teams	5	100,845
		2	68,880
		1.5	184
30	9 teams	5	86,400
		2	74,280
		1.5	2,484

3.5. Conclusion

In this chapter, we formulated a relaxed version of MTVRSPTW-MB as a mathematical program bounded with two travelling trips for each service team. Then, Column Generation was proposed to tackle the modelling problem more efficiently.

The experimental result was generated by using CPLEX search engine and showed that even for small problem inputs, it is computationally intractable. The problem is also proved to be time window sensitive. Therefore, this solution is not able to cope with real-world application with large problem instances.

Chapter 4: Time Window Reduction

4.1.Introduction

One of the important characteristics of this case study is the highly fluctuated service demand pattern in a day. Figure 4.1 shows the average service demand (based on the Standard Arrival Time) for five days' data. Clearly, the in-flight service demands spread unevenly throughout a day. The busiest delivery occurs from 6am to 9am; meanwhile off peak occurs from midnight, 12:00 a.m. to 4:00 a.m.

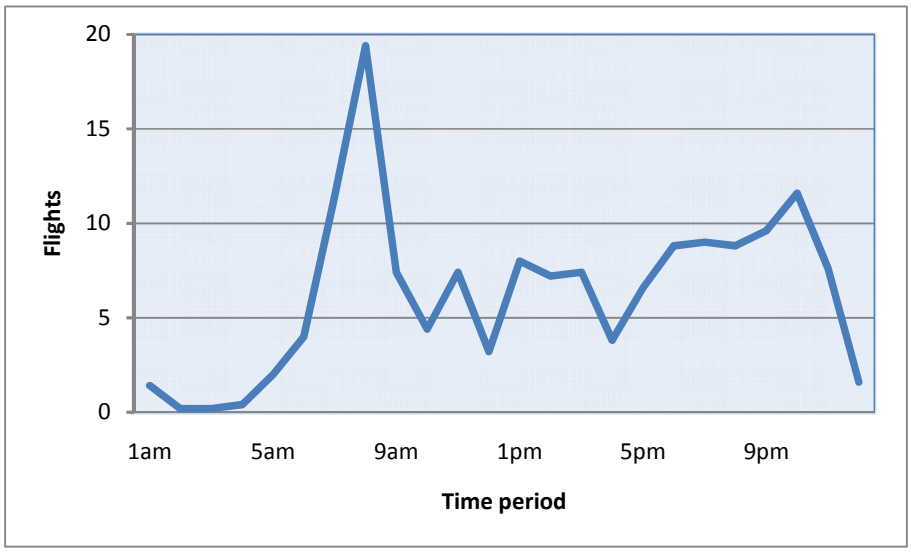


Figure 4.1: An average of five days in-bound flight distribution

Intuitively, schedule slips occur during peak demand periods. In order to avoid incurring penalties (e.g. compensation of flight delay) because of not meeting customer deadlines, the company is willing to assign additional loading teams to cope with the uneven workload. Due to this fact, the company plans its operation on the basis of one loading team per two flights, which creates excessive idle time between consecutive trips.

To reduce chances of a schedule slip, we propose an iterative time window reduction process which smoothen the demand pattern and eliminates the peaks. The rationale of the time window reduction process is to spread the jobs "as evenly as possible" throughout the planning horizon, by observing the frequency of overlapping time windows. The tasks contained in the high frequency time windows are then reviewed. Special attention is given to the jobs which have wide time windows and small processing times. The time windows of

this type of jobs are shortened by reducing the largest ratio of time window width to its processing time, *W*, with an attempt to avoid the peak periods. It is shown in Chapter 6 that TWR improves the efficiency of all of the heuristics developed in this research.

As in the example described in Figure 4.2, all the jobs whose time window overlaps at interval 6-7 will be examined. If a job has wide time window, TWR will shorten its time window to avoid interval 6-7. The reduction process will be repeated from the busiest to the least busy time interval, until all the jobs satisfied *M* constrained time window.

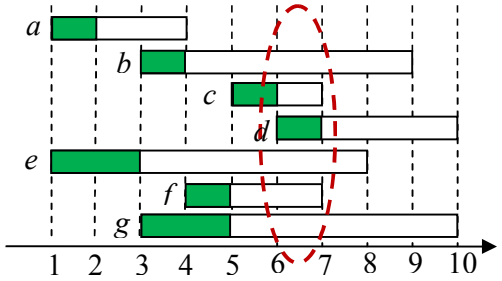


Figure 4.2: Time window of a set of jobs before TWR

Before the application of TWR, the sequence of jobs for allocation is *a – e – b – g – f – c – d* as illustrated in Figure 4.3(a). The allocation of jobs is shown in Figure 4.3(b). Two routes are required in this case.

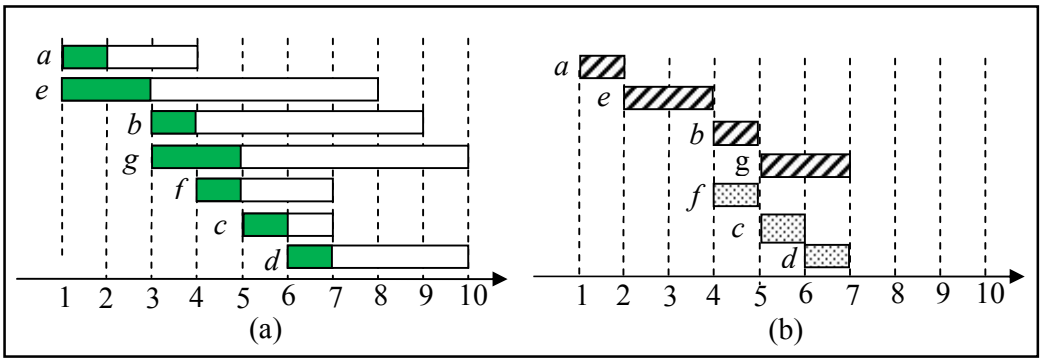


Figure 4.3: Sorted jobs and job allocation without TWR - two routes

Figure 4.4(a) shows the time windows of Figure 4.2 after TWR. The position of job *g* has changed after the application of TWR (*a – e – b – f – c – g – d*). This implies that jobs f and c are given higher priority than job g. This makes sense as job *g* has a wide time window spanning after the peak time period. In this case, all jobs can be served in a single trip as shown in Figure 4.4(b).

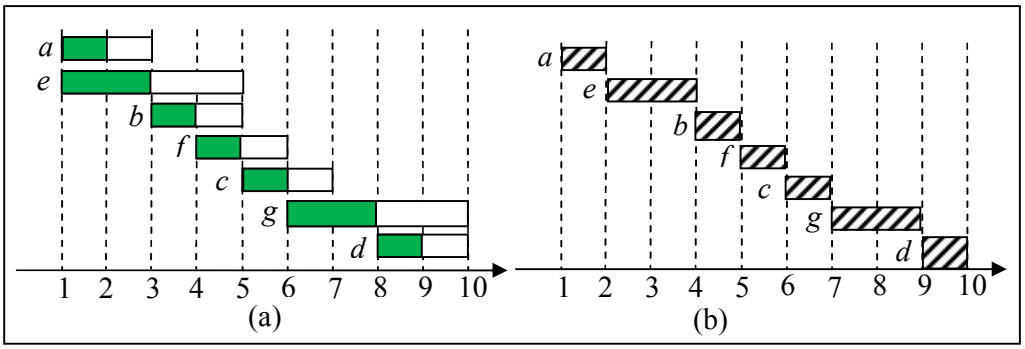


Figure 4.4: Sorted jobs and job allocation with TWR - 1 route

4.2. Time window reduction

Let W be the maximum ratio of time window width to its processing time allowed for all aircraft' split-tasks, defined as:

$$\frac{b_i - a_i}{p_i} \le W, \qquad \forall i \in B \qquad (1)$$

Let L be a set consisting of all time intervals for the planning time horizon. C_k is defined as a counter of tasks at time interval k, where two rules below must be fulfilled. Rule 1 and 2 ensure the correct counter of tasks involve at time interval k.

Rule 1: $r_i \le k$

Rule 2: $k \le d_i$

The algorithm for TWR is given below:

N = set of unchecked time slots

C_k = counter for number of customers that could be served at time slot k

while ($N \neq$ null) do the following:

1. Choose the highest C_k in N
2. For all customers i , in highest C_k and check $(k - r_i) - (d_i - k)$
   a. case 1: > 0

      if $\frac{k - r_i}{p_i} \ge W$, then new $d_i = k$

end if

b. case 2: < 0

if $\frac{d_i - k - 1}{p_i} \geq W$, then new $r_i = k + 1$

end if

c. case 3: = 0

if $\frac{k - r_i}{p_i} \geq W$, then

if $C_{r_i} + \cdots + C_{k-1} \geq C_k + \cdots + C_{d_i}$, then new $r_i = k$

else $d_i = k$

end if

end if

3. *end for*
4. $N = N \setminus k$
5. Update C_q for $\forall q \in Q$
6. *end while*

Figure 4.5 illustrates another example where TWR manages to reallocate and avoid aggregation of customers during time period 6 to 7. The value of *W* is set to 2. Initially, each customer with a time window overlapping with the interval between periods 6 and 7 will be divided into 2 parts, *x* and *v*. Partition *x* is on the left side of the peak period and partition *v* is on the right side. The partition that fulfils constraint (1) is selected. In case of a tie, the longer partition is chosen. For example, in first iteration, job *e* has 5 units of time in *x* partition and only 1 unit of time in *v* partition. The job requires 1 unit of time as procession time, Therefore, *x* partition will become the new time window for job *e* because it fulfils constraints (1) whereas *v* partition does not.

