SOLVING A REAL MANPOWER ROSTERING PROBLEM

ACADEMIC EXERCISE 06/07
BY: YAP JUN WEI U034534W

NUS BUSINESS SCHOOL
NATIONAL UNIVERSITY OF SINGAPORE
Acknowledgement

Firstly, the accomplishment of this academic exercise will be virtually impossible without the guidance of my supervisor, A/P Teo Chung Piaw, and I am indeed fortunate to have the chance to work with him on my AE. He has taught me meticulously in the past months, showing me different approaches of analyzing problems and opening up my mind to new ideas. Once again, I will like to extend my heartfelt thanks and gratitude to him.

In addition, I will also like to thank the people involved in the manpower rostering project and they include Dr. Nilotpal, Dr. Liu Qi Zhang, Mr. Manish and his team. I have benefited much from your wealth of experience and understanding of the problem. I thanked Dr. Nilotpal for allowing me on this great project and Dr. Liu for taking time off to share his expertise in Tabu Search and programming. Mr. Manish has also been patience in letting me understand the nature of the manpower rostering problem during my internship stint.

I will also like to thank all the other professors who have imparted their knowledge and have given me valuable advices in the past 3 years. I will mention a few here: A/P Albert Teo who has taken good care of me as my USP academic advisor, Dr. Mabel Chou who has broaden my perspectives in the field of Decision Science research, A/P Hui Tak Kee and A/P Sum Chee Chuong who have been excellent and approachable teachers.

I am also very grateful to my family who has supported me all these years. Rain or shine, they have given me all the support they can rally in whatever I do. My mum and dad have provided me with a conducive environment to nurture and realize my greatest potential. I will also take this chance to thank my sister, Wendy, for the graphic design in this paper. In addition, I also thank my girlfriend, Shumei, for walking through this AE journey with me and sharing my sweet-and-sour for the past year.

Last but not least, I thank all my friends in NUS who has made this 3 ½ years’ journey in NUS a truly memorable one.
Abstract
This study is targeted at a real-life manpower rostering problem based on an UK automobile company. The UK automobile company provides on-the-road assistance in 100 over regions in UK and it faces the problem of rostering its servicing staff while managing complex constraints. A consultancy company was then tasked in May 2006 to produce a more efficient and automated way to do the roster. A 3-stage heuristic was designed by the consultancy company. The first stage comprises of an optimization program to fix the workday assignment and the second stage encompasses a construction heuristic to fix the shift assignments within the days. The last stage is a search algorithm to seek a better solution. I joined the project in June and was tasked with the coding of the construction heuristic (stage 2) to construct a good initial feasible solution. This paper is divided into 2 parts. The first part will provide the detailed problem setup and give you an introduction to the 3-stage heuristic. I will also go through the details of the construction heuristic which I have coded. In all, based on the 9 test regions, the results of the 3-stage heuristic mark a 48% improvement over the current roster. In the second part, I will touch upon a mathematical programming approach to further improve the construction heuristic. The derived mathematical model is then translated into an optimization programming language to be solved in ILOG CPLEX. The results are encouraging and the CPLEX model manages to make further improvements to the original construction heuristic.
PART 1 THE BEGINNING

1 INTRODUCTION & PROBLEM DEFINITION .......................................................................................... 6
  1.1 PROBLEM SET-UP .......................................................................................................................... 7
  1.2 CONSTRAINTS .................................................................................................................................. 8
  1.3 WORK DAY ASSIGNMENT ............................................................................................................ 9
  1.4 SHIFT ASSIGNMENT .................................................................................................................... 11
2 CURRENT ROSTERING PROCEDURE .................................................................................................. 17
  2.1 MANUAL ROSTERING ..................................................................................................................... 18
3 THE ALTERNATIVE ALGORITHM ......................................................................................................... 21
  3.1 STAGE 1 COT/SWAP OPTIMIZATION & DATA PROCESSING .......................................................... 23
  3.2 STAGE 2 CONSTRUCTION HEURISTIC .......................................................................................... 25
    3.2.1 Initial Construction Heuristic .................................................................................................... 27
    3.2.2 Revised Construction Heuristic (Oversupplied Region) ............................................................ 30
    3.2.3 Revised Construction Heuristic (Undersupplied Region) .......................................................... 33
    3.2.4 Start Time ................................................................................................................................... 34
  3.3 STAGE 3 TABU SEARCH ............................................................................................................... 36
4 RESULTS ................................................................................................................................................. 38

PART 2 THE CPLEX MODEL

1 LITERATURE REVIEW .......................................................................................................................... 44
  1.1 DAYS-OFF SCHEDULING ............................................................................................................... 44
  1.2 SHIFT SCHEDULING ...................................................................................................................... 45
  1.3 TOUR SCHEDULING & FLEXIBILITY ............................................................................................ 46
  1.4 START-TIME BAND & ANNUALIZED HOURS .............................................................................. 47
2 THE MODEL .......................................................................................................................................... 49
  2.1 BASIC FORMULATION ................................................................................................................... 49
  2.2 MULTI-DAY MODEL & ANNUALIZED HOURS ............................................................................ 52
  2.3 DIVIDE & CONQUER ...................................................................................................................... 54
  2.4 THE FULL MODEL ......................................................................................................................... 57
3 RESULTS ............................................................................................................................................... 63
4 CONCLUSION ......................................................................................................................................... 67
  4.1 CONTRIBUTIONS ............................................................................................................................ 67
  4.2 FUTURE OUTLOOK ......................................................................................................................... 68
  4.3 FINAL WORDS .................................................................................................................................. 68
APPENDIX A – CONSTRUCTION HEURISTIC VB CODE ........................................................................... 73
APPENDIX B – CONSTRUCTION HEURISTIC REPORT .......................................................................... 76
APPENDIX C – OPL CODE ...................................................................................................................... 81
PART 1

SOLVING A REAL MANPOWER ROSTERING PROBLEM

THE BEGINNING
1 Introduction & Problem Definition
This part will orientate the readers on the scope of the manpower rostering problem by providing a walkthrough of my internship stint with a consultancy company, which is tasked by an automobile company in UK to produce an automated and efficient way to plan their staffs’ working schedule (The two companies will not be named in this paper). The UK automobile company provides a variety of motoring-related products for its members and business customers and one of their core competencies lies in providing first class on-the-road assistance to motorists. With over 15 million members, 3500 patrols and 100 service regions in UK (Servicing staffs that provide on-the-road assistance, are termed as “patrols”), the automobile company takes pride in their premium services and makes continual efforts to uphold the quality standards. The patrols respond to vehicle breakdowns or other emergency events on the road and do the repairs on-site. Equipped with state-of-the-art technology and wealth of experience from their well trained patrols, they even promise to provide the members with a replacement car if the patrols are unable to fix the vehicle on-site. Each patrol is required to have a couple of years of experience in motor repair or engineering related industry before joining the company. Understanding that their reputation lies in the hands of their patrols, the company also emphasizes on continual training and development. To attract and retain these patrols, the company offers a highly competitive remuneration package and a range of staff benefits. In addition, as these patrols are scheduled to work on shifts (irregular working hours), the patrols are allowed to state their preferences for their shift work. Giovanni et al. (2004) state, “The pattern of the non work days (days off) in the working period is highly related to staff satisfaction”. In addition, the company implemented
certain rules to allow the patrols to cope the shift work with minimum inconvenience, such as ensuring that the shift’s starting times in a row of continuous working days are within certain predetermined hours (these constraints will be elaborated in the later sections). In a recent paper “Reinventing Crew Scheduling at Netherlands Railway”, Abbink et al. (2006) wrote about the development of an alternative set of scheduling rules called Sharing-Sweet-and-Sour at the Dutch railway operator NS Reizigers. The motivation behind this paper arises from an incident in 2001 when 6500+ drivers and conductors were dissatisfied with their duties schedules and structure and this led to massive nationwide strikes. The alternative scheduling rules eventually satisfied the staffs’ requests and greatly improved the efficiency and punctuality of the railway operator. This demonstrated the importance of having a good duties’ structure in the implementation of shift work and it is extremely crucial that the shifts are well designed to ensure high satisfaction of the patrols. These inducements will secure lower turnover rates, reduce the need to re-train new patrols and indirectly ensure consistency in service standards. However, we see that such flexibility and welfare in roster planning given to their patrols pose a strain on the operations of the company and compromise the company’s capability to meet their service standards. In the rest of this section, we will try to understand the problem faced by the company in planning their patrols’ roster.

1.1 Problem Set-up
The patrols’ rosters are planned over a 6-month period according to the manpower forecast. An example of the manpower forecast is shown in Table 1.
Table 1

The manpower forecast is made for the 6-month period and manpower requirement forecast is made hourly for each of the 183 days in all the 100 over regions. In Table 1, we can see that the forecasted manpower requirement for day 1 (0000h to 0100h) is 1 and the predicted manpower for day 183 (2300h-0000h) is 3. The objective is simple. A roster should be constructed to match the manpower deployment to the forecasted requirement.

We define an undersupply to be the situation where the manpower deployed in the hour is lower than the manpower required and vice versa, oversupply occurs when the manpower deployed is higher than the manpower requirement. A key performance indicator of a roster would be the total undersupply hours over the planning horizon and our goal is to minimize the sum of undersupply hours across the 183 days. Mathematically, we define $d$ as the day index, $h$ as the hour index, $D_{d,h}$ as the forecasted demand in hour $h$ of day $d$ and $M_{d,h}$ as the manpower deployed in hour $h$ of day $d$. The objective function thus becomes:

$$\text{Minimize } \sum_{d=1}^{183} \sum_{h=1}^{24} (D_{d,h} - M_{d,h})^+$$

1.2 Constraints

The problem can be easily solved by treating each day as a separate optimization problem but by giving their patrols the flexibility to choose their working days and introducing
rules to make shift work more bearable, the solution space becomes very restricted and feasibility of a solution becomes an issue of concern. We will now understand the constraints facing us in this manpower rostering problem. There are 8 main contract types (PBA 1, PDA 1 etc.) and each can be specifically tailored to suit individual patrols in terms of number of over time hours, number of stand-by hours etc. However, we will explain these contracts in 2 general categories: Post-2005 Agreement and Pre-2005 Agreement. In 2005, there was an option for the existing patrols to switch to the Post-2005 agreement and the new patrols that joined after 2005 will be automatically routed into the new scheme. We will see that there are substantial differences in these 2 types of agreements later. The constraints are broadly separated into 2 classes: workday assignment and shift assignment. The terminologies used in the rest of this paper are adopted from the UK automobile company.

1.3 Work Day Assignment
Based on individual contract types, the patrols are permitted to choose a working-day pattern such as 6W (Work)-3R (Rest)-6W-3R or 7W-3R-7W-2R. In addition, the patrols are allowed to plan their leaves, training days and protected days (Rest days that cannot be converted to a working day) for the 6-month period. The patrols will thus provide a tentative roster of workdays’ assignment. Table 2 shows an example of a patrol’s workday assignment.

<table>
<thead>
<tr>
<th>DAY</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
<th>14</th>
<th>15</th>
<th>16</th>
<th>17</th>
<th>18</th>
</tr>
</thead>
<tbody>
<tr>
<td>P1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>P</td>
<td>L</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>T</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Table 2
The patrol above has chosen a 6W-3R-6W-3R workday pattern and 1 indicates a working day while 0 represents a resting day. P, L and T indicate a protected day, a leave and a
training day respectively. The patrol is shown working for the first 6 days and it is followed by 3 rest days. The rest day on day 9 is converted to a protected rest day while he chooses to take leave on day 10 and his training day falls on day 15. Leave and training days will fall on workdays while protected days are for rest days. The rationale for protected days is to prevent the planner from making the patrol work overtime or come back for standby duties on that day. The patrol may be planning a weekend trip on day 9 and 10 and does not hope to be interrupted. This is because this workday assignment above is only tentative and the planner has the flexibility to change some of the work and rest days to meet the manpower requirement subjected to certain constraints. The 2 main types of changes to this tentative assignment are contractual overtime (COT) and swaps. A COT can change a rest day to a workday and swaps are done between a workday and a rest day. We see that COT can potentially ease manpower shortages and swaps can reduce undersupply situation on one day and prevent oversupply situation on the other.

Of course, not all swaps and COT are feasible and while COT applies to Pre and Post-2005 agreements, swaps are only permissible for post-2005 agreements. In general, when we convert a rest day to a workday, we need to adhere to the following constraints:

- This day is not a bank holiday or protected day.
- This day should be a weekday.
- This day is on the edge of a rest block.
- There should not be three or more consecutive leaves in continuation of this day.
The first 2 constraints are easily understood and I will illustrate the 3rd and the 4th in Table 3.

<table>
<thead>
<tr>
<th>DAY</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>P1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>P2</td>
<td>1</td>
<td>0</td>
<td>L</td>
<td>L</td>
<td>L</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

Table 3
We define a row of continuous rest days as a rest block. For patrol 1, day 3 to 5 represents a rest block and since day 3 and 5 are on the edge of a rest block, they are feasible days for COT or swaps. However, for P2, even though day 2 is on the edge of the rest block, since there are 3 leaves following day 2, a swap or COT is forbade. For swap, we need to identify feasible working days to be converted to rest days and the key criterion is that these workdays must be on the weekends. In short, we can only swap a working weekend with a resting weekday. COT and swaps are also limited to 1 and 2 per month respectively. These set of rules will ensure minimum disruption to the patrols and as a result of these restrictions, the number of feasible swaps is likely to be only 100 – 200 per month in a region.

1.4 Shift Assignment
I will now highlight the major constraints that affect the shift assignment within the day (note that only the important constraints are covered). For each region, there is a set of capping hours. Capping hours refer to the range of hours where we may not roster our patrols. For instance, the capping hours may be from 0000h to 0600h. Therefore, we will not roster any patrols in this time period even if there is manpower needed in these hours. However, the manpower requirement in these hours tends to be very small and it is assumed that these works will be outsourced.
The duration of the shift is restricted by the maximum and minimum shift length and these parameters vary with contract types and work day patterns. Typically, the minimum shift length is between 5 to 6 hours while the maximum shift length is around 10 hours.