TWR is efficient in dealing with a large number of jobs with a combination of tight and wide time windows, especially in instances where both peak and off peak time periods coexist. This situation occurs in most real life situations.

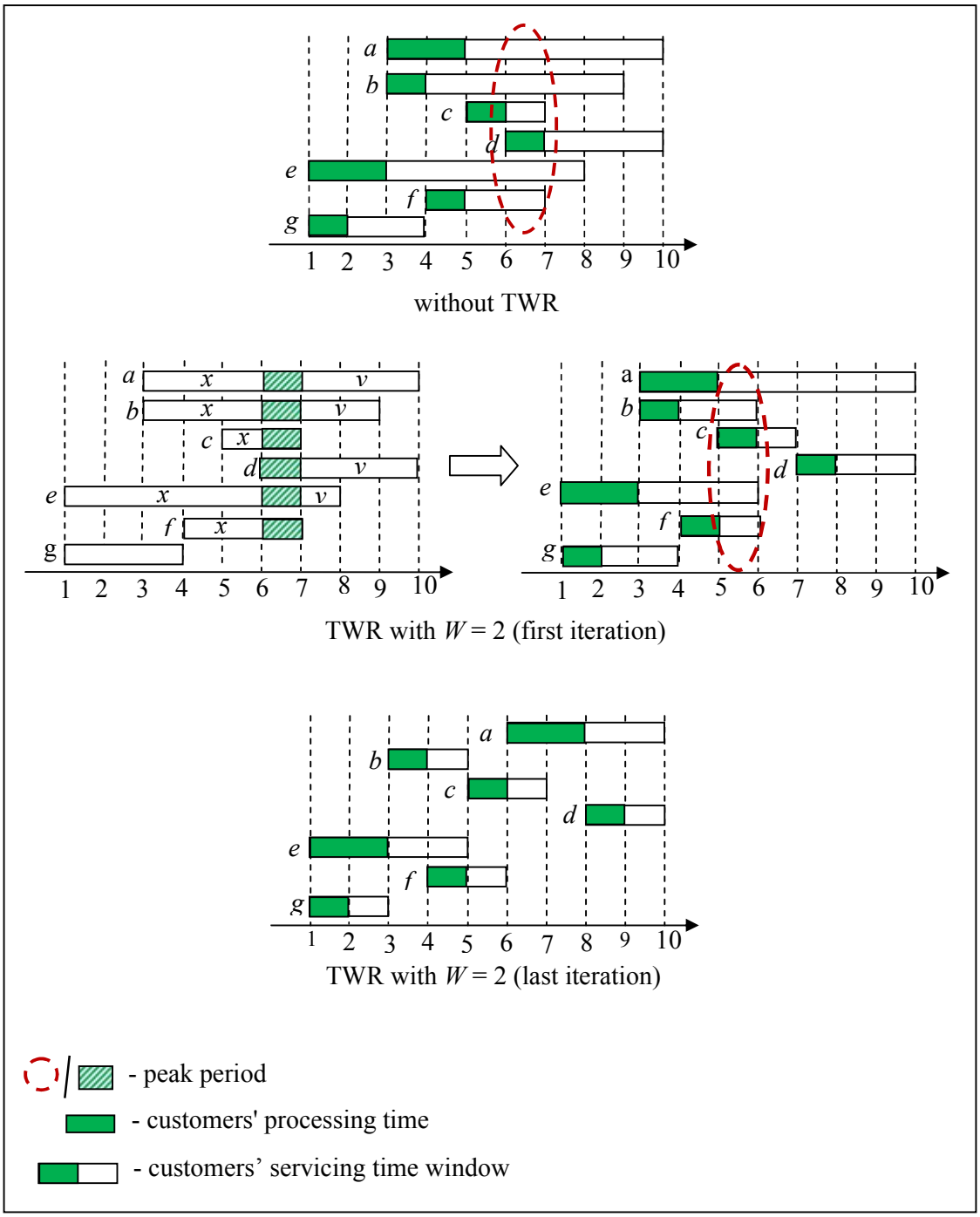


Figure 4.5: Time window reduction process

However, TWR can worsen the situation if all customers have wide time windows coupled with small processing times, as shown in Figure 4.6. Without TWR, all the jobs can be served in a single route. On the other hand, two routes are needed when TWR is applied, as demonstrated in Figure 4.7. Therefore, it is important to choose an appropriate W value using trial and error (from the maximum ratio of the original time window up to a value of 2).

It is important to observe that each execution requires approximately 0.4 seconds, even in the instance of 500 jobs.

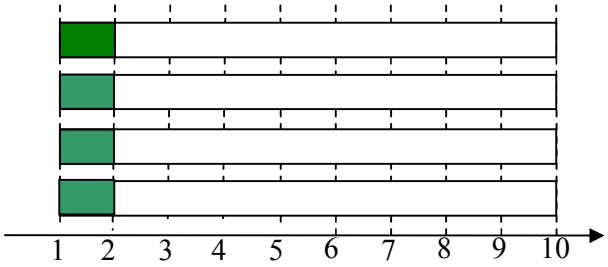


Figure 4.6: Wide time windows with small processing time

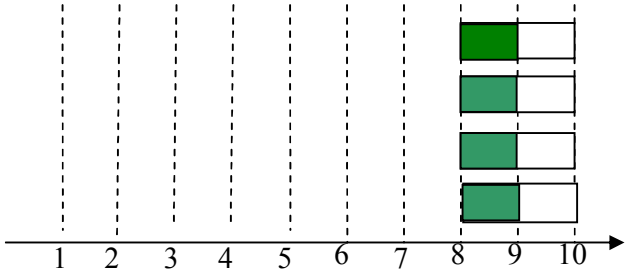


Figure 4.7: Time windows after TWR ($W = 2$)

4.3. Performance Analysis and discussion

Figure 4.8 illustrates part of the service demand of aircraft on day 1 in this case study. Clearly, the peak periods fall from time period 84 to 96. After applying TWR (with $W = 2$), the aircraft are spread much more evenly over that time interval as observed in Figure 4.9.

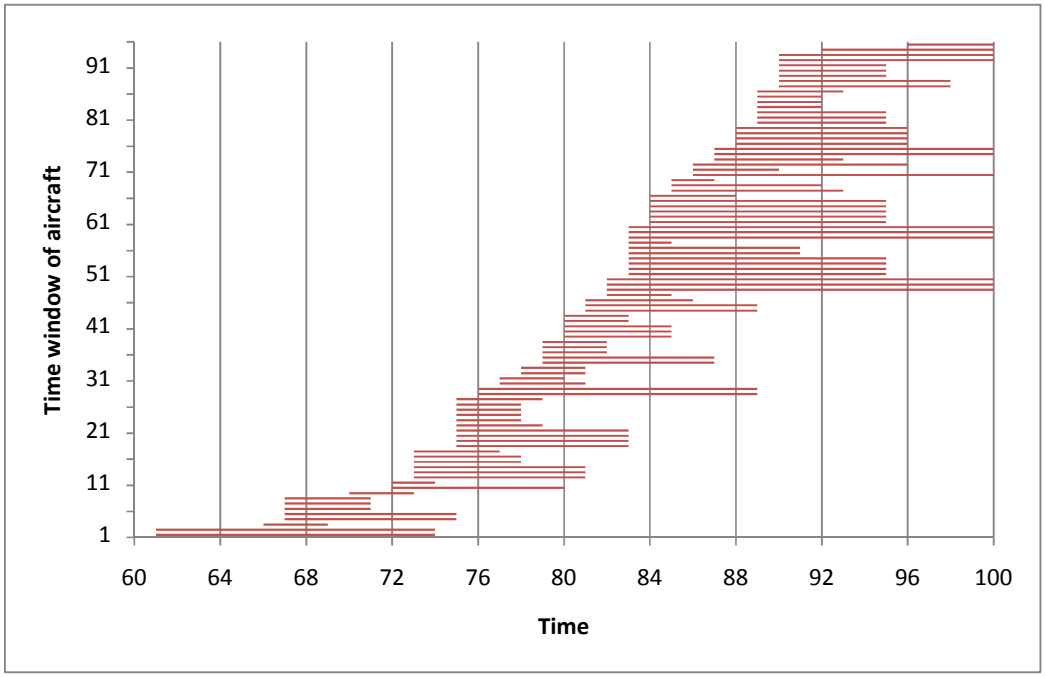


Figure 4.8: Original service demand of Day 1

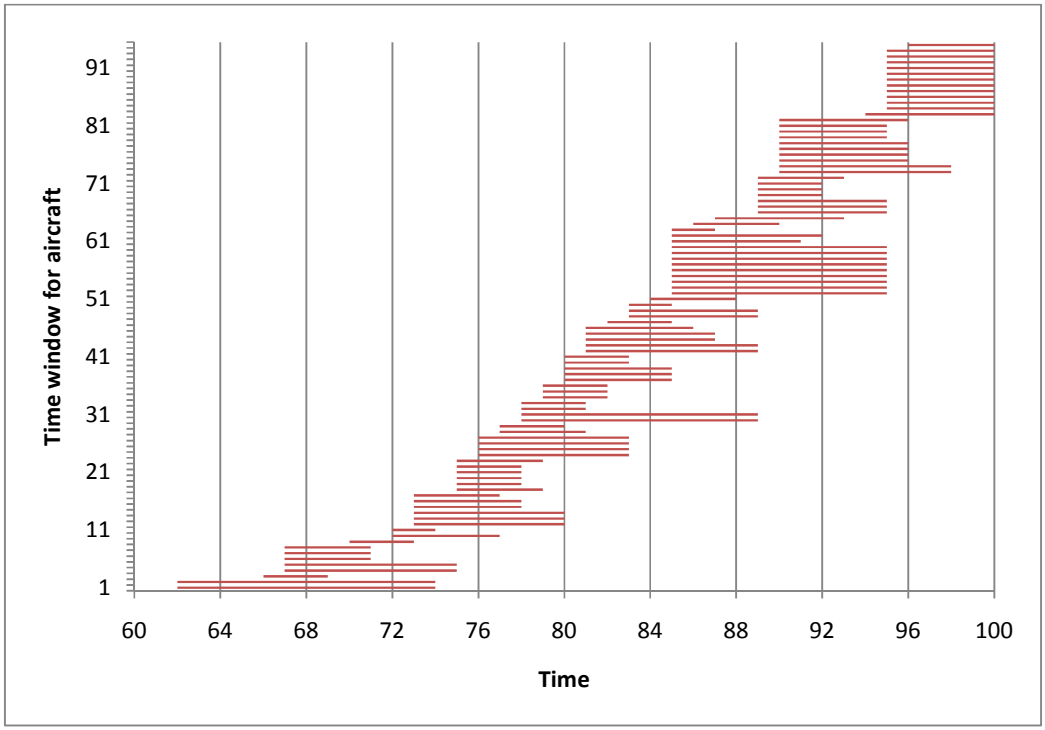


Figure 4.9: Service demand of Day 1 after TWR ($W = 2$)

4.4. Conclusion

TWR is developed as a pre-processing algorithm in order to spread the jobs as evenly as possible throughout the planning horizon. This procedure allows to modify the job processing order which increases the flexibility in allocating the jobs to loading teams. Due to the fact that most industries experience frequent peak and off peak time periods coupled with varying time windows, the suggested approach is highly applicable.

Data containing a whole day's service demand from one of the in-flight catering companies is used to illustrate the applicability of our approach in a real life setting. We show that demand can be smoothed out and that peak demand is significantly reduced. It is important to emphasize that TWR's computing time is less than 1 second, even for a data set containing 500 instances.

Chapter 5: A MTVRSPTW with Meal Break Consideration

5.1.Introduction

In this chapter, we present an insertion-based heuristic to solve Multiple Trips Vehicle Routing and Scheduling Problem with Time Windows and meal break consideration (MTVRSPTW-MB). Insertion heuristics are proven to be popular techniques for solving complicated combinatorial optimization problems with low computational effort. As a result, many commercial routing and scheduling software packages use this heuristic as their solution approach (Palmer et al, 2004).

The insertion algorithm was first implemented in 1977 to solve travelling salesmen problems (Campbell and Savelsbergh, 2004). In the following years, the insertion heuristic was widely used to solve VRP variants, such as multiple vehicle dial-a-ride problems (Jaw et al, 1986), VRPTW (Solomon, 1987), asymmetric capacitated VRP (Vigo, 1996), and a real-life application of transporting handicapped persons (Toth and Vigo, 1997). Furthermore, the insertion technique is also a typical choice in developing an initial solution for metaheuristic approaches, such as Tabu Search and Simulated Annealing (Bräysy and Gendreau, 2005).

Hunsaker and Savelsbergh (2002) and Campbell and Savelsbergh (2004) present an effective implementation of the insertion heuristic in a low complexity algorithm.

In recent works, Dessouky et al (2003) embedded an environmental component into the insertion algorithm to solve pickup and delivery problems with time window. A parallel regret insertion algorithm is proposed by Diana and Dessouky (2004) based on a case study at Los Angeles County. Lu and Dessouky (2006) present a new measurement criterion, the cost of reducing time window slack into classical insertion algorithm to solve a multi-vehicle pickup and delivery problem with time windows. Visual attractiveness is also quantified in the proposed insertion algorithm study, Ren et al (2010), a multi-shift VRP with overtime inspired by a healthcare delivery system. Tabu Search is embedded in the insertion algorithm framework for improvement. The empirical results show the improvement in terms of time and cost by the proposed solution.

To the best of our knowledge, no research has been done on MTVRSPTW-MB using the insertion heuristic. The most related paper, Campbell and Savelsbergh (2004), provides a brief description model of multiple trips per vehicle. This problem is different from our problem because they predefined the number of trips in a route and meal break is not

considered. Furthermore, the trip time limit in our model complicates the classical insertion implementation. Traditionally, the insertion heuristic schedules a set of customers in sequential order. In our problem, we need to break the tradition sequence into multiple trips with additional tasks in between, such as meal break and multiple travelling to the depot. Therefore, basic insertion heuristics will not necessarily produce a feasible solution in our case.

5.2.Problem Definition

We considered the MTVRSPTW-MB as defined in Chapter 2; however, synchronisation of loading teams is not included in this chapter. This constraint will be considered in Chapter 7.

In most of the delivery and pickup application, each customer required only one servicing team. Although the vehicle routing problem with split deliveries allows more than a vehicle to serve a customer, manpower synchronisation does not seem to be critical in most of the case studies. This might due to the holding cost and warehouse capacity at customers' site. Therefore, it is sensible not to include this constraint in this chapter.

5.2.1. Insertion Algorithm

In classical insertion approach, there is only one trip. Therefore, a decision has to be made on which new customer (aircraft, in our case) is to be inserted and where to insert this customer.

However, in our case, since there is more than one trip in a route and a meal break has to be inserted, additional factors need to be considered. The number of trips (a complete travelling cycle from service centre to apron) is not fixed which complicates the implementation further. An extra question the needs to be answered in such scenario is when to define a new trip? To overcome this, there seem to be two options: (a) insert all the customers in a trip, then pack the sequential customers into trips; (b) sequentially construct the trips by insertion. Both options can potentially provide the maximum savings by aggregating the customers in each trip, and achieve minimum trips required. However, option (a) could easily make the previous schedule become infeasible because customers' time windows would be not be met by adding the travelling tasks of newly defined trip at the packing stage. For option (b), we need to consider if the customer to be inserted into a partial trip or to form a new trip. This makes the evaluation process more complex and creates either

too few trips or too many trips in a route. Furthermore, when a customer is proposed to be inserted into any trip of the route, the feasibility check needs to go through every element/task from first trip to the last trip on the route. This is because an insertion will push the precedent trip to earlier time unit and forward the following trip to later time unit. By doing this, the computational effort will be greatly increased. On top of that, when the number of trips is not fixed, travelling tasks (between apron and service centre) can only be allocated along with the first aircraft insertion in a newly added trip. Therefore, potential insertion slots have become another important factor in our case study.

In addition, in our case, there are extra task requirements (the meal break and a few travelling tasks between apron and service centre) that need to be allocated; whereas, previous studies only schedule a sequence of customers. A meal break must be allocated between two consecutive travelling tasks, as they must be conducted at depot, within a given time window. Here, we need to consider when to insert the meal break appropriately. For example, if we only insert the meal break after constructing the trips, it will easily make the previous partial solution impractical due to time window constraints. On the other hand, it will have either too many or too few trips before the meal break if it is allocated before constructing trips for a route.

In this study, the Insertion Algorithm is proposed to construct a shift in three stages to cope with multiple travelling requirements. Initially, all aircraft are sorted in increasing order according to its arrival time, a_i, followed by their laxity, $b_i - p_i$. Then a new pair of decision variables, e_i and l_i, are defined as the earliest time to start service and the latest time to start service, respectively, to ensure the feasibility of solution.

The three stages of the insertion algorithm to form a route are defined as follows:

(a) *Define new shift*

A new shift is initiated by the aircraft on the top of the sorted list. A pair of travelling tasks is allocated together with the aircraft, travel to apron with truck and return to service centre as one complete trip. The first aircraft to be served, the shift time, the meal break, and the timeline for the first trip are defined accordingly. The shift time limit is set by using a dummy aircraft with zero maintenance processing time; meanwhile, the timeline for the first trip is set by the returning travelling task. A Time Window Table (TWT) is created to keep track of each task allocated associated with e_i and l_i. Then, the aircraft chosen in the TWT will be removed from the sorted list. Figure 5.1 shows the components to define a new shift and the implementation of it.