The next few constraints relate to a work block (a row of consecutive working days). Depending on workday patterns and contract types, there is maximum number of hours a patrol can work in a work block. For example, a 6W-3R work pattern usually stipulates 55 hours of maximum working hours in a block. In addition, the start times of the shifts in the work block have to be contained within a certain variance. For pre-2005 agreement, the start time variance is typically 1 and 2 hours on weekday and weekend respectively. For post-2005 agreement, the start time variance is typically 2 and 3 hours on weekday and weekend respectively. This constraint probably makes it easier for the patrols to adapt to their shift work as the start time in a work block are about the same and they are able to get appropriate rest between the shifts. I shall now further elaborate on this constraint in table 4 which shows the start times for a patrol that has a work block from Monday to Sunday.

<table>
<thead>
<tr>
<th>MON</th>
<th>TUES</th>
<th>WED</th>
<th>THURS</th>
<th>FRI</th>
<th>SAT</th>
<th>SUN</th>
</tr>
</thead>
<tbody>
<tr>
<td>0700H</td>
<td>0800H</td>
<td>0900H</td>
<td>0800H</td>
<td>0700H</td>
<td>0600H</td>
<td>0800H</td>
</tr>
</tbody>
</table>

Table 4

For weekday start time variance, we will look at only the weekdays’ start time in the block. However, for weekends’ start time, we will look at all the days (inclusive of weekdays) to determine the difference. The weekend’s variance is usually larger than the weekday’s. In the above example, the patrol has weekday and weekend start time range
of 2 and 3 hours. The above assignment is a feasible one as the weekday start times range from 0700h to 0900h while the entire block’s start times adhere to the weekend start time variance of 3 hours (0600h – 0900h). We see that the start time on Sunday cannot be 1000h given that the start time on Saturday is 0600h. Similarly, the start time on Wednesday cannot be 1000h given that Monday and Friday start time is 0700h. Therefore, the shift assignment for days within a work block is closely related and this adds complexity to the rostering process.

On a macro scale, all patrols will have to fulfill 1883 hours in a year and since the roster are planned on a 6-month period, the hours are prorated according to the relative manpower requirement in the 2 halves of the year. These 1883 hours are inclusive of leaves, COT hours and training hours and the relative proportion depends on contract type. 1883 hours less the leaves and training hours represent the total available supply of manpower that has to be distributed over the entire planning horizon. This is a big constraint as it “links” up all the workdays in the planning horizon. The optimization process has to decide how to distribute the available supply to meet the given objective.

Another constraint is the presence of withheld hours and these are hours where we may not roster a patrol to work. They are set aside for recreation activities or training events for the patrols. If the withheld hours fall on the patrol’s workday, his shift must either end before the withheld hours or start after the withheld hours slot. The withheld hours’ constraint indirectly restricts the start times in the block too. I will illustrate this in table 5.
Table 5

The patrol above has a withheld hours’ slot from 1200h to 1700h on day 2 in a 4-day work block. First we look at day 2, assuming a minimum shift length of 5 hours and capping hours from 2300h to 0700h (meaning that the earliest start time is 0700h while all shifts will have to end by 2300h), we see that there is only 2 feasible start times 0700h and 1700h (denoted by ☼). The rest of the workdays within the same block will be restricted by the withheld hours as the start time will need to be within (say) 2 hours variance from 0700h or 1500h (denoted by ☼).

The above constraints apply to both pre and post-2005 agreements but there are constraints applicable only for the pre-2005 agreement. Although these constraints will be phased out with time, they are still effective for about 10% of the patrols. These patrols are given a choice of start-by preferences or finish-by preferences. Their shifts will either start with or end latest by their preferred time. The shifts are generally classified as “earlies” or “lates”. Typically, an “earlies” shift will start before the time of 1200h and “lates” shift will start from 1200h. In addition, they can state their preferences
for all “earlies,” “lates” or even alternating “earlies” and “lates” (across different work blocks) in their shift rosters. Table 6 shows an example of a pre-2005 patrol work shift preferences;

<table>
<thead>
<tr>
<th></th>
<th>WEEKDAY EARLIES</th>
<th>WEEKDAY LATES</th>
<th>WEEKEND EARLIES</th>
<th>WEEKEND LATES</th>
<th>ALT.</th>
<th>PREF</th>
</tr>
</thead>
<tbody>
<tr>
<td>P1</td>
<td>0700 0800 2300</td>
<td>1200 1400 2300</td>
<td>0700 0900 2300</td>
<td>1200 1400 2300</td>
<td>1</td>
<td>N</td>
</tr>
</tbody>
</table>

Table 6

The above patrol has a start-by preference and we can see his preference for start times for weekday and weekend “earlies” and “lates” shift. For example, he should start work between 0700h to 0900h on a weekend if he is on early and 1200h to 1400h on weekend if he is on “lates” shift. A “1” in the ALT. column indicates that he wants an alternating shift pattern (i.e. if work block 1 is on “earlies,” work block 2 should be on “lates” etc.) and an N in the PREF column indicates that he has no preference for “earlies” or “lates.”

Note that if he has preference for all “earlies” or “lates” shift (by indicating an E or a L), the ALT. column will contain a 0.

All the constraints are hard constraints and must be strictly adhered to. However, there are situations where there are conflicting constraints. For example, a patrol with start-by preference of 800h to 900h and a preference for all “earlies” shift, has a withheld hour slot from 1000h to 1500h. In this case, if the earliest start time is at 0600h (because of capping hours) and the minimum shift length is 5 hours, the “earlies” shift that starts at 0800h will violate the withheld hour slot (since the earliest end time is 1300h). Therefore not all constraints will be satisfied and some prioritization of constraints is used to decide the allocation of shifts. In this case, the start by time takes precedence. The shift will start at 0800h and end at 1300h. The withheld hour will be shortened or delayed.
I have now briefly covered the main constraints relating to the problem and one can imagine the complexity in deriving a feasible and optimal solution to this manpower rostering problems. In table 7, I have summarized the constraints and categorized them into pre and post 2005 agreements.

<table>
<thead>
<tr>
<th>CONSTRAINTS</th>
<th>PRE-2005 PATROLS</th>
<th>POST-2005 PATROLS</th>
</tr>
</thead>
<tbody>
<tr>
<td>CAPPING REGION</td>
<td>APPLIES TO ALL</td>
<td></td>
</tr>
<tr>
<td>MAX/MIN SHIFT LENGTH</td>
<td>APPLIES TO ALL</td>
<td></td>
</tr>
<tr>
<td>MAXIMUM BLOCK HOURS</td>
<td>APPLIES TO ALL</td>
<td></td>
</tr>
<tr>
<td>START TIME VARIANCE</td>
<td>1 / 2 HOURS</td>
<td>2 / 3 HOURS</td>
</tr>
<tr>
<td>WITHHELD HOURS</td>
<td>APPLIES TO ALL</td>
<td></td>
</tr>
<tr>
<td>START BY / FINISH BY</td>
<td>APPLICABLE</td>
<td>N.A.</td>
</tr>
<tr>
<td>ALL EARLIES/LATES/ALT</td>
<td>APPLICABLE</td>
<td>N.A.</td>
</tr>
<tr>
<td>TOTAL HOUR</td>
<td>APPLIES TO ALL</td>
<td></td>
</tr>
</tbody>
</table>

Table 7

It is observed that the post 2005 agreement granted more flexibility to the company to plan the roster but as we do not have direct contacts with the UK company, it is unknown what is the tradeoff in this increase in flexibility (It can be a reduction of the total base hours or other benefits granted to the patrols). In the next section, we will take a look at the current rostering procedures and methods that the company uses.
2 Current Rostering Procedure

Figure 1
The rostering procedure starts with the tentative workday roster (where the patrols choose their workday pattern, plan their leaves, training and protected days) and the 6-month manpower forecast. With this tentative roster, the company can now allocate shifts to the patrols working on each day or conduct swaps and COT subjected to the various constraints. Figure 1 shows the current rostering process. First, several of these starting feasible solutions will be generated. The idea is to change a starting initial solution to a better solution while maintaining feasibility. Instead of conducting local search, a simulated annealing approach is used. In local search, we ask if there is a better solution in the neighbouring points and if it exists, we move to that point and we continue to assess the neighbourhood points. The neighbourhood of the current solutions may include swapping of working day and a rest day for a patrol, changing the shift length for a particular day or changing the starting time for the entire block of working days etc. This iterated approach seeks to find a local optima solution but it is unusual that the local optima coincides with the global optima. Simulated annealing tries to overcome this problem by entailing a probabilistic approach to escape from the local optima. An initial feasible solution is used and its neighbourhood is assessed. A worse-off solution may be
accepted with a probability that changes as the algorithm proceeds. With more starting initial solutions, several runs of the search may be conducted to obtain a near optimal result.

The UK company has been using this method to plan their patrols’ roster and in May this year, they engaged a Singapore based consultancy company to help find better ways to plan the roster. The project also involves a team of engineers from India and I was very fortunate to have a chance to work with them on this project as well.

2.1 Manual Rostering
The initial plan was to assess if there are any rooms for improvement to the current solution obtained via the simulated annealing process. The India’s engineers thus derived a manual rostering method and this method literally means doing COT and swaps and allocating shifts manually (one by one). The first phase is to make amendments to the tentative workday assignment before doing the actual shift assignment in the second phase. Figure 2 shows the manual rostering procedures.

A simple intuition was used by the engineers to make amendments to the tentative workday assignment. First, they work out the average workload of the patrols working on
each day. For example, on day 1, the total manpower requirement (obtained by summing up all the manpower requirement of the non capping hours) is 84 and the total number of patrol working on this day is 10. The average workload of each patrol would be about 8.4 hours (84 / 10). Similarly, the average workload of all the other 182 days can be calculated. The objective is to “even out” the average workload of the patrols across the planning horizon. Table 8 shows a sample of the calculated average workload of some days.

<table>
<thead>
<tr>
<th>DAY</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td>AVE</td>
<td>8.4</td>
<td>7.55</td>
<td>8.12</td>
<td>8.43</td>
<td>7.45</td>
<td>7.9</td>
<td>6.7</td>
<td>9.3</td>
<td>7.2</td>
</tr>
</tbody>
</table>

Table 7
Suppose day 7 has the lowest average work load and day 8 has the highest among the 183 days, we will try to add a COT day or do swaps to correct the manpower imbalance. Of course, sometimes it cannot be done due to certain constraints. For example, day 8 may be a bank holiday and it is not possible to assign more patrols to work on that day. By visually inspecting each day and make manual adjustments, we continue until we can no longer make any corrections to the manpower imbalances. In the second phase, with the number of patrols working on each day fixed, we then make the shift assignments while adhering to all constraints. This manual rostering process would be almost impossible without the aid of macro programming in excel to guide the planner. For example, COT and swap feasibility will be automatically checked when the planner assign a COT or swap and the feasible start time will be highlighted in yellow for shift assignment. Figures such as hourly manpower requirement are also automatically updated as the shift allocation takes place. With these additional aids, a region can be rostered in approximately 2-3 days (I consider this very impressive!). The apparent
downside of this manual process is that it is sensitive to changes in the input information such as the available base hours or changes in the constraints. Therefore, as the project proceeds thru the months, new constraints’ update invalidates the manual roster’s solution and makes it infeasible. However, despite minor constraint violations, the manual roster provided our project with a good benchmark to develop our alternative algorithm as the manual roster results has surpassed the current simulated annealing results. Most importantly, it has been shown that there are still plenty of rooms for improvement in the current solutions.
3 The Alternative Algorithm
With the success of the manual process, the project team went on to think of ideas to efficiently do the roster. With an average of 30 patrols in over 100 regions, the process should be automated and works reasonably well for all regions. Gendron (2005) managed to solve weekly scheduling problems for the 3000 staffs at Quebec liquor store. Although he is faced with complex lunch break constraints, the planning horizon is only a week and the problem decomposes by employees. The unique scheduling problem at Quebec liquor stores requires the planner to award the best shift to the most senior employees (union regulation). Therefore, it sufficed to consider the employees sequentially in the optimization process. However, in another instance, Dowsland et al. (2000) used heuristic and integer programming model to solve a nurse rostering problem at an UK hospital. The complicating constraints did not allow either of the heuristic or the integer programming alone to produce a satisfactory solution. From these examples, we see that formulation of a full mathematical model and solving it via optimization software depends very much on the nature and characteristics of the problem.

| DAYS | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | … |
|------|---|---|---|---|---|---|---|---|---|----|----|----|----|----|----|----|----|----|…|
| P1   |   |   |   |   |   |   |   |   |   |    |    |    |    |    |    |    |    |    |…|
| P2   |   |   |   |   |   |   |   |   |   |    |    |    |    |    |    |    |    |    |…|
| P3   |   |   |   |   |   |   |   |   |   |    |    |    |    |    |    |    |    |    |…|
| P4   |   |   |   |   |   |   |   |   |   |    |    |    |    |    |    |    |    |    |…|

Table 8
Table 8 shows a sample workday assignment for 4 patrols of a particular region and the highlighted boxes represent the working days. Majority of the constraints relate to work block constraints and there is no way to decompose the model by time frame (the full mathematical model will require us to solve the entire 6-month period). In addition, the
constraints such as the maximum block hours and the starting time variance closely relate the days together and changing any decision variables (such as start time) would affect the feasibility of the other day’s assignment. Changing the start time of a patrol on the 183rd day may affect the optimality of day 1 assignment! With 30 patrols, 183 days and a whole list of constraints and decision variables, the possibilities of formulating an optimization model for our problem were thus ruled out in the beginning due to the problem’s extensive scale. The project team thus refined the idea of the manual rostering process and incorporated the concept of using search strategies to find an optimal solution. A 3-stage process was devised and figure 3 illustrates this process.