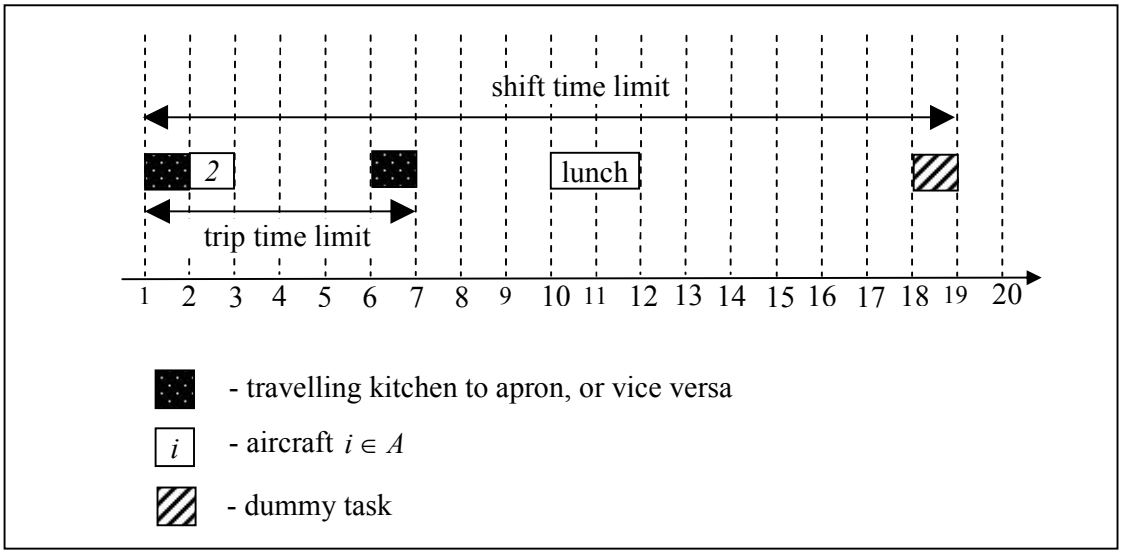


Figure 5.1. Components to define a new shift

(b) *Expand trip*

In this stage, the remaining aircraft in the sorted list will be evaluated for insertion into the newly defined trip. The potential insertion slots are restricted to slots after scheduled aircraft on the particular trip. In other words, the remaining aircrafts are not allowed to be inserted after any travelling tasks, meal breaks or dummy tasks. The best insertion slot, C_1 for each remaining aircraft, u, is selected by using the following equations:

$$C_1(u) = \min[\alpha c_1(u) + \beta c_2(u) + \gamma c_3(u)]; \quad (1)$$

$$\alpha + \beta + \gamma = 1; \quad (2)$$

$$0 \le \alpha \le 1,\ 0 \le \beta \le 1,\ 0 \le \gamma \le 1; \quad (3)$$

Equation (1) evaluates the maximum saving caused by the insertion, which includes distance reduction, c_1, shorter absolute waiting, c_2, and wider width of new time window, c_3. The time window cost, c_3, tends to offer the aircraft a wider time window due to the insertion. By adding this, it allows more aircraft insertions in future iterations. Parameters α, β, and γ are used to represent the weight of each "saving" criteria, based on the different objective functions or scenarios, as managerial parameters for managerial preferences. For example, the greater value of α on the objective function of minimising the total travelled distance, or the smaller value of β in the single travelling trip.

Next, the equation to evaluate the optimum aircraft, C_2 to be inserted is defined as follows:

$$C_2\big(i(u^*), u^*, j(u^*)\big) = \text{optimum}[\mu\, d_{0,u} + 2\theta\, p_0 - C_1(u)]; \tag{4}$$

$$\mu + \theta = 1; \tag{5}$$

$$\mu \geq 0,\ \theta \geq 0; \tag{6}$$

The first and second terms in (4) represent the distance and new pairs of travelling tasks required if the aircraft is served on a new trip. $d_{0,u}$ represents the travelling distance in time between depot to customer u ; meanwhile p_0 refers standard setup processing time at depot plus the travelling time returning to depot. The managerial parameter weight of each criterion is given as μ and θ, respectively. This stage will be repeated until no more feasible aircraft in the sorted list can be inserted into the particular trip.

(c) *Insert new trip*

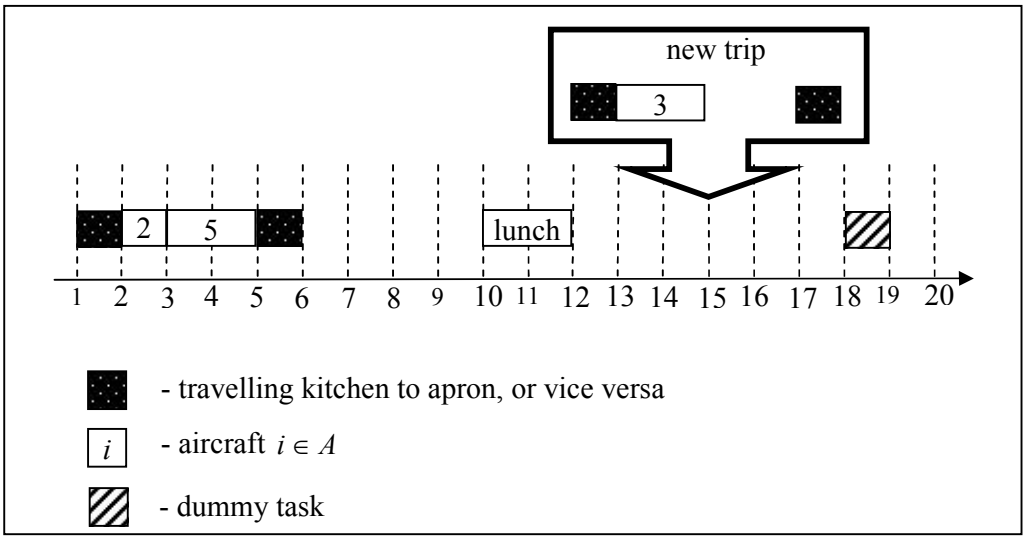


Figure 5.2. Components of new trip to be inserted

A travelling time is defined as the travelling activity between depot and the customer's site plus the standard setup time in the kitchen. The meal break, the dummy task and the travelling task are considered as non-aircraft tasks. A new trip is defined as a "block" of tasks including the best feasible aircraft and a pair of travelling tasks. The potential insertion slots for this new group of tasks are restricted to any slot between two non-aircraft tasks, as shown in Figure 5.2. An evaluation equation similar to Equation (1) is used to obtain

the most feasible aircraft to initiate a new trip. Then, go to stage (b). These stages will be repeated until no more feasible aircraft in the sorted list can be inserted into the route.

The e_i (earliest time to start service) and l_i (the latest time to start service) of each task in the route will be updated after each insertion, except l_i of the returning travelling task of the particular newly defined trip. This is due to its function which consists of setting the time limits of a trip. However, it will be updated at the end of stage (b) by $l_i = l_{i-1} + p_{i-1}$. The time window update of other tasks is as follows:

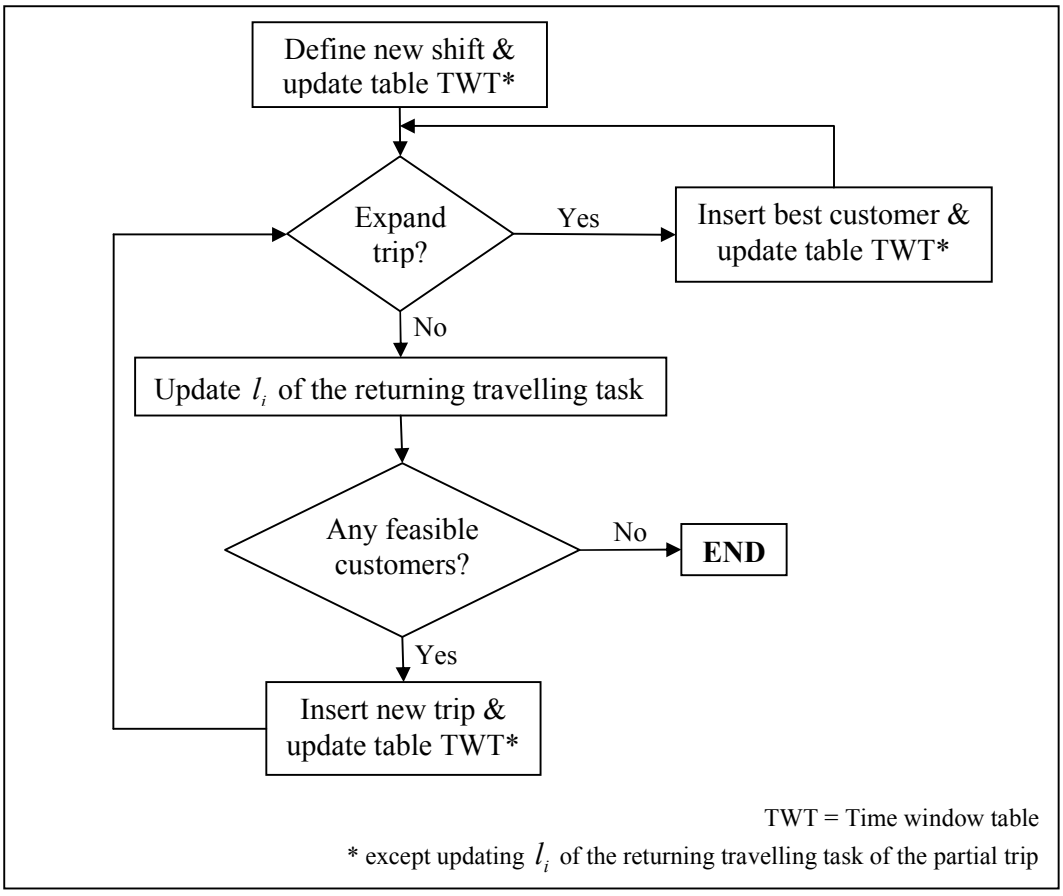
- Update the l values for all prior tasks from the new insertion backwards by,

$$l_i = \min(l_i, l_{i+1} - p_i) \tag{7}$$

- Update the e values for all subsequent tasks from new insertion forwards by,

$$e_i = \max(e_i, e_{i-1} + p_{i-1}) \tag{8}$$

The Insertion Algorithm is presented as flowchart below:



5.3. Performance Analysis

A five-day data set based on an in-flight caterer from Malaysia, was studied to evaluate the performance of the proposed solution. The data can be concluded as follows:

Day 1: 234 aircraft tasks

Day 2: 267 aircraft tasks

Day 3: 252 aircraft tasks

Day 4: 278 aircraft tasks

Day 5: 259 aircraft tasks

The heuristic was coded in C# and all experiments were carried out on a Dell Pentium 4 1.8GHz computer. All experimental tests can be computed in less than five seconds. Computational results with different parameter settings are given to demonstrate the robustness and efficiency of the proposed algorithm.

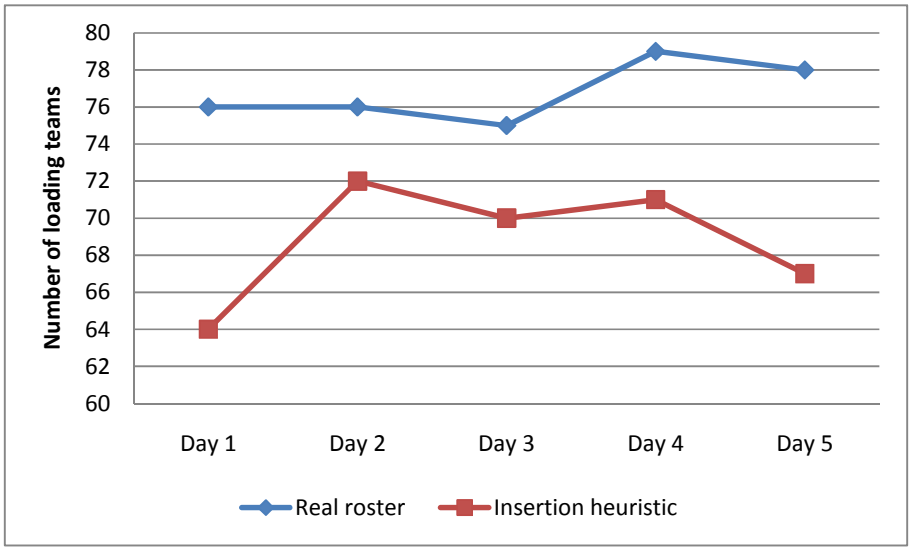


Figure 5.3: Comparison of the number of loading teams required between real roster and Insertion Algorithm solution

The in-flight caterer needs 75-79 loading teams each day within the five-day period with synchronisation of loading teams is applied. Under the same simulation setting,: an eight-hour working shift, a one-hour meal break, a two-hour trip time limit, two aircrafts per trip, and 15 minute kitchen-apron travelling times, Figure 5.3 shows the insertion heuristic is able to reduce the number of loading teams needed when the synchronisation of teams is relaxed. On average, the insertion heuristic reduced the amount of loading teams by eight each day, with a maximum of 12 loading teams on Day 1.

We ran a variation where meal break is not given. However, the working shift remains at eight hours. The number of loading teams required reduces further by 5.55% to 7.46%, as shown in Figure 5.4. This result makes sense as each loading team is given an extra hour to handle the services.

Figure 5.4: Meal break sensitivity analysis

We ran the same test data under different scenarios as follow:

(a) *Limit two travelling trips for each maintenance team*

Each maintenance team returns to the service centre to have a meal break after servicing one or more aircraft in a trip. After that, the teams go back to the apron for more maintenance work before they finish their shift. In this scenario, the meal break is given in the middle of the shift, between the third and sixth hours of working. Furthermore, the limited truck capacity and trip travelling time constraints are also relaxed.

(b) *Two travelling trips where no time restriction on meal break*

It is similar to scenario (a), but teams are given the flexibility to take a one-hour meal break anytime during their shift.

(c) *Single trip travelling*

In this scenario, the problem is reduced to the Vehicle Scheduling Problem with Time Window (VSPTW), where all aircraft must be inspected during their transit times. All

multiple travelling requirement constraints are relaxed, such as limited truck capacity, trip travelling time limits, and meal break allocation. However, the maximum working hours constraint is applied.

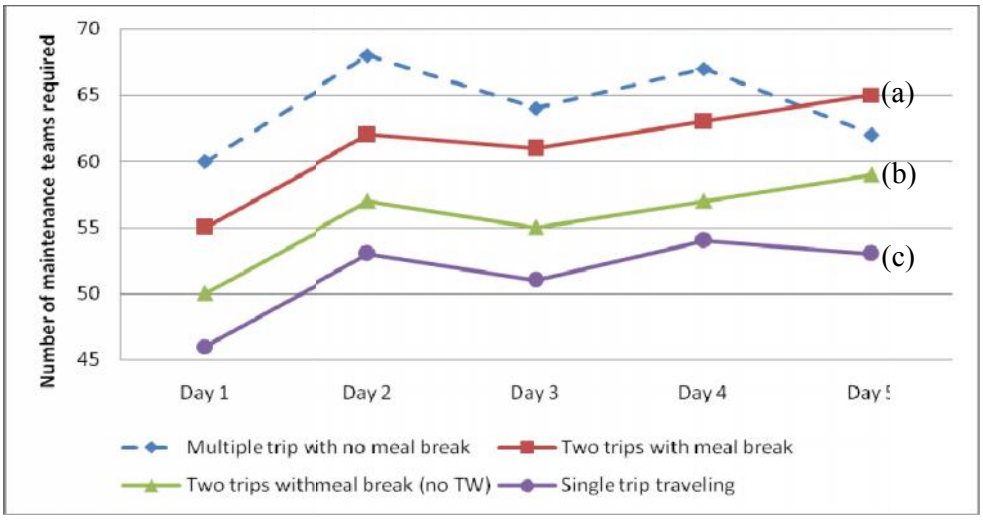


Figure 5.5: Sensivity analysis of Insertion algorithm

Figure 5.5 shows the sensitivity analysis result of the insertion algorithm for scenarios (a), (b), and (c). The dotted lines refer to the insertion solution of the original problem disregarding meal breaks. This sensitivity result demonstrates that the more operations constraints are relaxed, the fewer maintenance teams are required. This result is expected due to the fact that the travelled distance has been reduced accordingly. For example, when the truck limited capacity and the trip travelling time limits are relaxed, the travelling tasks between the apron and the service centre can also be reduced dramatically.

Overall, the manpower reduction increases from scenarios (a), (b), and (c) in a nearly fixed pattern, excluding Day 5 data. This is due to the unavoidable idle time caused by meal break allocation or its time window.

By comparing scenario (a) and (b), the manpower can be reduced by as much as 36.36% by relaxing the time window of meal break allocation. The analysis shows that the insertion algorithm can efficiently accommodate the meal break either at the beginning or at the end of the shift so that more maintenance tasks can be accomplished in a shift. This demonstrates the ability of the insertion algorithm to accommodate complicated operational constraints such as multiple trips travelling and meal break allocation.

More numerical analysis on the effectiveness and efficiency of insertion heuristic is conducted in Chapter 7.

5.4. Conclusion

In this chapter, the insertion heuristic is proposed to solve manpower scheduling for the in-flight catering delivery system, where loading team synchronisation is not considered. After aircraft tasks are sorted according to time window, the insertion heuristic is developed by constructing a trip by trip scenario due to multiple trip travelling requirements, while fulfilling the aircraft's time window.

Real data from an in-flight caterer in Malaysia was collected and tested on insertion heuristic. Computational results illustrates that the insertion heuristic can handle complex manpower scheduling, such as multiple trips travelling and meal break allocation efficiently and effectively. Furthermore, the insertion heuristic can generate large problem instances in seconds.

Chapter 6: A MTVRSPTW with Meal Break Consideration and Workforce Synchronisation

6.1. Introduction

Workforce synchronisation may not be critical in some industries; however, it is important in in-flight catering delivery application in Chapter 2. This is because manpower synchronisation is one of the operation requirements to ensure the aircraft maintenance service is conducted within their transit time, especially on wide body passenger aircraft with tight time window.