![Diagram](image)

**Figure 3**

The key difference between the alternative algorithm and the current rostering process is that the alternative algorithm strives to construct a “good” feasible initial solution. In the current rostering process, simulated annealing is done on some random starting point and neglects the quality of the starting point.

The first stage is the COT/Swap optimization to make adjustment to the working days and rest days and is undertaken by the team of engineers from India. The aim is to optimize the swaps and the adding of COT days subjected to the various constraints. The
India office also pre-processes certain data to ensure consistency in the input files for the latter phases. I was assigned the work of devising a construction heuristic to obtain a good starting initial solution by assigning appropriate shift times to the patrols based on the finalized workday assignment (output from the COT/Swap optimization). The third stage involves the use of Tabu Search. It is a refinement of local search and similar to simulated annealing, it tries to escape local optima to seek better solution. It will take in the solution from stage 2 (construction heuristic) as the starting feasible solution. In the later sections, we will take a look at each phase of the work and understand the intuition behind the algorithm.

3.1 Stage 1 COT/Swap Optimization & Data Processing

The India office was charged with the responsibilities of data processing and producing the ILOG CPLEX model for COT/Swap optimization. The data that are to be processed includes the patrols’ contract type and their corresponding constraints. These data are vital to all stages of the roster process and this data processing was crucial to ensure consistency in the input data to the construction heuristic phase and the tabu search phase. Misunderstanding of the complex constraints often led to different input to the two phases. After the data processing, latter phases do not need to understand the contract types and the corresponding constraints. Rather, the patrol ID and its associated constraints will be listed in an input file. Constant updating of the input file was done throughout the entire project to reflect new constraints or parameters introduced.

The intuition behind the COT/Swap optimization model is very similar to the manual process where the planner attempts to correct manpower imbalances. Each day’s target
manpower is determined by summing the total manpower requirement and dividing it by the average shift length in the region (pre-assigned parameter). With this daily target manpower requirement, we can identify the shortage or surplus with the current workday assignment. Minimizing the sum of total shortages and surplus is not sufficient, as we do not want to see days with severe shortages or surplus. Therefore, a penalty term is introduced to punish days with shortages or surplus above a limit (arbitrary determined). Instead of visually inspecting the average workload of a patrol on each day and manually adjusting the manpower level, the COT/Swap model now determine the optimal manpower required for each day and tries to match that requirement. In addition, it tries to prevent severe shortages or surplus on each day.

However, we can see that the “optimal” manpower target is only an approximation and this approximation pays no attention to the demand distribution patterns. Different demand distribution may warrant different level of optimal manpower level. Figure 4 shows 2 demand distributions with the same total manpower requirement of 48 hours from 0700h to 2300h.
The demand distribution on the left shows a constant manpower requirement of 3 from 700h to 2300h (16 time slot). Assuming an average shift length of 8 hours, it is easy to see that the optimal manpower required is 6 (scheduling 3 from 700h to 1500h and the other 3 from 1500h to 2300h). This is exactly the same as the approximation (48/8 = 6). However, we see that the demand distribution on the right has a peak of 12 from 700h to 1000h and 1 from 1000h to 2300h. This day may need more than 12 patrols to effectively cover the entire manpower requirement (though this may not be the optimal when we consider all the days’ requirement). The point is that the approximation may convey the wrong information and the target may bear no resemblance to the true value. Failure to correctly identify days with manpower shortages and surplus can seriously undermine the effectiveness of the model but the approximation method is probably an improvement at the current stage.

3.2 Stage 2 Construction Heuristic

Given the optimized workday assignment, I will proceed to construct a feasible initial solution. I joined the project in late June and before I embarked on this project, I was told by the project coordinator to be prepared for “brutal programming”. It was indeed a mammoth task to code the construction heuristic and the first 2 weeks of my work was really about understanding the long list of constraints and determining the data structure to store these data for the code. The code was eventually done in visual basic in Excel (the partial program code is attached in Appendix A).
The construction heuristic works by choosing a permutation of days (e.g. calendar order) and within the days fix a permutation of patrols and assign them sequentially based on the objective function given to the program. Table 9 shows an example of how the roster is constructed for day 1 in the heuristic.

Table 9
The day’s manpower requirement is shown in the “Initial Dd” row. For patrol 1 and 2, all the start time are feasible (since this is the first day and start time constraint is not in effect) and they have a maximum and minimum shift length of (say) 5 and 10 hours respectively. With the feasible start times highlighted and the objective function of minimizing total undersupply and oversupply, we assign the first patrol to start at 0700h and ends at 1700h (1 in the “Patrol 1” row indicates that patrol 1 is working at the particular hour). The manpower requirement is now updated and shown in row “Updated Dd 1”. Patrol 2 shift will start at 0900h (since the demand for 700h and 800h is 0) and end at 1900h. The remaining patrols are assigned in this same way. After this day is done, we move on to the day 2.

Table 10
Table 10 shows the assignment of patrol 1 and 2 on day 2 and we see that the start time constraint has come into effect (we assume a start time variance of 2 hours for these 2 patrols). Ideally, with the objective function of minimizing total undersupply and oversupply, patrol 1 should start at 1000h. However, the latest feasible start time is 0900h and he starts at that time even if his shift will produce an oversupply (-1) at time 0900h. Similarly, patrol 2 tries to find the best start time and shift length.

As we proceed through the days, the program notes the earliest and latest start time in the block, the start time preference, the total number of hours used in the block, the withheld hour slot etc. This is to make sure that none of the constraints is violated. The extensive list of constraints made the debugging hard and getting the program to produce a feasible solution took me almost 2 weeks!

3.2.1 Initial Construction Heuristic
I was charged with the responsibilities of testing a basic 3-pass heuristic derived from the manual rostering process. Recall that there is a total hour constraint that the patrols cannot use more hours than their total available base hours in the entire rostering horizon. If the patrols use up too much hours in the early part of the heuristic (allocating too much shifts with maximum shift length), they are then forced to work shift with minimum shift length and this constitutes a big loss in flexibility over the next part of the heuristic. Juggling this constraint has been the toughest part of the construction heuristic. In the manual process, the planner first reduces the maximum shift allowed for each patrol and gives an initial assignment to all the days and patrols. Thus, the patrols will never run out
of hours at the end of the first round. Then, he will re-look at the shift assignment and uses the remaining hours to cover as much undersupply as possible.

This is the basic idea of the 3-pass heuristic. In pass 1, a lower parameter is assigned for the maximum shift length (10 hours may be reduced to 7 hours). Our objective function is to minimize the sum of undersupply and oversupply. We then proceed to allocate the shifts. In pass 2, we revisit each patrol’s assignment and try to extend their shifts to cover as much undersupply as possible. If there are still remaining hours left, we proceed to pass 3 to finish the allocation (note that the hours allocated in pass 3 will no longer cover any undersupply hours). Table 11 shows an example of this 3-pass heuristic.

<table>
<thead>
<tr>
<th>DEMAND</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
<th>14</th>
</tr>
</thead>
<tbody>
<tr>
<td>PATROL 1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>PATROL 2</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>PATROL 3</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>PATROL 4</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>PATROL 5</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>PATROL 6</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>PATROL 7</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>PATROL 8</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>PATROL 9</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>PATROL 10</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

| 0 | 0 | 0 | 0 | 1 | 2 | 0 | 0 | 0 | -1 | -1 | 0 | 0 | 0 |

Table 11

With maximum shift length set at 7 hours, the hours allocated in pass 1 are highlighted in blue. Hours added to cover additional undersupply in pass 2 are highlighted in light blue. The yellow cells highlight the undersupply (+) and the oversupply (-). The total undersupply hours are 3 and oversupply hours are 2 hours. Variants of the heuristic can be produced by altering the permutation of the patrols and days (like placing days with the largest undersupply in the front) and by trying out different variants, some good
solutions may be produced. However, I found out that the start time is sub-optimal if we use the 3-pass heuristic. I will illustrate this using table 12;

<table>
<thead>
<tr>
<th>Time</th>
<th>0700h</th>
<th>0800h</th>
<th>0900h</th>
<th>1000h</th>
<th>1100h</th>
<th>1200h</th>
<th>1300h</th>
<th>1400h</th>
<th>1500h</th>
<th>1600h</th>
</tr>
</thead>
<tbody>
<tr>
<td>Situation</td>
<td>U</td>
<td>U</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>U</td>
<td>U</td>
<td>U</td>
<td>U</td>
<td>U</td>
</tr>
</tbody>
</table>

Table 12

Assuming we are faced with a decision of allocating a patrols’ shift (Maximum / minimum length is 10 and 5 respectively) and “U” represent an undersupply situation while “O” represent an oversupply situation. The objective function used is to minimize the sum of total undersupply and since the maximum shift is 7 hours, the best choice is to start at 1200h and end at 1700h. However, we have left undersupply at 0700h and 0800h uncovered and even if we have sufficient hours to cover more undersupply hours in pass 2, we would not be able to do so as changing the shift start time would affect the feasibility of the shift assignment of other days. Technically speaking, it is possible to add COT hours to the front of the shift (start time remains unchanged) but the coding would be very complex as the COT hours need to be closely tracked. Therefore, if we have sufficient hours to cover all the undersupply hours, we should not lower the maximum shift length, as it would affect the optimality of the start time.

In addition, as the construction heuristic assigns shifts in a “greedy” manner without looking ahead, there will be mistakes in the assignment process. We see such an example in table 13.

<table>
<thead>
<tr>
<th>Time</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
<th>14</th>
<th>15</th>
<th>16</th>
<th>17</th>
<th>18</th>
<th>19</th>
<th>20</th>
<th>21</th>
<th>22</th>
</tr>
</thead>
<tbody>
<tr>
<td>Demand</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Table 13
Suppose that we have 2 more patrols left to assign to this day, patrol 1 can start at anytime while patrol 2 can only start at 0700h (due to start time constraints). If we assign patrol 1 first, we have 2 start time 0700h and 1800h. If the program knows that there is another patrol that can only start at 0700h, patrol 1 will choose to start at 1800h. However, the construction heuristic does not capture this information and arbitrary assign patrol 1. Therefore, we ought to incorporate some algorithm to correct the greedy assignment.

3.2.2 Revised Construction Heuristic (Oversupplied Region)

The problems encountered in the 3-pass heuristic motivated me to think of alternative method to do the rosters. Since there are oversupplied (total hours>>demand) and undersupplied (demand>>total hours) regions and we can probably employ different strategies to apply the construction heuristic. (The detailed report submitted to the consultancy on the revised construction heuristic is attached in Appendix B)

Firstly, the choice of objective function will be different for the 2 types of region. In a region, all the available hours will have to be utilized and a working hour either improves the undersupply situation or adds on to the oversupply hours. The problem is thus very unique as minimizing total undersupply, minimizing total oversupply and minimizing the sum of total undersupply and oversupply are the same. Mathematically;

\[
\sum_{d=1}^{183} \sum_{h=1}^{24} (D_{d,h} - M_{d,h})^+ \Leftrightarrow \sum_{d=1}^{183} \sum_{h=1}^{24} (M_{d,h} - D_{d,h})^+ \Leftrightarrow \sum_{d=1}^{183} \sum_{h=1}^{24} |D_{d,h} - M_{d,h}|
\]

Globally speaking, these three objective functions are essentially the same but the choice of the objective function will make a significant impact on the performance of our
proposed optimization algorithm. We show how different objective functions can alter the start time in table 14.

<table>
<thead>
<tr>
<th>Time</th>
<th>0700h</th>
<th>0800h</th>
<th>0900h</th>
<th>1000h</th>
<th>1100h</th>
<th>1200h</th>
<th>1300h</th>
<th>1400h</th>
<th>1500h</th>
<th>1600h</th>
</tr>
</thead>
<tbody>
<tr>
<td>Situation</td>
<td>U</td>
<td>U</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>U</td>
<td>U</td>
<td>U</td>
<td>U</td>
<td>U</td>
</tr>
</tbody>
</table>

Table 14

Suppose a patrol is deciding his shift start time and length, there are 2 choices and the selection will be dependent on the objective function. If the objective function places more weight on undersupply hours, the shift starts at 0700h and ends at 1700h. Alternatively, if the objective function contains equal weight on undersupply and oversupply, the shift from 0700h to 1700h will cover 7 undersupply hours and 3 oversupply hours and yields -4 in the objective value (-7+3=-4). However, we can start at 1200h instead and cover 5 undersupply hours (- 5). Based on this objective function, the latter will be selected (since -5<-4), leaving undersupply hours at 0700h and 0800h uncovered. In an oversupplied region, we should probably place more weight on undersupply region and try to cover as much undersupply as possible. In addition, the shift length should not be lowered, as it is unlikely we will use up all the hours. Basically, the construction heuristic for oversupplied regions, goes through 2 phases, pre-assignment and finalization phase.

In the pre-assignment phase, under the assumption of 1-2 peaks in the daily demand, a systematic way of assigning the patrols was devised and the intuition is illustrated in table 15.
Table 15
Since there are generally 2 peaks, there will be 3 types of shifts that start at the same time range. The first one will start from the beginning to the first peak while the second one will cover the 2 peaks and the third one will cover the second peak to the end. In the pre-assignment phase, we thus try to cover the beginning and the end of the day first before moving on to the middle section of the day.

Recall that the greedy approach will inevitably make mistake in the initial assignments. Therefore to “reverse” some costly mistake in the earlier assignment in the day, a finalization phase is introduced. Right after each day’s shift assignment is decided, we allow each patrol to re-determine their shift start time and length again while keeping the rest unchanged. The patrols are therefore “reshuffled” and some shifts may be reduced to reduce the oversupply hours or increased to cover more undersupply.

Table 16
We use the same example as table 11 and used the new heuristic to do the assignment. The boxes highlighted in blue represent the hours allocated in the pre-assignment phase and the hours added in the finalization phase are highlighted in light blue. We had 3 undersupply hours and 2 oversupply hours in the previous case and using the new heuristic, we only have 3 oversupply hours. The elimination of undersupply hours in an oversupplied region is crucial as there are sufficient hours to cover all the demand. Most importantly, this heuristic captures the optimal start time and allows the shifts to cover most undersupply possible.

3.2.3 Revised Construction Heuristic (Undersupplied Region)
For undersupplied regions, we do not have sufficient hours to cover all the demand and the construction heuristic is similar to the basic 3-pass heuristic. The objective function puts more weight on oversupply as we do not want to use up too much hours and the maximum shift length is adjusted to 7 hours to prevent over using of base hours. The finalization phase is also added and is similar to the previous heuristic where the patrols are “reshuffled” but the maximum shift length is still capped at 7 hours.

After this first pass, we proceed to the second pass but how do we allocate the remaining hours to cover as much undersupply as possible? As we have restricted the maximum shift length, we have many leftover hours and plenty of undersupply to cover, and the key is to assign them efficiently to cover as many undersupply hours without incurring excessive oversupply hours. In this pass, we process the patrols sequentially and try to extend their shifts. An intuitive way is to look at the ratio of undersupply hours to oversupply hours covered when extending the patrol’s shift. We will extend shifts with
higher ratios first. We will continue until no more hours can be added to cover undersupply or when the unallocated hours are used up. I illustrate with 2 examples. In the two scenarios (table 17 and 18), a patrol’s shift in pass 1 is from 0800h to 1300h (5 hours) and can be extended to 1800h (since maximum shift length is 10 hours).

<table>
<thead>
<tr>
<th>Time</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
<th>14</th>
<th>15</th>
<th>16</th>
<th>17</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>U</td>
<td>U</td>
<td>U</td>
<td>U</td>
<td>U</td>
<td>O</td>
</tr>
</tbody>
</table>

Table 17
In the above case 3 oversupply hours are needed to cover 2 undersupply hours and the ratio given will be 2/3.

<table>
<thead>
<tr>
<th>Time</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
<th>14</th>
<th>15</th>
<th>16</th>
<th>17</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>U</td>
<td>U</td>
<td>U</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
</tr>
</tbody>
</table>

Table 18
In this case, we can extend the shift to cover till 1500h without incurring any oversupply hours and we give a ratio a 10 to indicate a higher priority. If all the possible extensions are oversupply hours, a 0 is given. We will start extending the shifts with higher ratio till either all ratios are 0 or the hours have been used up. As this heuristic is very similar to the initial 3-pass heuristic, it suffers the same problem of having sub-optimal start time.

3.2.4 Start Time

<table>
<thead>
<tr>
<th>MON</th>
<th>TUES</th>
<th>WED</th>
<th>THURS</th>
<th>FRI</th>
</tr>
</thead>
<tbody>
<tr>
<td>1500H</td>
<td>13-1700H</td>
<td>15-1700H</td>
<td>15-1700H</td>
<td>15-1700H</td>
</tr>
</tbody>
</table>

Table 19
Table 19 shows the start time of a patrol that has a work block from Monday to Friday. We note that if he starts at 1500h on Monday, his Tuesday start time is restricted to 1300h and 1700h. Once Tuesday’s start time is chosen (say 1700h), the rest of the start times are fixed within the narrow range of 1500h and 1700h. We see a drastic reduction
in the flexibility of the rostering process. Such scenarios become worst when the starting time variance is only 1 hour. This constraint is usually tight and can severely impair the performance of the rostering algorithm. We will find that we are often caught in situation where we cannot find a patrol to start at an ideal time we desire and this poses many problems to the implementation of the construction heuristic. To reduce the interference of the start time constraints to the algorithm, it may be a good idea to restrict the starting time to an even smaller range than the allowed feasible starting times.

<table>
<thead>
<tr>
<th>Time</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
<th>14</th>
<th>15</th>
<th>16</th>
<th>17</th>
<th>18</th>
<th>19</th>
<th>20</th>
<th>21</th>
<th>22</th>
</tr>
</thead>
<tbody>
<tr>
<td>Feasible Start Time</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Cannot Start</td>
</tr>
</tbody>
</table>

Table 20 shows the range of starting time in a region where the capping hours is from 2300h to 0700h and the earliest start time is thus 0700h and the latest start time is 1800h (for patrols with minimum shift of 5 hours). Suppose a patrol has found the optimal start time to be at 1800h. This will have an adverse effect on the later stage of the heuristic as by starting at 1800h, the next day’s start time will be restricted to 1600h to 1800h (2 hours’ variance). Suppose even if the optimal start time is at 1800h, the patrol starts at 1500h, he just have to increase his shift length to cover the same undersupply hours (some hours will be wasted) but the next day’s start time range is increased from 1300h to 1700h. Therefore, we may impose an additional constraint that no patrol may start later than 1500h (although they are legally allowed to do so). The benefit is that the algorithm will not be much affected by the start time variance constraint. Although we may waste a few hours, we will ensure that in future moves, we will be less likely to come across a situation whereby we are unable to find a patrol to start at a time we need. Different values of the latest allowed start times can be tested to optimize algorithm performance.
The apparent trade off is that the smaller start time range will result in lesser interference from the start time variance constraint but more hours are used up in the rostering process.

### 3.3 Stage 3 Tabu Search

With the COT/Swap optimization and the construction heuristic, we hope that we have arrived at a good initial feasible solution and we will now employ a search algorithm to further improve our result. In Tabu search, a worst off solution may be selected and similar to simulated annealing, we hope to escape from a local optima. The algorithm focuses on searching the neighbourhood of the initial solution to find a better solution. The Tabu search uses the initial solution generated by the construction heuristic as the starting point. The neighbourhood of a solution is defined by the local moves and local moves involve changing the shift length of the patrol in a day, changing the starting time of a shift or changing the block starting time (by shifting the entire block in the assignment). However, we are not allowed to re-visit some points (tabu list) which we have moved to in the previous \( X \) moves (\( X \) is an arbitrary number). This is to prevent recycling of moves and through recycling, we may go back to the local optima. Of course there are cases where recycling of moves is allowed. By the use of additional parameters such as aspiration cost and tabu tenure, we can obtain some tradeoff between escaping from the local optima and finding a better solution thru recycling of moves. A reconstruction heuristic is also devised to generate another good solution from the current starting point and the purpose of this additional starting point is to enlarge the search space to increase the chances of finding a better solution. The current improvement may
not be much but if we increase the number of admissible moves, we may be able to improve the solution further (but it is not done due to time constraints).
4 Results
In this section, we will witness the alternate algorithm performance in doing the roster but first, let’s view the current results (using simulated annealing) in Table 21.

<table>
<thead>
<tr>
<th>Number</th>
<th>Name</th>
<th>Demand</th>
<th>Available Hours</th>
<th>Under-Supplied Hours</th>
<th>Over-Supplied Hours</th>
<th>Maximum Efficiency (%)</th>
<th>Efficiency (%)</th>
<th>Gap (Max Efficiency - Efficiency %)</th>
</tr>
</thead>
<tbody>
<tr>
<td>AN09B</td>
<td>Maidstone West</td>
<td>12,009</td>
<td>11,895</td>
<td>1,421</td>
<td>1,307</td>
<td>99.1%</td>
<td>88.2%</td>
<td>10.9%</td>
</tr>
<tr>
<td>AN12B</td>
<td>Guildford</td>
<td>17,122</td>
<td>11,589</td>
<td>5,674</td>
<td>140</td>
<td>67.7%</td>
<td>66.9%</td>
<td>0.8%</td>
</tr>
<tr>
<td>AN31B</td>
<td>Oxford</td>
<td>13,547</td>
<td>13,051</td>
<td>1,509</td>
<td>1,012</td>
<td>96.3%</td>
<td>88.9%</td>
<td>7.5%</td>
</tr>
<tr>
<td>AN32A</td>
<td>Reading</td>
<td>16,381</td>
<td>13,690</td>
<td>3,306</td>
<td>615</td>
<td>83.6%</td>
<td>79.8%</td>
<td>3.8%</td>
</tr>
<tr>
<td>AN45B</td>
<td>Skipton</td>
<td>11,287</td>
<td>8,770</td>
<td>2,857</td>
<td>340</td>
<td>77.7%</td>
<td>74.7%</td>
<td>3.0%</td>
</tr>
<tr>
<td>AN50</td>
<td>Wirral</td>
<td>18,605</td>
<td>19,984</td>
<td>1,140</td>
<td>2,518</td>
<td>100.0%</td>
<td>93.9%</td>
<td>6.1%</td>
</tr>
<tr>
<td>AN55</td>
<td>Northern Ireland</td>
<td>13,683</td>
<td>16,893</td>
<td>510</td>
<td>3,720</td>
<td>100.0%</td>
<td>96.3%</td>
<td>3.7%</td>
</tr>
<tr>
<td>AN62B</td>
<td>Edinburgh East</td>
<td>14,883</td>
<td>15,114</td>
<td>1,332</td>
<td>1,562</td>
<td>100.0%</td>
<td>91.1%</td>
<td>8.9%</td>
</tr>
<tr>
<td>AN64</td>
<td>Aberdeen</td>
<td>11,240</td>
<td>11,256</td>
<td>1,156</td>
<td>1,171</td>
<td>100.0%</td>
<td>89.7%</td>
<td>10.3%</td>
</tr>
</tbody>
</table>

Table 21
The above 9 regions are the test regions designated by the UK company and we were to conduct pilot runs of the alternative algorithm. The table above shows the results of the roster done using the current roster procedures (simulated annealing). The key performance indicator used is efficiency gap (requested by the UK company). The first 2 columns state the area code and the corresponding location. Column 3 is the total demand in the region for the entire roster period while column 4 is the total available hours (net of leaves and training days). The undersupply and the oversupply hours are listed in the following 2 columns. Maximum efficiency is a measure of the maximum number of demand that can be covered using the base hours. For instance, AN09B demand is 12009 hours and since total base hours are 11895 hours, the maximum efficiency is thus \( \frac{11895}{12009} = 99.1\% \). This indicates that at best, we can only cover 99.1% of the demand. In AN64, since the total base hours (11256) are larger than the demand (11240), the
maximum efficiency is thus 100%, stating that we have a chance to cover all the undersupply hours. Efficiency reflects the performance of the roster and is calculated by the demand covered, over the total demand. Using AN12B as an example, the total undersupply hours covered is $12009 - 1421 = 10588$ hours and the efficiency is thus $10588 / 12009 = 88.2\%$. With a maximum of 99.1\% efficiency, the roster has only achieved 88.2\% and that marks a gap of 10.9\% which is the area for improvement. The average gap for the nine regions can be calculated and is equivalent to $6.113\%$ which is approximately 7871 hours ($6.113\% \times \Sigma \text{Demand}$). This marks a big room for improvement and cost savings. Next, we will view the results of the alternative algorithm (after the 3-stage process).

<table>
<thead>
<tr>
<th>Number</th>
<th>Name</th>
<th>Demand</th>
<th>Total Available Hours</th>
<th>Under-Supplied Hours</th>
<th>Over-Supplied Hours</th>
<th>Maximum Efficiency (%)</th>
<th>Efficiency (%)</th>
<th>Gap (Max Efficiency - Efficiency %)</th>
</tr>
</thead>
<tbody>
<tr>
<td>AN09B</td>
<td>Maidstone West</td>
<td>12,009</td>
<td>11,876</td>
<td>835</td>
<td>765</td>
<td>98.9%</td>
<td>93.0%</td>
<td>5.8%</td>
</tr>
<tr>
<td>AN12B</td>
<td>Guildford</td>
<td>17,122</td>
<td>11,588</td>
<td>5,535</td>
<td>0</td>
<td>67.7%</td>
<td>67.7%</td>
<td>0.0%</td>
</tr>
<tr>
<td>AN31B</td>
<td>Oxford</td>
<td>13,547</td>
<td>13,102</td>
<td>967</td>
<td>873</td>
<td>96.7%</td>
<td>92.9%</td>
<td>3.8%</td>
</tr>
<tr>
<td>AN32A</td>
<td>Reading</td>
<td>16,381</td>
<td>13,694</td>
<td>2,803</td>
<td>181</td>
<td>83.6%</td>
<td>82.9%</td>
<td>0.7%</td>
</tr>
<tr>
<td>AN45B</td>
<td>Skipton</td>
<td>11,287</td>
<td>8,774</td>
<td>2,541</td>
<td>28</td>
<td>77.7%</td>
<td>77.5%</td>
<td>0.2%</td>
</tr>
<tr>
<td>AN50</td>
<td>Wirral</td>
<td>18,605</td>
<td>19,983</td>
<td>812</td>
<td>3,629</td>
<td>100.0%</td>
<td>95.6%</td>
<td>4.4%</td>
</tr>
<tr>
<td>AN55</td>
<td>Northern Ireland</td>
<td>13,683</td>
<td>16,882</td>
<td>166</td>
<td>5,701</td>
<td>100.0%</td>
<td>98.8%</td>
<td>1.2%</td>
</tr>
<tr>
<td>AN62B</td>
<td>Edinburgh East</td>
<td>14,883</td>
<td>15,108</td>
<td>703</td>
<td>1,172</td>
<td>100.0%</td>
<td>95.3%</td>
<td>4.7%</td>
</tr>
<tr>
<td>AN64</td>
<td>Aberdeen</td>
<td>11,240</td>
<td>11,228</td>
<td>865</td>
<td>1,117</td>
<td>99.9%</td>
<td>92.3%</td>
<td>7.6%</td>
</tr>
</tbody>
</table>

**Table 22**

We can see that the alternative algorithm outperforms the current simulated annealing process in every region (note that the total available hours are a little different from table 21). In addition, the gap has improved significantly and the average gap is now 3.168\%. There is an improvement of 2.945\% in gap and this translates to $2.945 / 6.113 = 48.2\%$ improvement, which is equivalent to covering 3793 hours more, over the current roster.
These results were taken from the report dated 02 August 2006 that were submitted to the UK company for review.

The results were improving as the project proceeded and the India engineering office also designed a construction heuristic to obtain more initial feasible solutions (The details of this construction heuristic will not be mentioned). This facilitated the search algorithm as the search space was now enlarged by different initial solutions. Table 23 shows the performance of my heuristic and the India’s office heuristic. Tabu search was conducted on all the solutions.