MTVRSPTW with meal break and teams synchronisation is discussed in this chapter. Workforce synchronisation has never been considered in any MTVRSPTW in the literature. It turns out that solving the MTVRSPTW with meal break and workforce synchronisation consideration by using insertion heuristic is a difficult task, described in Chapter 5. This is because workforce synchronisation is rigid with respect to the service start time. Furthermore, it requires parallel insertion. Therefore, it would complicate the insertion process further on top of number of trips in not predefined and extra tasks requirement. Thus, a Two-Stage Scheduling Heuristic (TSH) is proposed to solve the problem in this chapter.

6.2. Two-Stage Scheduling Heuristic (TSH)

In TSH, Stage I focuses on the vehicle's capacity and trip travelling time limit. It is a greedy algorithm which first tries to assign jobs to vehicles that is currently servicing in apron, followed by vehicles which are idle in the depot. The least priority is given to new vehicles that have not service any customers. A synchronisation procedure is then performed to assign and schedule the vehicle and workforce accordingly. Trips by particular vehicles are formed and each trip is defined as a succession of operations. Then, trips are used as input to Stage II as "block" jobs. Stage II accommodates service teams' unavailability for reasons such as meal break allocation and the maximum working hour limit.

First, all customers are sorted in non-decreasing order of their release times. The secondary sorting is according to their laxity $(d_i - p_i)$. In the case of a tie between two or more customers, we schedule first the customer with the smaller processing time. Each customer will be assigned to service teams based on the resulting customer list. Let U denote the sorted customers in this study.

6.2.1 Stage I

In this stage, we define a set of trips, *H,* to be served by the fleet of vehicles. A new variable, $w_i^{r,v}$, is defined as the service start time of customer $i \in U$ serviced by vehicle $v \in V$ on trip $r \in H$. Assuming the vehicles are able to work throughout the planning horizon, the main objective is to minimise the number of vehicles used.

A trip is defined as a complete travelling to and from the depot, and delivers and serves one or more customers during the travelling. The trips are in chronological order for each vehicle. Meanwhile, a route consists of one or more travelling trips to form a working shift for a worker. Thus, a vehicle serves trip *f* after trip *e* if, and only if, $e < f$.

A simple vehicle priority rule is used for assigning customers as follows:

(a) Customer site - Set *CS*

A vehicle located at a customer site and available to serve any unscheduled customers is given top priority. This is because of the possibility of accommodating more customers in a single trip.

Let *z* and *z'* be the last served customers on a partial trip and the partial trip on a given vehicle, respectively. After a feasibility evaluation, where the time window constraint and the vehicle limited capacity and travel time limitation are fulfilled, each unassigned customer will be assigned to any available vehicle, *k,* with the earliest end time of last customer served, i.e., $\min_k \left\{ w_z^{z',k} + p_z \right\}$. In case of a tie between two or more vehicles, we opt to use the vehicle with the earliest departure time from depot for the partial trip, i.e. $\min_k \left\{ w_{n+1}^{z',k} \right\}$. Following this approach, the idle time between two consecutive customers can be minimised.

(b) Depot – set *D*

Vehicles which returned to the depot after completing a trip will be considered for serving unscheduled customers if there is no feasible vehicle in group (a). A new trip is formed with the goal of serving the unscheduled customer by *k* feasible vehicles with the earliest end time of last returning travelling task, i.e. $\min_k \left\{ w_{n+2}^{s,k} \right\}$ on $s \in H$. This approach serves to minimise the idle time between two subsequent trips.

(c) New vehicle – set *NV*

A new vehicle which has not performed any previous trips is given the least priority. Initially, the fleet of vehicles is categorised in this group.

Let G_j^i be a set of feasible vehicles from $j \in \{CS, D\}$ to serve customer $i \in U$ we define the algorithm for Stage I below.

Stage I Algorithm:

For customer $i = 1, \cdots, n$, $i \in U$ do

Initiate feasible vehicle sets G_{CS}^i and G_D^i for customer *i* from *CS* and *D*

if ($G_{CS}^i \neq 0$)
Assign customer *i* to available vehicles with $\min_k \{w_z^{z',k} + p_z\}$, followed by $\min_k \{w_{n+1}^{z',k}\}$.

else if ($G_D^i \neq 0$)
Assign customer *i* to available vehicles with $\min_k \{w_{n+2}^{s,k}\}$.

else
Assign customer *i* to vehicle in set *NV*.
end if

Update $w_i^{r,k}$
end for-loop

A simple example of scheduling eight aircraft is illustrated in Figure 6.1 to demonstrate the procedures in Stage I. We have applied the same simulation setting as in the real roster: seven-unit trip time limit, two aircrafts per trip, and one-unit kitchen-apron travelling, in this example.

The aircraft are sorted according to their arrival times and laxity, as shown in Figure 6.1. Based on the sorted list, aircraft will be assigned to vehicles accordingly. Initially, all vehicles are in Set *NV*.

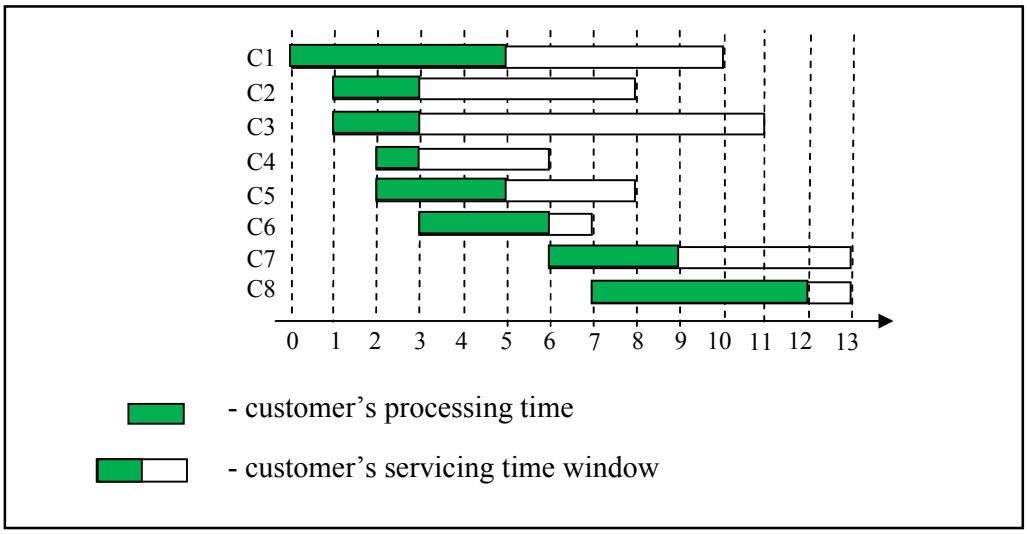


Figure 6.1. An example of eight aircrafts

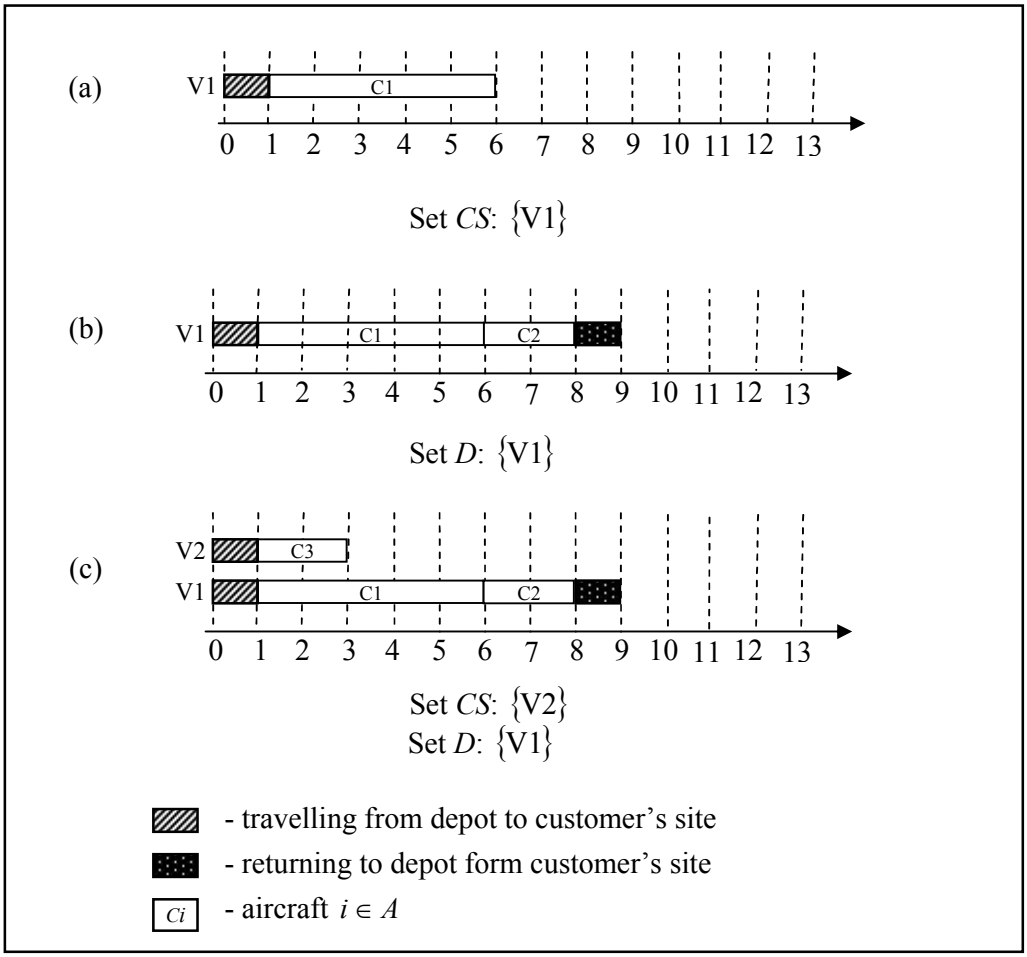


Figure 6.2. C1 assignment

C1 is first assigned to a vehicle based on the sorted list, as shown in Figure 6.2(a). Since set *CS* and set *D* are empty, C1 is assigned to a vehicle in set *NV*. The travelling task from depot to customer's site is also allocated accordingly for every newly defined trip. Then, C2 is assigned to V1 since it has top priority in vehicle groups as in Figure 6.2(b).VI has to return to depot after servicing C2 and categorised in set *D* because no more than two aircraft can be served in a trip. The travelling task of returning depot is allocated after C2.