<table>
<thead>
<tr>
<th>Region</th>
<th>Jun Wei's Undersupply</th>
<th>Tabu</th>
<th>Improvement</th>
<th>India Office's Undersupply</th>
<th>Tabu</th>
<th>Improvement</th>
</tr>
</thead>
<tbody>
<tr>
<td>AN08</td>
<td>3470.25</td>
<td>3009.25</td>
<td>13.28%</td>
<td>3428</td>
<td>2922.25</td>
<td>14.75%</td>
</tr>
<tr>
<td>AN09B</td>
<td>407.5</td>
<td>385.25</td>
<td>5.46%</td>
<td>483</td>
<td>409.25</td>
<td>15.27%</td>
</tr>
<tr>
<td>AN12B</td>
<td>4682</td>
<td>4670.50</td>
<td>0.25%</td>
<td>4692</td>
<td>4672.5</td>
<td>0.42%</td>
</tr>
<tr>
<td>AN31B</td>
<td>555.25</td>
<td>519.50</td>
<td>6.44%</td>
<td>678</td>
<td>568.25</td>
<td>16.19%</td>
</tr>
<tr>
<td>AN32A</td>
<td>2431.75</td>
<td>2310.75</td>
<td>4.98%</td>
<td>2332</td>
<td>2297.5</td>
<td>1.48%</td>
</tr>
<tr>
<td>AN45B</td>
<td>1917.5</td>
<td>1863.75</td>
<td>2.80%</td>
<td>2006</td>
<td>1919</td>
<td>4.34%</td>
</tr>
<tr>
<td>AN50</td>
<td>253.75</td>
<td>233.75</td>
<td>7.88%</td>
<td>436</td>
<td>331.25</td>
<td>24.03%</td>
</tr>
<tr>
<td>AN55</td>
<td>90.75</td>
<td>83.25</td>
<td>8.26%</td>
<td>120</td>
<td>96.25</td>
<td>19.79%</td>
</tr>
<tr>
<td>AN62B</td>
<td>484.25</td>
<td>385.25</td>
<td>20.44%</td>
<td>515</td>
<td>375.25</td>
<td>27.14%</td>
</tr>
<tr>
<td>AN64</td>
<td>470</td>
<td>422.00</td>
<td>10.21%</td>
<td>637</td>
<td>444</td>
<td>30.30%</td>
</tr>
</tbody>
</table>

Table 23
There are ten test regions in this table and the undersupply hours only include the demand in the non-capping hours. We see that the tabu search is able to improve my solution by an average of 8.00% and India office’s solution by 15.37%. The India office construction heuristic did manage to obtain some better results and the best results (after tabu search) are highlighted. The construction heuristic described in this paper still performs generally well compared to the India office’s heuristic. The India office is trying to incorporate other features in the construction heuristic to further improve the
results and features include adding of pre-shift COT hours to cover undersupply as this will not affect the feasibility of the start time. However, this requires the program to keep track of the COT hours assigned. These add-on features were not designed in my heuristic due to time constraint.

Upon careful observation of the data, we realized that a better initial solution would almost definitely produce a better result. For instance, for AN09B, my construction heuristic’s result was 407.5 undersupply hours and the other’s result was 483 undersupply hours. After tabu search, my result of 385.25 hours was still better than the other heuristic’s (409 hours). The other 8 regions revealed similar trends and this implicitly showed that the final result is highly dependent on the quality of the initial solution. Therefore, we have an incentive to further improve our construction heuristic to better our achievements and this will be our objective for the rest of this paper.

Till this point, part 1 has provided you with an understanding of the manpower rostering problem faced by an UK automobile company. We discussed the current roster technique, simulated annealing, and through manual rostering, we found out that there are rooms for improvement. A 3-stage heuristic was designed by the consultancy company to produce an efficient and automated roster process. Stage 1 encompassed the COT/Swap optimization that fixed the workday assignment while stage 2 involved fixing the shift assignment. An initial 3-pass construction heuristic was proposed but it did not perform well, especially in oversupplied regions. The construction heuristic was then revised and gladly, it did produce good initial solutions that were better than the current results. Stage
3 included a tabu search algorithm and the final results marked an impressive 48% improvement.

In the next part of the thesis, I will attempt to build a mathematical model of the construction heuristic and solve it with the CPLEX engine. We will start with a literature review of techniques used in solving similar manpower scheduling problems and discuss some of the methods used in formulating such a complex mathematical model. This will be followed by a detailed explanation of the model used and lastly we will review the results.
PART 2

SOLVING A REAL MANPOWER ROSTERING PROBLEM

THE CPLEX MODEL
1 Literature Review

We begin our literature review by looking at the basic classifications of the manpower-scheduling problems: days-off scheduling, shift scheduling and tour scheduling. We will discuss some of the major developments in these areas. Next, we will look at past models that dealt with the notion of flexibility, start time constraints and total hours’ constraints.

1.1 Days-Off Scheduling

Days-off scheduling deals primarily with the decision of work and rest days for the staffs and the common objectives are balancing of workload among the staffs and maximizing satisfaction of the staffs by altering the pattern of work and rest days (Giovanni et al., 2004). With the assumptions of deterministic weekday and weekend demand, Baker et al. (1977) derived a series of lower bounds for the number of staffs needed under special cases such as each staffs are entitled to 2 rest days or 2 consecutive rest days etc. An algorithm was developed to meet these lower bounds and optimality is reached. Following, Burns et al. (1985) generalized these bounds and develop a simple approach to solve the problem optimally. His results were based on the following conditions:

- Demand is variable on all 7 days.
- The staffs are given A out of B weekends off.
- The staffs work 5 days a week.
- The staffs do not work for more than 6 consecutive days.

Koop (1988) later worked out a multiple shift workforce lower bound that incorporates the effect of constraints. Following, Hung (1994) extended these results by developing an algorithm that could be applied to a multiple shift in a 3-4 workday (weekly) setting. We note that days-off scheduling is applicable to our current rostering problem but our scale
is much smaller. As the patrols are allowed to choose their workday pattern, the planner can only conduct swaps and add COT days to make minor changes to the 6-month workday assignment. Although the COT/Swap optimization is not perfect, we will not attempt to improve it in the rest of this paper.

1.2 Shift Scheduling
Shift scheduling is concerned in determining the start, end and break time of each shift in the planning period. Dantzig (1954) first modeled this class of problems as a set-covering model. Traditionally, the set-covering model consists of selecting a set of workers to perform all the tasks, while minimizing the cost. Dantzig’s formulations for shift scheduling are shown below:

\[
\text{Minimize } \sum_j c_j x_j \\
\text{Subject to: } \sum_j a_{ij} x_j \geq r_i \quad \forall i \\
x_j \geq 0 \quad \forall j \text{ Integer}
\]

\( j \) is the index of the shifts in the planning horizon. \( c_j \) and \( x_j \) refer to the unit cost of employing a staff at shift \( j \) and number of staff hired at shift \( j \) respectively. The objective function is simply the minimization of total workforce cost. \( a_{ij} \) is an integer variable stating if period \( i \) is a work period in shift \( j \) while \( r_i \) is the manpower requirement. \( x_j \) should be non-negative and integer. Segal (1954) formulated this problem as a network problem and eliminated the need to impose the integer constraints. This special structure enables the use of efficient algorithm to solve the model. Break allocation can also be done through the use of a heuristic. Bechtold et al. (1990) made improvement to the model by reformulating the break variables as the number of staffs starting a break at a
time in the planning period. This greatly reduces the number of variables in the integer programming and made the model tractable. In our current model, patrols are given half an hour break for shift length above 5 hours. However, the UK company permitted the project team to ignore the placement of the breaks to reduce the complexity of the problem.

If the planning period spans 24 hours, the shift scheduling is said to be continuous (Jarrah et al., 1994). This creates a new class of problems, cyclic shift scheduling. Bartholdi (1981) proved the NP-completeness of the cyclic scheduling problem. Hung (1991) prescribed 4 tactics to raise efficiency of the cyclic work schedule of nurses at a hospital. Alfares et al. (1998) also designed a 2-phase algorithm to solve the cyclic scheduling problem with 2 consecutive days off each week. For discontinuous problems, the solution procedures can be simpler. Jarrah et al. (1994) managed to decompose the problem into 7 sub-problems (1 for each day) as the labour requirement did not span 24 hours. Thus, they were able to solve a relatively large problem. Although our rostering problem is discontinuous, we are unable to decompose by days. The days cannot be rostered independently as the block constraints relate the days together.

1.3 Tour Scheduling & Flexibility
A tour is a row of continuous workdays and is similar to the concept of a work block in our paper. Tour scheduling is a more general problem than days-off scheduling. The objective is to find an iterating tour schedule to minimize cost over the entire planning horizon. Some heuristics to solve the tour scheduling problems were proposed by scholars such as Bechtold et al. (1994). Giovanni et al. (2004) and Monia et al. (2004)
then produced more compact models to better solve the problem. Since the workday patterns are chosen by the patrols, tour scheduling problem does not exist in our current context. The flexibility of determining the tours is given to the patrols and interestingly, we see that more real life problems are also incorporating employees’ preferences (Bailey, 1985; Kostreva et al., 1989; Sitompul et al., 1993). Barney (1991) emphasized the need for alternative work arrangements that suit the preferences of the workers, to attract and retain quality workers. This has been a long-standing issue in the nursing industry and various methods are deployed to meet the conflicting interest of the hospitals and the nurses. Irem et al. (1988) used a goal programming approach to deal with this problem at the hospital. The conflicting interests are likely to produce infeasible solutions and this necessitates relaxation in some constraints. The constraints thus become goals and the degree of achieving these goals is modeled in the objective function. Seyda et al. (2004) extended this idea by using implicit goal programming approach to solve tour schedule problems while considering staff preferences. However, the UK company requested all preferences to be treated as hard constraints thus there is no need to manage these trade-offs in our problem.

1.4 Start-Time Band & Annualized Hours
Jarral et al. (1994) commented that start-time band should be considered in tour scheduling (Start-time band is similar to the start time variance in our problem). Previous researches on tour scheduling have assumed a start-time band of 1, indicating that the shifts start at the same time in each of the tours. Jacobs and Bruso (1996) explored the possibilities of increasing this band in the tour scheduling problem and they found that scheduling efficiency is increased as the bands increased. In their paper, there is a finite
set of overlapping start time band in the planning period \( \{1,2,3\}, \{2,3,4\}, \{3,4,5\} \) and there are \( m \) workday patterns. The decision variables are the number of employees working in start band \( k \) and on workday pattern \( m \). The aim is to minimize the manpower cost over the planning horizon and identify the number of employees on each workday pattern. Following, Jacobs and Bruso (2000) incorporated meal breaks into the model. However, start time flexibility has not been studied in the context of shift scheduling.

Our manpower rostering problem involves a global constraint on the total hours to be used in the planning period and this is termed as “annualized hours” in some literature. Annualized hours present a new class of rostering problems (Albert et al., 2004). The introduction of annualized hours imputes more flexibility into the rostering process. The companies are allowed to shuffle their manpower to deal with high peak seasons and the resulting distribution of the work hours for the employees may be very uneven for the entire planning horizon. Carlos et al. (2004) used a combination of heuristic and mixed integer programming to solve the annualized hours’ problem under a single shift setting. The solution encompasses the number of work hours to be allocated to each staff in each week and the daily work length to meet the firm’s requirement. With the inclusion of start time band and annualized hours under a multi-shift setting, we can see that our manpower rostering problem is much harder to solve than cases encountered in past literature.
2 The Model
In this section, we will go through the development of the mathematical formulation of
the rostering problem and difficulties faced in each of the model-building phases.

2.1 Basic Formulation
The initial step was to solve a day’s rostering problem. The decision variables should
encompass the start and end time of each shift. In addition, we should be able to capture
the total number of patrols working at any particular time. We used the network model
(figure 5) to accomplish our goal.

The start time and end time nodes represent the time period in the non-capping hours
(0700h to 0000h in the above example). All the edges are binary variables and we let $p$
denote the patrol index, $t$ and $s$ are time indices. $\text{Start}_{p,t}$ denotes that patrol $p$ is starting
the shift at time $t$. $\text{Flow}_{p,t,s}$ indicates that patrol $p$ started the shift in time $t$ and ended in
time $s$. $End_{p,t}$ shows the shift end time $t$ of patrol $p$. To ensure that only a start time and end time is set to 1, we formulate the following model:

\begin{align*}
\sum_{t=1}^{N} Start_{p,t} &= 1 \quad \forall p \tag{1} \\
Start_{p,t} &= \sum_{s=t}^{N} Flow_{p,t,s} \quad \forall p,t \tag{2} \\
\sum_{s=1}^{t} Flow_{p,s,t} &= End_{p,t} \quad \forall p,t \tag{3} \\
\sum_{t=1}^{N} End_{p,t} &= 1 \quad \forall p \tag{4}
\end{align*}

Equation (1) states that for each patrol, only a $Start_{p,t}$ is equated to 1 (i.e. 1 start time permitted only). $N$ is the total number of time periods. Equation (2) models the flow and since start time must be earlier than end time, we will only sum the flow elements for $t$ to $t, t+1, t+2, \ldots N$. For instance, at time 2, we impose the following condition:

$$Start_{p,2} = Flow_{p,2,2} + Flow_{p,2,3} + \ldots + Flow_{p,2,N}$$

If we are starting the shift at time 2, $Flow_{p,2,1}$ cannot be set to 1 and thus need not be considered in the equation. Equation (3) follows the same logic, as the end time must be later than the start time. Equation (4) conserves the flow by permitting only 1 end time.

These 4 sets of equations will accurately model the fact that only 1 start time and end time is allowed for each patrol.

The next step is to model the total number of patrols working in each time period. We let $z_{p,t}$ denotes the working status of patrol $p$ at time $t$ and it is modeled by the following set of equations:

$$z_{p,t} = \sum_{e=t, s e \in N} Flow_{p,e,s} \quad \forall p,t \tag{6}$$
I will illustrate equation (6) with an example in figure 6. Suppose $\text{Flow}_{p,2,4}$ is set to 1 (the patrol starts his shift at time period 2 and ends at time period 4), we should expect $z_{p,2}$, $z_{p,3}$ and $z_{p,4}$ to be set to 1 only.

$z_{p,2}$ is calculated by summing all the $\text{Flow}$ arcs shown in figure 6 (Note that only 1 $\text{Flow}$ variable is 1 while the rest are zero). The patrol must be working at time 2 for $z_{p,2}$ to be 1 and this is only possible if a shift has started in shift 1 or 2 and the shift ends on or after time 2. Therefore, it is sufficient to check these $\text{Flow}$ variables to determine the working status of the patrol at time 2. Following similar arguments, $z$ will be set to 1 at time 3 and 4 while the rest will be zero, indicating non-working status. With the computation of $z$, we can now do a summation of $z$ over all the patrols at time $t$ to find the total number of patrols working at time $t$. If our objective function is to minimize $\sum y_t$, the sum of oversupply and undersupply hours, our basic model will be as follows:
Minimize \[ \sum_{t=1}^{N} y_t \]

Subject to:

\[ \sum_{t=1}^{N} \text{Start}_{p,t} = 1 \quad \forall p \]  
\[ (1) \]

\[ \text{Start}_{p,t} = \sum_{s=t}^{N} \text{Flow}_{p,s,t} \quad \forall p,t \]  
\[ (2) \]

\[ \sum_{s=1}^{t} \text{Flow}_{p,s,t} = \text{End}_{p,t} \quad \forall p,t \]  
\[ (3) \]

\[ \sum_{t=1}^{N} \text{End}_{p,t} = 1 \quad \forall p \]  
\[ (4) \]

\[ z_{p,t} = \sum_{e \in [t,..,N]} \text{Flow}_{p,e,s} \quad \forall p,t \]  
\[ (5) \]

\[ y_t \geq D_t - \sum_{p} z_{p,t} \]  
\[ (6) \]

\[ y_t \geq \sum_{p} z_{p,t} - D_t \]  
\[ (7) \]

\[ \text{Start}_{p,t}, \text{End}_{p,t}, \text{Flow}_{p,t,s} = \{0,1\} \quad \forall p,t,s \]  
\[ (8) \]

Equations (6) and (7) model the shortfall or surplus in the manpower deployment relative to the demand, \( D_t \). This model is tested in ILOG CPLEX and a day’s roster can be solved in a few seconds. (Note that there is still a need to define the flow elements as integer to ensure integer solutions, unlike conventional network flow problems)

### 2.2 Multi-Day Model & Annualized Hours

The basic model can be easily modified to solve a multi-day problem and we will try to incorporate the annualized hours’ constraint (utilization of total base hours) into our model. Define \( d \) as the day indices, \( \text{Work}_{p,d} \) as the working status of patrol \( p \) on day \( d \) and \( \text{TotalHour}_p \) as the total base hours of patrol \( p \) in the planning horizon. The model becomes:
Minimize \[ \sum_{t=1}^{N} y_{d,t} \]

Subject to:

\[ \sum_{t=1}^{N} \text{Start}_{p,d,t} = \text{Work}_{p,d,t} \quad \forall p, d \quad (1) \]

\[ \text{Start}_{p,d,t} = \sum_{s=1}^{N} \text{Flow}_{p,d,t,s} \quad \forall p, d, t \quad (2) \]

\[ \sum_{s=1}^{t} \text{Flow}_{p,d,s,t} = \text{End}_{p,d,t} \quad \forall p, d, t \quad (3) \]

\[ \sum_{t=1}^{N} \text{End}_{p,d,t} = \text{Work}_{p,d,t} \quad \forall p, d \quad (4) \]

\[ z_{p,d,t} = \sum_{e \in \mathbb{I}, t \in \mathbb{N}} \text{Flow}_{p,d,e,s} \quad \forall p, d, t \quad (5) \]

\[ y_{d,t} \geq D_{d,t} - \sum_{p} z_{p,d,t} \quad (6) \]

\[ y_{d,t} \geq \sum_{p} z_{p,d,t} - D_{d,t} \quad (7) \]

\[ \text{Start}_{p,d,t}, \text{End}_{p,d,t}, \text{Flow}_{p,d,t,s}, \text{TotalHour}_{p} = \{0, 1\} \quad \forall p, d, t, s \quad (8) \]

\[ \sum_{d \in T} \sum_{t} z_{p,d,t} = \text{TotalHour}_{p} \quad \forall p \quad (9) \]

All the variables now have day indices as we are solving the 183 days’ problem. For instance, \( z_{p,d,t} \) shows the working status of patrol \( p \), on day \( d \) at time \( t \). RHS of equations (1) and (4) now equate to the day working status of the patrols. If Work\(_{p,d} \) is 0, all the flow elements will be 0 (indicating rest day). The total hour constraint is factored in equation (9). Summing the \( z \) variables across all days and time periods for each patrol gives us the total hour utilized and annualized hour constraint requires the patrols to fully utilize the available base hours. Expectedly, although CPLEX solved the multi-day model (with constraints 1-8) with relatively ease, CPLEX was not able to solve the annualized hours’ constraints, even in the absences of other constraints (start time etc). The annualized hour constraint requires the solver to determine the distribution of the working hours across
the 6-month period and the problem is too large and complicated for the solver to handle. Therefore, we need to find some ways to break down the model into smaller sub-problems. I adopted a *Divide and Conquer* strategy to decompose the original problem, which is similar to the heuristic approach from the alternative algorithm.

### 2.3 Divide & Conquer

The *Divide and Conquer* strategy is about being cost effective (Michalewicz, 2000). The idea is to find an appropriate number of days that can be readily solved. The 183 days are divided into stages and the problem is then solved in stages until all 183 days are solved.

The solutions derived from each stage are then pieced together to form the final solution.

| DAYS | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | ...
|------|---|---|---|---|---|---|---|---|---|----|----|----|----|----|----|----|----|----|---
| P1   | 1 | 1 | 1 | 1 |   | 1 | 1 | 1 | 1 |    | 1 |    | 1 |    | 1 |    | 1 |    | 1 |   |
| P2   |   | 1 | 1 | 1 | 1 | 1 | 1 |   | 1 |    | 1 |    | 1 |    | 1 |    | 1 |    | 1 |   |
| P3   |   |   | 1 | 1 | 1 | 1 | 1 | 1 | 1 |    | 1 |    | 1 |    | 1 |    | 1 |    | 1 |   |
| P4   | 1 | 1 | 1 | 1 |   |   |   | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |   |   |

**Table 24**

Table 24 shows a sample of the workday assignment and in this example, we solve the problem in 6-day stages (day 1-6, 7-12, 13-18...). The issue in this approach is about ensuring the feasibility of the final solution. In the previous construction heuristic, we did the shift assignment on a daily basis and as the algorithm proceeds, we impose the constraints arising from the previous assignments. For instance, if a patrol started his shift on day 1 at 0700h, the feasible start time for day 2 will be restricted to a smaller range (depending on the start time range). In the *Divide and Conquer* strategy, we also used this method to ensure the feasibility of the final solution. Referring to table 24, we see that for patrol 2 (day 7-9) and patrol 3 (day 7-11), the shift assignments are restricted by the solution produced in stage 1.
Although this strategy allows us to reduce the scale of the problem, it does not deal with the annualized hour constraint effectively. Firstly, we must not over-utilize the total base hours in the final solution. An intuitive approach is to prorate the number of hours given to each patrol in each stage by the number of workdays he has in that stage. In table 25, we present the calculations for the hours allocated to stage 1 for each patrol. The hours allocated to stage 1 is calculated by the number of workdays in stage 1 divided by the total number of workdays in the entire planning horizon and then multiply this ratio by the total number of base hours. For patrol 1, the hours allocated to stage 1 are (4 / 105) * 989 = 37 hours. Similar calculations can be done for the rest of the patrols.

<table>
<thead>
<tr>
<th>DAYS</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>…</th>
<th>183</th>
<th>Total Workdays</th>
<th>Total Hours</th>
<th>Total Workdays in Stage 1</th>
<th>Hour allocated to Stage 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>P1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>183</td>
<td>105</td>
<td>989</td>
<td>4</td>
<td>37</td>
</tr>
<tr>
<td>P2</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>114</td>
<td>876</td>
<td>4</td>
<td>4</td>
<td>30</td>
</tr>
<tr>
<td>P3</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>98</td>
<td>1002</td>
<td>2</td>
<td>2</td>
<td>20</td>
</tr>
<tr>
<td>P4</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>100</td>
<td>914</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Table 25
This approach is definitely not the most effective, but it does reduce the magnitude of the annualized hours’ constraints as the solver only has to decide the allocation of the hours in the individual stages. We then impose the constraint that the patrol may not use up more than the allocated hours given in any stage. The unallocated hours at the end of a particular stage will then be added to the allocated hours in the next stage. We thus ensure that we do not over-utilize the total base hours in the final solution. We define $UBHour_{p,s}$ to be the hours allocated to each patrol at stage $s$ and equation (9) is modified to the following:
\[ \sum_d \sum_t z_{p,d,t} \leq UB_{Hour_{p,s}} \quad \forall p \quad (9) \]

However, since we are solving the rostering problem by stages, the solver has no macro perspectives on the utilization of the total base hours. If we incur too much oversupply hours in the early stages, we will be left with little hours to cover the undersupply in the later stages. For example, if the objective function is to minimize total undersupply and this is an undersupplied region (total demand >> total hours), we will use up all the hours in early stages (incurring many oversupply hours), causing heavy undersupply in later stages. The choice of the objective function and the total number of base hours thus influence the final outcome of the result. We modify equations (6) and (7) to the following:

\[
y_{d,t} \geq \alpha \times \left( D_{d,t} - \sum_p z_{p,d,t} \right) \quad (6)
\]

\[
y_{d,t} \geq \beta \times \left( \sum_p z_{p,d,t} - D_{d,t} \right) \quad (7)
\]

\( \alpha \) and \( \beta \) are introduced to calculate \( y_{d,t} \) and these variables affect the weights given to oversupply and undersupply. The intuition is similar to the construction heuristic; in an oversupplied region, we know that we have sufficient hours to cover all undersupply and a higher weight should be given to undersupply hours, punishing the model more on undersupply hours. Vice versa, in an undersupplied region, we should be cautioned in incurring oversupply hours and thus more weight should be given to oversupply hours. The implicit effects of placing more weights on undersupply hours will cause the solver to use more hours at each stage while placing weights on oversupply will conserve the work hours wherever possible. Therefore, varying the parameters \( \alpha \) and \( \beta \) in different
regions allows us to efficiently utilize the hours in each stage. With these settings, we are now ready to formulate the full model.

2.4 The Full Model
Our model has taken care of the annualized hours’ constraints and we will now incorporate other constraints: start time, shift length, maximum block hours, withheld hours and other pre-2005 agreement’s constraints.

We will first model the start time constraints and recall that there are 2 types of start time variances (weekday and weekend). Define $WkDayVar_p$ and $WkEndVar_p$ as the weekday and weekend variance for each patrol, $t$ and $v$ as the indices for the time periods, $d$ and $e$ as the day indices in the current stage, $Block_{p,d}$ as the block index of the workday for patrol $p$, $Weekend_d$ as the binary variables representing the weekend status. The start time constraints are modeled as follows:

$$\left| \sum_t (t \times Start_{p,d,t}) - \sum_v (v \times Start_{p,e,v}) \right| \leq WkDayVar_p \quad (10)$$

$$\forall p,d,e \ni Block_{p,d} = Block_{p,e}, Weekend_d = Weekend_e = 0$$

Equation (10) refers to the weekday variance. $\sum_t (t \times Start_{p,d,t})$ simply returns the value of $t$ when the $Start_{p,d,t}$ is set to 1, which is the start time of the shift for patrol $p$. $\sum_v (v \times Start_{p,e,v})$ thus refers to the difference in start times of the 2 days, $d$ and $e$, for patrol $p$. The weekday variance only applies to the workdays that are weekdays ($Weekend_d = Weekend_e = 0$), in the same block ($Block_{p,d} = Block_{p,e}$).
Correspondingly, equation (11) models the weekend variance, which applies to all days in the block:

\[
\sum_t \left( t \times Start_{p,d,t} \right) - \sum_v \left( v \times Start_{p,e,v} \right) \leq WkEndVar_p \\
\forall p, d, e \in Block_{p,d} = Block_{p,e}
\]  

To ensure the feasibility of the final solution while solving by stages, we need to make sure that the start times that have been assigned to the previous stage are captured and we define \( StartTime_{p,f,t} \) as the binary variable that indicates that the start time \( t \), of patrol \( p \) on day \( f \), which has been rostered in previous stages. These new sets of constraints are pretty much the same as the previous ones.

\[
\sum_t \left( t \times StartTime_{p,f,t} \right) - \sum_v \left( v \times Start_{p,e,v} \right) \leq WkDayVar_p \\
\forall p, d, f \in Block_{p,d}, Weekend_d = Weekend_f = 0
\]

\[
\sum_t \left( t \times StartTime_{p,f,t} \right) - \sum_v \left( v \times Start_{p,e,v} \right) \leq WkEndVar_p \\
\forall p, d, f \in Block_{p,d} = Block_{p,f}
\]

Next, we will formulate the constraints for maximum block hours and shift length. We define a few more sets of data: \( MaxShift_p \) as the maximum shift length for patrol \( p \), \( MinShift_p \) as the minimum shift length for patrol \( p \) and \( UBBlock_{p,b} \) as the maximum hours allowed in block \( b \) of patrol \( p \).

\[
\sum_t z_{p,d,t} \leq MaxShift_p \times Work_{p,d} \quad \forall p, d
\]

\[
\sum_t z_{p,d,t} \geq MinShift_p \times Work_{p,d} \quad \forall p, d
\]

\[
\sum_{d \in b} \sum_t z_{p,d,t} \leq UBBlock_{p,b} \quad \forall p, b
\]

Equations (14) and (15) are straightforward. The \( z \) variables of each day for each patrol are summed and they have to be within the given range. We multiply \( Work_{p,d} \) to make
sure that the model remains feasible when the patrol is not working ($\Sigma z = 0$). Equation (16) takes care of the maximum block hour constraint. For all days in the same block, the $z$ variables are summed and this sum is capped at the upper bound of each block. $UBBlock_{p,b}$ is updated dynamically as we solve the problem in stages. We see an example of the updating process in table 26:

<table>
<thead>
<tr>
<th>Stage</th>
<th>1</th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td>$UBBlock$</td>
<td>55</td>
<td>15</td>
</tr>
<tr>
<td>Workdays</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Shiftlength</td>
<td>10</td>
<td>10</td>
</tr>
</tbody>
</table>

Table 26
Suppose the patrol is on a 6W-3R workday pattern and his workdays are shown above. The maximum block hours are 55 hours and in stage 1 (4 days), we assigned him maximum shift length (10 hours each day). The $UBBlock$ is then updated and changed to 15 hours (55-40), which are distributed to the remaining 2 workdays. This is not a full proof method as there may be a possibility that we used up (say) 46 hours in stage 1 (in this example) and only 9 hours are left in stage 2. Since the minimum shift length for this patrol is 5 hours, the model becomes infeasible (minimum of 10 hours is needed). However, these situations seldom occur and in this case, we can never run into an infeasible scenario.