As Figure 6.2(c), C3 is assigned to V2 from set *NV*, even though V1 has the higher priority than it. This is because assignment to V1 does not fulfil the time window constraint of C3.

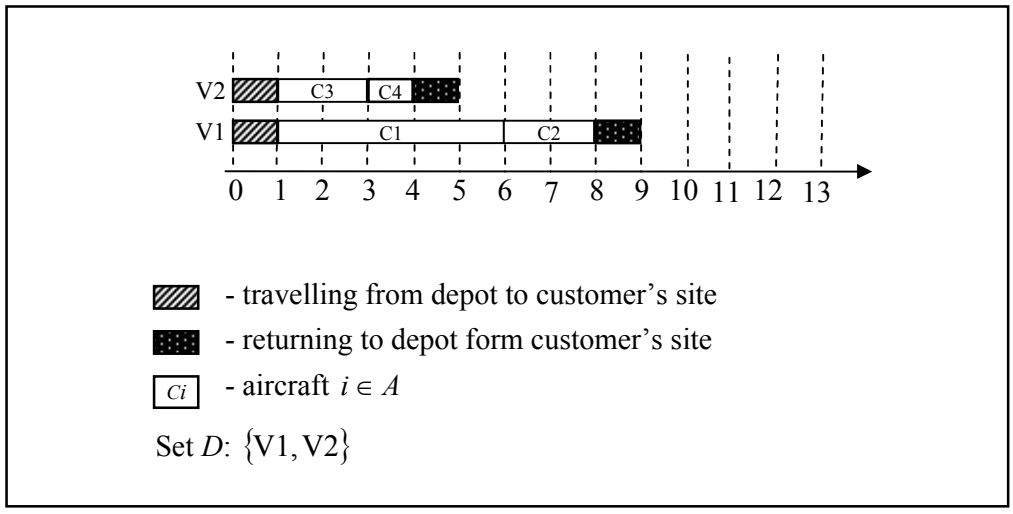


Figure 6.3. C4 assignment

In terms of assigning C4, it could feasibly be assigned to either V2 or a vehicle in set *NV*. Due to the vehicle priority rule, C4 is assigned to V2, which belongs to set *CS,* as shown in Figure 6.3. Again, travelling task is allocated at the end of each trip to return to depot.

A new vehicle is used to assign C5, as shown in Figure 6.4. There is no feasible vehicle in both set *D* and set *CS* to accommodate C5 in its tight time window. Similarly, a new vehicle is assigned for C6.

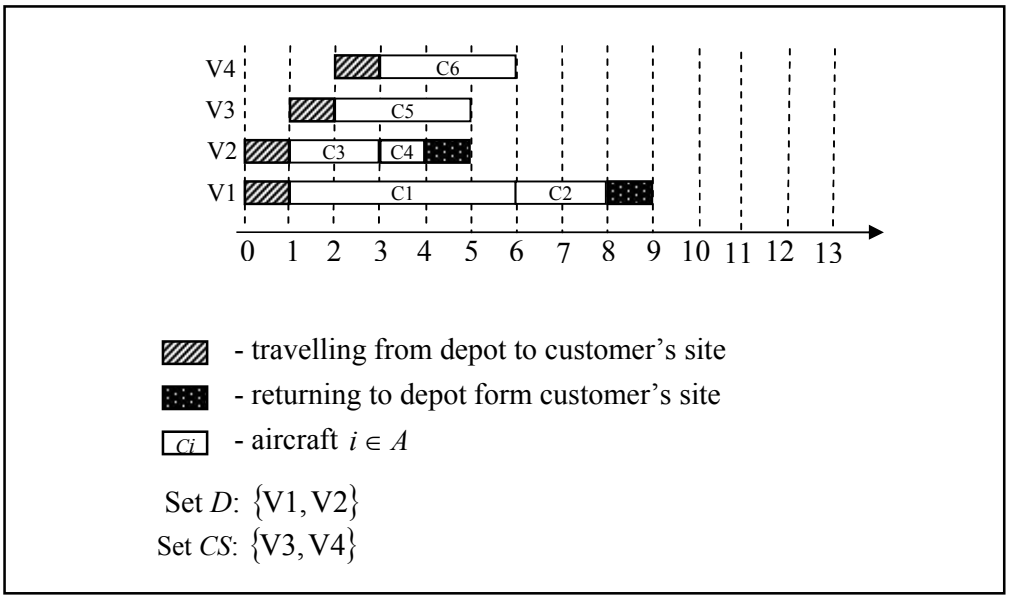


Figure 6.4. C6 assignment

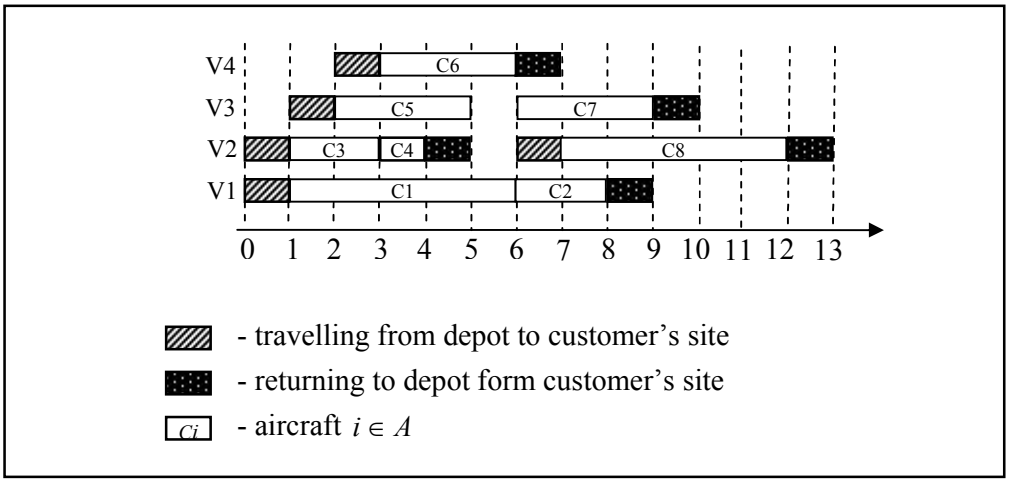


Figure 6.5. Stage 1 scheduling completed

C7 has two options of allocation, either at V3 or V4 as shown in Figure 6.4. Both fulfil the time window and vehicle capacity constraints. In this case of a tie between two vehicles, we opt to use the vehicle with the earliest departure time from depot for the partial trip, i.e. $\min_{k\in(V3,V4)}\left\{w_{n+1}^{z',k}\right\}$, which is V3. For C8, it could only be assign to V2 due to feasibility constraints. Finally, a complete Stage I scheduling for these eight aircraft is shown in Figure 6.5.

6.2.2. Synchronisation of Service Teams

When synchronisation of service teams is considered, a customer must be serviced by the different service teams during the same time interval. Despite the fact that this is a common operational constraint in many real-life systems, past studies have not considered it.

The procedure of teams' synchronisation is to generate a required number of vehicles based on the vehicle priority rule as Stage I. It might produce few different service start times. Then, the service start times will be adjusted to an identical service start time. For example, consider the case where a customer requires two loading teams. Two assigned vehicles with two start times will be generated by using vehicle priority rules. If the start times are different, the earlier start time will be push forward as the later one. However, we need to check the feasibility of the adjusted start time. If it is not feasible, the vehicle with late start time will be removed and reassign the next vehicle. This process will be repeated until the service start times are identical.

r_i is defined as the number of service teams required to serve customer *i*. Let a_i be the service start time for customer *i.* The algorithm used to synchronise the service teams is defined as follows:

For customer $i = 1, \cdots, n$, $i \in U$ do

Step 1 Use vehicle's priority rule in Stage I algorithm for r_i times to generate r_i vehicles without updating $w_i^{r,k}$ of customer *i* and vehicles in *CS* and *D*.

Step 2 Sort all the service start times of all a_i vehicles. Let *v* be vehicle with the latest service start time among all.

Step 3 Push forward service start time of other vehicles as *v*'s and check the feasibility conditions.