We are now left with the withheld hour constraint and other pre-2005 agreement’s constraints. Pre-2005 agreement’s constraints include start-by and end-by preferences, and early, late and alternating shift requests. To simplify the model, I have pre-processed these data and translated them to a list of infeasible start time for the patrols. I will illustrate this in table 27 and table 28.
Table 27

Columns 1 shows the workday status of the patrol and suppose there is a withheld hour slot (time period 7, 8, 9) in the block. We will need to make modifications to the infeasible start time. Since we cannot roster the patrol in the withheld hours and minimum shift length is (say) 5 hours, the latest time the patrol may start before the withheld hour slot is time 2. Subsequently, the patrol is prohibited to start at time 3 to 10 (marked by 1). He may also start after the withheld hour slot. The empty cells indicate feasible start times. All the workdays in the block are affected as well.

Table 28

Table 28 shows a patrol with alternating shift request. We define an early shift to be one that starts on or before time 6 and a late shift otherwise. The first block (2\textsuperscript{nd} and 3\textsuperscript{rd} row) has early shifts and the late shift are prohibited (no shifts are allowed to start after time 6. The next block will enforce the opposite and the early shift start times are blocked out. Similar logic follows for the rest of the constraints and a C program was written to pre-process the data. The model formulation is simple:

\[
\begin{align*}
\text{Start}_{p,d,t} = 0 & \quad \forall p,d,t \ni I\text{Start}_{p,d,t} = 1 \\
z_{p,d,t} = 0 & \quad \forall p,d,t \ni I\text{Z}_{p,d,t} = 1
\end{align*}
\]
\(I_{\text{Start}}_{p,d,t}\) indicates the infeasible start time for patrol \(p\) on day \(d\) at time \(t\). \(IZ\) indicates the day \(d\) and the time \(t\), where patrol \(p\) cannot be rostered to work. These time periods belong to the withheld hour slot. For instance, referring to table 27, \(IZ_{p,d,7}, IZ_{p,d,8}\) and \(IZ_{p,d,9}\) will be set to 1.

The formulation is complete and I show the basic structure of the flow control (for solving the stages) and the full model:

\[
\begin{align*}
\text{begin} & \\
\text{determine total number of stages, } k & \\
\text{while current stage } \neq k & \{ \\
\text{solve model} & \\
\text{//Post-Process Directives} & \\
\text{update } \text{Start} & \text{Time, UB} \text{Hour, UB} \text{Block} & \\
\text{generate new model} & \\
\} & \\
\text{end} & \\
\text{Model} & \\
\text{Minimize} & \sum_{t=1}^{N} y_{d,t} & \\
\text{Subject to:} & \\
\sum_{t=1}^{N} I_{\text{Start}}_{p,d,t} = Work_{p,d,t} & \forall p,d & (1) \\
I_{\text{Start}}_{p,d,t} = \sum_{s=t}^{N} Flow_{p,d,t,s} & \forall p,d,t & (2) \\
\sum_{s=t}^{N} Flow_{p,d,s,t} = End_{p,d,t} & \forall p,d,t & (3) \\
\sum_{t=1}^{N} End_{p,d,t} = Work_{p,d,t} & \forall p,d & (4) \\
z_{p,d,t} = \sum_{s \in t,s \in N} Flow_{p,d,s,t} & \forall p,d,t & (5) \\
y_{d,t} \geq \alpha \times \left( D_{d,t} - \sum_{p} z_{p,d,t} \right) & (6)
\end{align*}
\]
\[
y_{d,t} \geq \beta \times \left( \sum_{p} z_{p,d,t} - D_{d,t} \right)
\] (7)

\[\text{Start}_{p,d,t}, \text{End}_{p,d,t}, \text{Flow}_{p,d,t,s} = \{0,1\} \quad \forall p,d,t,s\] (8)

\[
\sum_{d} \sum_{t} z_{p,d,t} \leq UB_{\text{Hour}}_{p,s} \quad \forall p
\] (9)

\[
\left| \sum_{t} (t \times \text{Start}_{p,d,t}) - \sum_{v} (v \times \text{Start}_{p,e,v}) \right| \leq \text{WkDayVar}_{p}
\] \(\forall p,d,e \in \text{Block}_{p,d} = \text{Block}_{p,e}, \text{Weekend}_{d} = \text{Weekend}_{e} = 0\) (10)

\[
\left| \sum_{t} (t \times \text{StartTime}_{p,f,t}) - \sum_{v} (v \times \text{Start}_{p,e,v}) \right| \leq \text{WkEndVar}_{p}
\] \(\forall p,d,f \in \text{Block}_{p,d} = \text{Block}_{p,f}, \text{Weekend}_{d} = \text{Weekend}_{f} = 0\) (11)

\[
\sum_{t} (t \times \text{StartTime}_{p,f,t}) - \sum_{v} (v \times \text{Start}_{p,e,v}) \leq \text{WkEndVar}_{p}
\] \(\forall p,d,f \in \text{Block}_{p,d} = \text{Block}_{p,f}\) (12)

\[
\sum_{p} \sum_{d} z_{p,d,t} \leq \text{MaxShift} \times \text{Work}_{p,d} \quad \forall p,d
\] (14)

\[
\sum_{p} \sum_{d} z_{p,d,t} \geq \text{MinShift} \times \text{Work}_{p,d} \quad \forall p,d
\] (15)

\[
\sum_{d \in b} \sum_{t} z_{p,d,t} \leq \text{UBBlock}_{p,b} \quad \forall p,b
\] (16)

\[
\text{Start}_{p,d,t} = 0 \quad \forall p,d,t \in \text{IStart}_{p,d,t} = 1
\] (17)

\[
z_{p,d,t} = 0 \quad \forall p,d,t \in \text{IZ}_{p,d,t} = 1
\] (18)

This mathematical formulation is translated to an Optimization Programming Language (OPL) and solved using the CPLEX engine in OPL Development Studio 4.1. The complete syntax is attached in appendix C. The formulation is tested on the previous 9 regions and we review them in the next section.
3 Results
In this section, we will take a look at the performance of the CPLEX model and the construction heuristic. Comparison between the CPLEX model and the tabu search is done at the end. I decided on a stage length of 3 days (61 stages in all) because the CPLEX engine was able to solve the 3 days’ problem relatively fast with the full model and this will aid the debugging and testing work. The average time for the CPLEX engine to solve the problem in stages is about 15 minutes and sometimes, a time limit is set if the solver has difficulties in proving optimality.

<table>
<thead>
<tr>
<th>Region</th>
<th>Non-capping Demand</th>
<th>Total Base Hours inc. lunch hours</th>
<th>Optimal</th>
<th>CPLEX</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>Oversupply</td>
<td>Undersupply</td>
</tr>
<tr>
<td>AN09B</td>
<td>11689</td>
<td>12496</td>
<td>807</td>
<td>0</td>
</tr>
<tr>
<td>AN12B</td>
<td>16667</td>
<td>12014</td>
<td>0</td>
<td>4653</td>
</tr>
<tr>
<td>AN31B</td>
<td>13307</td>
<td>13751</td>
<td>444</td>
<td>0</td>
</tr>
<tr>
<td>AN32A</td>
<td>15749</td>
<td>13539</td>
<td>0</td>
<td>2210</td>
</tr>
<tr>
<td>AN45B</td>
<td>11047</td>
<td>9254</td>
<td>0</td>
<td>1793</td>
</tr>
<tr>
<td>AN50</td>
<td>18033</td>
<td>21293</td>
<td>3260</td>
<td>0</td>
</tr>
<tr>
<td>AN55</td>
<td>13612</td>
<td>17715</td>
<td>4103</td>
<td>0</td>
</tr>
<tr>
<td>AN62B</td>
<td>14695</td>
<td>15225</td>
<td>530</td>
<td>0</td>
</tr>
<tr>
<td>AN64</td>
<td>10979</td>
<td>11831</td>
<td>852</td>
<td>0</td>
</tr>
</tbody>
</table>

Table 29
Table 29 shows the result of the CPLEX model, presented in a slightly different format from the previous section. The first column is the non-capping demand and this represents the demand, which we can fully cover in our roster. By incorporating the lunch hours, we are able to obtain an optimal oversupply and undersupply, shown in the next 2 columns. For instance, in AN09B, with a demand of 11689 and total hours of 12496, the optimal undersupply the model can produce is 0 (since available hours >> demand) and the optimal oversupply should be 807. We see that in 2 regions, AN32A and AN12B, we
have obtained the optimal solutions. Now we benchmarked our solution against the original construction heuristic.

<table>
<thead>
<tr>
<th>Region</th>
<th>CPLEX</th>
<th>Junwei’s Construction Heuristic</th>
<th>India Office Construction Heuristic</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Undersupply</td>
<td>Undersupply</td>
<td>Improvement due to CPLEX</td>
</tr>
<tr>
<td>AN09B</td>
<td>393.75</td>
<td>407.5</td>
<td>3.37%</td>
</tr>
<tr>
<td>AN12B</td>
<td>4653</td>
<td>4682</td>
<td>0.62%</td>
</tr>
<tr>
<td>AN31B</td>
<td>489.25</td>
<td>555.25</td>
<td>11.89%</td>
</tr>
<tr>
<td>AN32A</td>
<td>2210</td>
<td>2431.75</td>
<td>9.12%</td>
</tr>
<tr>
<td>AN45B</td>
<td>1831</td>
<td>1917.5</td>
<td>4.51%</td>
</tr>
<tr>
<td>AN50</td>
<td>215</td>
<td>253.75</td>
<td>15.27%</td>
</tr>
<tr>
<td>AN55</td>
<td>55</td>
<td>90.75</td>
<td>39.39%</td>
</tr>
<tr>
<td>AN62B</td>
<td>348</td>
<td>484.25</td>
<td>28.14%</td>
</tr>
<tr>
<td>AN64</td>
<td>387</td>
<td>470</td>
<td>17.66%</td>
</tr>
</tbody>
</table>

Table 30
All the regions shown significant improvements and the average reduction in undersupply (over the better of the 2 construction heuristic) is 68 hours (marking a total reduction of about 680 hours). Since we have obtained the optimal solution for AN12B and AN32A, the actual reduction might have been more. The cost savings will be substantial when we consider all the 100 over regions. We can also compare our results with the tabu search outcome.
Table 31

Table 31 shows the Tabu search results of the 2 construction heuristic. The best results (undersupply), after tabu search, are consolidated in the “Best” column. The CPLEX model even outperforms the tabu search results in 8 of 9 regions. In the previous section, we note that a better initial solution is likely to yield a better result after tabu search. Tabu search also incorporates other constraints such as adding COT hours to increase the maximum shift length allowed (which are not considered in our part). Therefore, the solutions obtained by the CPLEX model are likely to improve further after tabu search.

In addition to the 9 test regions, another region, AN08, was selected. It is one of the largest regions with 44 patrols. The results are reported in table 32.

Table 32

The best undersupply achieved by the better of the 2 construction heuristic is 2922.25 hours. However, the CPLEX model managed to bring it down to 2441. The optimal undersupply is 2271 and this represents a total reduction of 481.25 hours and a 74%
improvement to the original solution which is also far better than the result (after tabu search). Although the running time for this exceptionally large region is about 3 hours, the CPLEX model produced a much better result than the construction heuristic. We note that the previous 9 test regions have about 15-17 patrols on the average and thus the construction heuristic works reasonably well. However, as the number of patrols increases, we expect the performance of the heuristic to drop and the CPLEX model proves itself effective in a large region.
4 Conclusion
As we have seen in the last section, the CPLEX model has performed pretty well, even with large regions, and the main improvements came from the ability of the solver to decide a better start time range (block-wise) for the shifts and it was able to roster effectively within the day, unlike the previous construction heuristic that operates on a micro-basis and does its assignments greedily. In terms of implementation, the mathematical programming approach is more intuitive and user-friendly (compare the programming code length for the construction heuristic and the OPL code!) The data structure used in the mathematical approach is also more natural and we do not need to have complicating data structures, required for the construction heuristic. Violations in the constraints are easily detected, as the solver will return an error on any infeasible model while logic errors are hard to detect in the construction heuristic. Thus, the current mathematical approach can be used more readily than the construction heuristic.