If a vehicle does not fulfil feasibility constraints, remove *v* and repeat Stage I algorithm once to generate a new truck, without updating $w_i^{r,k}$ of customer *i* and vehicles in *CS* and *D*.
Then, go to Step 2.

Or else, update each $w_i^{r,k}$ of customer *i* and vehicles in set *CS* and *D*.

end for-loop

6.2.3. Stage II

The main objective of Stage II is to minimise the number of service teams. The working shift times, together with a single break allocation, are assigned to each team.

Prior to this stage, each trip generated from Stage I is considered as a "block" of job, which is defined as a new set of jobs in Stage II. Each "block" job containing one service operation and several customer servicing tasks, is performed in succession. Without altering the service start time assigned in Stage I, we proceed by allocating these "block" jobs to the service teams.

At this stage, a greedy packing approach is used. We assign and schedule all the "block" jobs to service teams according to the earliest start time first policy. Ties are broken according to the smaller processing time first criterion.

In addition, we assign each service team a meal break. We treat the meal break as a distinct job (or as a "block" job) with its own time window. Service teams are assigned as many jobs as possible before l_{early} hours, the minimum number of working hours before a meal break is available. A meal break is given upon completion of a job where the job is completed within $\left[l_{early}, l_{late}\right]$. If a job is completed earlier than l_{early} but the next job completion time is more than l_{late}, then meal break is allocated exactly on l_{early} (creating an unavoidable idle period between the completion time of the last job before the break and the start time of the break). After assigning the meal break, service teams are again assigned as many jobs as possible until the maximum working hours are completed, T.

6.3. Performance Analysis

The TSH procedure was coded in C# and all experiments were carried out on a Dell Pentium IV 1.8GHz computer.

6.3.1. Case Study Analysis

Clearly, one would be interested in comparing the solution quality of TSH and current practice from the company. Again, we evaluated the performance of TSH for larger data sets with over 200 customers by using the same five-day data set as shown in section 4 in Chapter 5. In this case study, the aircraft play the role of customers in the MTVRSPTW-MB with loading teams' synchronisation consideration. The total number of aircraft and the total

number of split tasks when the synchronisation of loading teams is considered are illustrated in Table 6.1.

Table 6.1. Five-day data from case study

	Total aircrafts	No. tasks
Day 1	141	234
Day 2	151	267
Day 3	147	252
Day 4	158	278
Day 5	155	259

In Table 6.2, we provide the solutions obtained by TSH and compare them with the actual roster developed by the company's professional planner. The number of loading teams required by both solutions, the percentage improvement made by TSH, and the computing time (in seconds) of TSH are given. The manpower is reduced by 13.28% on average compared to the actual roster. In addition, the loading teams are synchronised to carry out the servicing operations.

TSH requires less than three seconds of computational time, where most of the computational time is mainly spent on scheduling customers at synchronisation procedures in Stage I. This demonstrates that TSH can solve the MTVRSPTW-MB with workforce synchronisation consideration efficiently and effectively for large data sets.

Table 6.2. Comparison result of TSH and real roster on the number of loading teams

	Real	TSH	Improve (%)	CPU (sec)
Day 1	76	60	21.05	3
Day 2	76	72	5.41	3
Day 3	75	61	18.67	3
Day 4	79	71	10.13	3
Day 5	78	69	11.54	3
Total	384	333	13.28	

Since the raw data from the company consists of varying types of time windows, we propose to pre-process the data with Time Window Reduction (TWR), which is described in

Chapter 4. From Table 6.3, it follows that the use of manpower can be reduced further by 7.81%, on average, when TWR is performed prior to TSH. In total, 26 teams, i.e. 208 man-hours can be saved. The best solution of TWR with TSH is chosen among different values of $W = \{1,2,3,5,10\}$. This result makes sense as the time window reduction is based on the idea of spreading the customers as evenly as possible throughout the planning horizon.

Table 6.3. Computational result when TWR is applied

	TSH	TWR + TSH	Improve (%)	CPU (sec)
Day 1	60	54	10	3
Day 2	72	65	9.72	3
Day 3	61	59	3.28	3
Day 4	71	67	5.63	3
Day 5	69	62	10.14	3
Total	333	307	7.81	

The computational time of the whole process remains less than three seconds. Clearly, the execution time of TWR is negligible compared to that of TSH.

6.3.2. Synchronisation of Loading Teams Relaxed

When synchronisation of loading teams is not considered in the model, different loading teams are allowed to serve the particular aircraft at different time intervals. In some cases, a loading team might serve the same aircraft more than once, if the aircraft has a sufficiently wide time window and $r_i > 1$. This flexibility leads to a further reduction in the number of teams required, as demonstrated in Table 6.4.

Table 6.4. Computational result of TSH when synchronisation of loading teams is relaxed

	TSH	TSH- Team	Reduce (%)	CPU (sec)
Day 1	60	58	3.33	3
Day 2	72	64	11.11	3
Day 3	61	56	8.20	3
Day 4	71	68	4.23	3
Day 5	69	63	8.70	3

Total	333	309	7.21	

#TSH-Team = TSH where synchronisation of loading teams is relaxed

On average, there is a 7.21% reduction in the number of teams required when loading teams' synchronisation is not required compare with the case with synchronisation. These results clearly suggest that TSH can easily be adapted to different requirements and operational constraints.

Table 6.5 provides computational results when TWR is performed prior to TSH for the case where synchronisation of loading teams is relaxed. When TWR is applied, the manpower can be reduced approximately 2.92% in total. The reduction is relatively small compared to the previous experiment. This is due to the flexibility that loading teams possess in serving aircraft anytime within their respective time window.

Table 6.5. Computational result when TWR is applied

	TSH-team	TWR + TSH-team	Reduce (%)	CPU (sec)
Day 1	58	55	5.17	3
Day 2	64	62	3.13	3
Day 3	56	56	0	3
Day 4	68	65	4.41	3
Day 5	63	62	1.59	3
Total	309	300	2.92	

6.3.3. Meal Break Sensitivity

In order to examine the impact of the meal break together with loading teams' synchronisation, we ran the same test data with TSH by disregarding the meal break allocation. The sensitivity results are summarized in Table 6.6.

Table 6.6. Meal break sensitivity analysis

	TSH	TSH-Lunch	Reduce (%)	CPU (sec)
Day 1	60	53	11.67	3
Day 2	72	66	8.33	3
Day 3	61	56	8.20	3
Day 4	71	68	4.23	3

Day 5	69	61	11.59	3
Total	333	304	8.71	

TSH-Lunch = TSH where meal break constraint is relaxed

The increase in the number of teams is due to the fact that the mandatory meal break is considered as an additional job. Therefore, the shift times are now reduced by the break length. Furthermore, due to the time window constraint on the break, it is more likely that unavoidable idle time before and after the break is created. Overall, the results are very satisfactory, since that increase in the number of teams is almost equivalent to the increase in the number of tasks' working hours divided with shift working hours.

6.3.4. Analysis of the Relaxed-MTVRSTW

Since we are the first to study the problem, there is no exact, or even heuristic solution for the problem studied in literature, we could not provide a performance gap with respect to the optimal solution. However, in order to get a sense of how the TSH performs, besides comparing it with the real life data given in previous section, we compare with the solution given by CPLEX for the two trips VRPTW with meal break consideration, as in Chapter 3.

Table 6.7 below provides the number of teams and the computational time for different instances. The CPLEX solution and the TSH heuristic are compared for the case where W is set to 5. The computational (CPU) time of TSH is less than one second for all instances considered.

Table 6.7. Comparison of performance by CPLEX and TSH

No.	TSH	CPLEX	
10	4 teams	2 teams	6 sec
15	5 teams	3 teams	2,608 sec
20	6 teams	6 teams	57,615 sec
25	7 teams	7 teams	100,845 sec
30	9 teams	9 teams	86,400 sec

Since our objective is to develop a heuristic that can efficiently and quickly construct a quality solution, we are not attempting to outperform the CPLEX solution. First, the CPLEX algorithm (branch and bound in Column Generation framework) is significantly

more complex than our approach. Even though TSH did not perform well for 10 and 15 customers, it matches CPLEX for larger problem instances. Furthermore, CPLEX proves to be very sensitive to the size of the data set and requires exponentially increasing computation time as the number of tasks increases (over 15 hours are required to solve instances of 15 to 20 customers), whereas TSH is computationally bounded.

6.4. Conclusion

In this chapter, we developed an original, Two-Stage Scheduling Heuristic (TSH) for solving the MTVRSPTW with meal break and workforce synchronisation consideration. The proposed heuristic is a computationally bounded heuristic and is shown to generate very high quality solutions in a short time. Furthermore, TSH is easily adaptable to different operational environments and can accommodate even the most complicated logistical constraints, such as loading team synchronisation and multiple vehicle trips.

TSH was compared to the actual roster used by a commercial airline operator. The computational results verify that TSH is a superior tool in solving MTVRSTW-MB, both in terms of solution quality and computing time, even when workforce synchronisation is applied. Besides, a comparison between TSH and solutions obtained by CPLEX was undertaken for the two trips VRPTW with meal break consideration. The time window reduction algorithm which is used to pre-process the tasks is shown to further improve the performance of the TSH procedure as it allows smoothing the demand for tasks over the working day.