4.1 Contributions
We have successfully dealt with the problem using the Divide and Conquer strategy. The problem size was reduced dramatically when we broke down the problem into 61 stages. We found ways to tackle the annualized hour constraint and the other tricky constraints such as start time, to make the final solution feasible. In the past, manpower scheduling problems could be optimally solved, as shown in "Integrated Days Off and Shift Personnel Scheduling" (Bailey, 1985). However, our manpower rostering problem is a reflection of the increasing complexity in today’s manpower scheduling problems. Sometimes, when the problems get complicated, a search algorithm is preferred. This is shown by the current rostering procedures of the UK automobile company and in "Using
simulated annealing and genetic algorithms to solve staff-scheduling problems", Bailey et al. (1997) also illustrates the application of search algorithms to find near optimal solutions. Marta et al. (2006) used a heuristic approach to solve a crew-scheduling problem. This study has thus shown the flexibility of the using a mathematical approach in solving problems with difficult constraints and it can be readily extended to other real life manpower rostering problems.

4.2 Future Outlook
This study has only experimented with a few possible ways to solve this manpower rostering problem. There are still many ways to further improve the results. Intuitively, the stage length should be directly related to the performance of the algorithm. We may try to assume equal start time in the block to simplify the problem and a post-fixing program can be worked out to improve the quality of the solution by changing the solver solutions. Maikol (2006) also solved a vehicle and crew scheduling using a combination of heuristic and integer programming. An integer programming approach was used to produce good feasible solutions and the heuristic improved the quality of the former. The issue is to find ways to revise the formulations to reduce the number of integer variables and improve the solving speed to solve the problems in longer stages.

4.3 Final Words
This study has attempted to tackle a real manpower rostering problem and has been successful in doing so. Personally, the learning experience is a great one. Not only had I gain many technical skills in mathematical programming and the use of CPLEX engine, I had the chance to understand the complicating scheduling problems faced by companies today. These skills and knowledge will definitely prepare me well for my future work and
I truly thank everyone who has helped me in this learning process.
References


Appendix A – Construction Heuristic VB Code

Declarations of Variables

Public Sub Populate()

    Storing Input Data & Constraints Processing

End Sub

Public Sub Main() ‘Main Program

    Application.ScreenUpdating = False
    Dim Output As Worksheet, found As Boolean, threshold As Single
    Set Output = Worksheets("Output")
    Call Populate 'Populate the arrays

' Determine patrol order, day order & Feasible Start Time

    ReDim Dayorder(0 To 182)
    ReDim Patrolorder(Patrolnum)
    'No Permutation - ordering of days
    For r = 0 To 182
        Dayorder(r) = r
    Next r
    For r = 0 To 182
        dayno = Dayorder(r)
        Call GetStart 'Sub to find start time range
        'For oversupplied Regions
        Call SortStart 'Sub to sort most restricted patrols in front
        'For undersupplied regions
        'Call SortStart_Hours 'Sub to sort most restricted patrols and ties won by patrols with most hours

' Preassignment

    'For oversupplied Regions
    Call Over
    'For undersupplied Regions
    'Call Under

' Finalization

    possible = UBound(Patrolorder)
    For c = 1 To possible
        Patrolno = Patrolorder(c)
        For y = 0 To (Length(Patrolno, dayno, 2) - 1)
            Demand(dayno, Int(Length(Patrolno, dayno, 1) + y)) = _
            Demand(dayno, Int(Length(Patrolno, dayno, 1) + y)) + 1
        Next y
        Length(Patrolno, dayno, 1) = 0
        Length(Patrolno, dayno, 2) = 0
        startLB = Results(Patrolno, 2)
        startUB = Results(Patrolno, 3)
        blockno = Block(Patrolno, dayno)
        minshift = Patrol(Patrolno, 2)

        'Check for COT days and adjust maxshift.
If OT(patrolno, dayno) Then
   OTleft(patrolno) = OTleft(patrolno) - 1
   maxshift = Patrol(patrolno, 4)
Else
   Nleft(patrolno) = Nleft(patrolno) - 1
   maxshift = Patrol(patrolno, 3)
End If

threshold = Int(Patrol(patrolno, 10) + Patrol(patrolno, 11) - minshift * (Nleft(patrolno) + OTleft(patrolno)))

maxshift = min(maxshift, threshold)
maxshift = min(maxshift, Totalhours(patrolno, blockno)) 'Make sure that maxblock not exceeded
If Nice(patrolno, dayno) Then maxshift = min(maxshift, 8) 'Nice day

'Different Objective Functions
'For undersupplied regions. Prevent overallocating all the hours
'maxshift = 7 'Experiment with a few values
'Call Optimal_Under
'For oversupplied regions.
Call Optimal_Over

Length(patrolno, dayno, 1) = startLB + startIncre
Length(patrolno, dayno, 2) = lengthIncre

'Update start time for block
'For post deal and finish by patrols with/without preferences and start by patrols with start time
'violation due to withheld hours
If Pref(patrolno, 1) = 0 Or Pref(patrolno, 3) <> cappingUB Or (Pref(patrolno, 3) = cappingUB And _
   Start(patrolno, blockno, 7) = 1) Then
   If Start(patrolno, blockno, 1) = 0 And Start(patrolno, blockno, 3) = 0 Then
     If Weekend(dayno) = 1 Or Weekend(dayno) = 7 Then
       Start(patrolno, blockno, 3) = startLB + startIncre
       Start(patrolno, blockno, 4) = startLB + startIncre
     Else
       Start(patrolno, blockno, 1) = startLB + startIncre
       Start(patrolno, blockno, 2) = startLB + startIncre
       Start(patrolno, blockno, 3) = startLB + startIncre
       Start(patrolno, blockno, 4) = startLB + startIncre
     End If
   ElseIf Start(patrolno, blockno, 1) = 0 And Start(patrolno, blockno, 3) <> 0 Then
     If Weekend(dayno) = 1 Or Weekend(dayno) = 7 Then
       If startLB + startIncre > Start(patrolno, blockno, 4) Then
         Start(patrolno, blockno, 4) = startLB + startIncre
       ElseIf startLB + startIncre < Start(patrolno, blockno, 3) Then
         Start(patrolno, blockno, 3) = startLB + startIncre
       End If
     Else
       Start(patrolno, blockno, 1) = startLB + startIncre
       Start(patrolno, blockno, 2) = startLB + startIncre
     End If
   ElseIf Start(patrolno, blockno, 1) <> 0 Then
     If Weekend(dayno) = 1 Or Weekend(dayno) = 7 Then
       If startLB + startIncre > Start(patrolno, blockno, 4) Then
         Start(patrolno, blockno, 4) = startLB + startIncre
       ElseIf startLB + startIncre < Start(patrolno, blockno, 3) Then
         Start(patrolno, blockno, 3) = startLB + startIncre
       End If
     Else
       Start(patrolno, blockno, 1) = startLB + startIncre
       Start(patrolno, blockno, 2) = startLB + startIncre
     End If
   End If

74
Else
If startLB + startIncre > Start(patrolno, blockno, 2) Then
    Start(patrolno, blockno, 2) = startLB + startIncre
ElseIf startLB + startIncre < Start(patrolno, blockno, 1) Then
    Start(patrolno, blockno, 1) = startLB + startIncre
End If
If startLB + startIncre > Start(patrolno, blockno, 4) Then
    Start(patrolno, blockno, 4) = startLB + startIncre
ElseIf startLB + startIncre < Start(patrolno, blockno, 3) Then
    Start(patrolno, blockno, 3) = startLB + startIncre
End If
End If
End If

'For first day assigned of the block (start by patrols) To fix the range to adhere to.
ElseIf Start(patrolno, blockno, 5) = 0 Then
    If Weekend(dayno) = 1 Or Weekend(dayno) = 7 Then
        If startLB + startIncre <= Pref(patrolno, 8) Then
            Start(patrolno, blockno, 5) = 1
        Else
            Start(patrolno, blockno, 5) = 2
        End If
    Else
        If startLB + startIncre <= Pref(patrolno, 2) Then
            Start(patrolno, blockno, 5) = 1
        Else
            Start(patrolno, blockno, 5) = 2
        End If
    End If
End If

'Adjust lunch time
If lengthIncre = 5 Then
    lunch = 0
Else
    lunch = 0.5
End If

'Update total hours used in the block and total base/COT hours
Totalhours(patrolno, blockno) = Totalhours(patrolno, blockno) - lengthIncre + lunch
If OT(patrolno, dayno) Then
    Patrol(patrolno, 11) = Patrol(patrolno, 11) - lengthIncre + lunch 'Update avail. COT
Else
    Patrol(patrolno, 10) = Patrol(patrolno, 10) - lengthIncre + lunch 'Update avail. Base
End If
Next c
Next r

'For undersupplied regions to allocate remaining hours
'Call FinalAllocation

Print Results
Appendix B – Construction Heuristic Report

Second Construction Heuristic (Jun Wei)

The second heuristic has two versions – one for oversupplied and one for undersupplied regions. A region is undersupplied if Demand > Total Available Hours and oversupplied if Demand < Total Available Hrs for the planning period. Both algorithms use days as the outer loop in normal calendar sequence and as patrols as inner loop (though the method of permuting the patrols differ in the two algorithms). Both algorithms go through a Pre-assignment phase and a Finalization phase. However, the algorithm for undersupplied regions also goes through an additional phase for allocation of remaining hours.

Construction Heuristic for Oversupplied Regions

Pre-assignment Phase

The general idea of the algorithm is to populate the hours at the start (700h – 900h) and at the end of the day (2000h – 2200h) first. We need to adjust these ranges to suit the capping hours of each particular region. (These time ranges were chosen for AN50 and found to be working reasonably well in the other regions.) Usually demand has two peaks within the day, viz. a morning and an evening peak. If the peaks fall within the selected hours at the beginning and end of the day this allocation works well as no patrol can cover both peaks. After the demand has been fulfilled for either the selected hours at the beginning or the end of day we assign the remaining patrols. The patrols are considered in the order of most to least constrained in terms of starting times. For each patrol determine the best feasible starting time and the allowed shift length, i.e. one that reduces total undersupply the most.

1. Consider the days in chronological order.
2. Start with the first day. Determine the feasible starting times for each patrol. The feasible start times are determined by various constraints such as start time variance, withheld hours, early preference etc. Now order the patrols from the most to the least constrained in terms of starting times. The patrol with the least number of feasible starting times will be placed first. Ties are broken arbitrarily.
3. In the patrol ordering, look for the first patrol able to start at 700h and assign the patrol a shift length of 7 hours. This shift length will be changed during the next phase if necessary.
4. Look for the first unassigned patrol in the order who can start at 1500h and assign the patrol a shift length of 8 hours (till 2300h). All these shift duration are arbitrarily fixed and only serve as a guide to the program.
5. Go back to 700h and check if the demand requirement for the hour is fulfilled. If not repeat Step 3. Else, (i.e., if the demand requirement for 700h is fulfilled) look for the first unassigned patrol able to start at 800h and assign the patrol a shift length of 7 hours.
6. Go back to 2300h and check if the demand requirement for the hour is fulfilled. If not repeat Step 4. Else, (i.e., if the demand requirement for 2300h is fulfilled) we look for the first unassigned patrol in the order who can start at 1400h and assign the patrol a shift length of 8 hours (till 2200h).

7. We will continue this process until either
   a. the required demand has been fulfilled for either the selected hours at the beginning or the end of day;
   b. Or all patrols working on the day have been assigned.
   If (b) occurs terminate. Else continue.

8. Now consider the unassigned patrols in the given order. Assign a shift start times and shift lengths sequentially for patrols. For each patrol determine the best feasible starting time and the allowed shift length, i.e. one that reduces total undersupply the most.

9. Continue till all patrols working on the day have been processed.

10. Continue till all days have been processed.

**Finalization Phase**

In this phase we allow patrols to be “reshuffled” within the working day. The patrols are again considered in the order of most to least constrained in terms of starting times. Each patrol is assigned the best feasible starting time and the allowed shift length, i.e. one that reduces total undersupply the most, while keeping all the other patrols fixed.

1. Consider the days in chronological order.
2. Start with the first day. Determine the feasible starting times for each patrol. The feasible start times are determined by various constraints such as start time variance, withheld hours, early preference etc. Now order the patrols from the most to the least constrained in terms of starting times. The patrol with the least number of feasible starting times will be placed first. Ties are broken arbitrarily.
3. Now consider the patrols in the given order. Assign a shift start time and shift length for the first patrol. For each patrol determine the best feasible starting time and the allowed shift length, i.e. one that reduces total undersupply the most, while keeping the assignment of all other patrols fixed. 
   *The maximum shift length that a patrol may be assigned is* Max. Shift = \( \min(\text{Allowed Max. Shift}, \text{Total Hours Left for the Patrol} - \text{No. of days left} \times \text{Min. Shift}) \)
4. Continue till all patrols working on the day have been processed.
5. Continue till all days have been processed.

**Construction Heuristic for Undersupplied Regions**

**Pre-assignment Phase**

1. Consider the days in chronological order.
2. Start with the first day. Determine the feasible starting times for each patrol. The feasible start times are determined by various constraints such as start time variance, withheld hours, early preference etc. Now order the patrols from the
most to the least constrained in terms of starting times. The patrol with the least number of feasible starting times will be placed first. Ties are broken according to the total number of hours left (i.e. unallocated hours) for each patrol. The one with the most available hours wins the tie, (i.e., is considered earlier.)

**Comment:** The impact of this tiebreaker is significant as the patrols considered first tend to start earlier in the day with more “room” to extend their shift length and they should ideally therefore be the patrols with the more hours.

3. Increase the min shift length of all patrols with a min shift length of 5 hours to 6 hours.*
4. Set max shift length to 7 hours for all patrols.
5. Now consider the unassigned patrols in the given order. Assign a shift start times and shift lengths sequentially for patrols. For each patrol determine the best feasible starting time and the allowed shift length, i.e. one that reduces total undersupply the most.
6. Continue till all patrols working on the day have been processed.
7. Continue till all days have been processed.

*Comment: We run the algorithm with and without Step 3. The better solution is selected at the end.

**Finalization Phase**

This phase is exactly similar to the finalization phase for oversupplied regions except that maximum shift length is still capped at 7 hours.

**Allocation of Remaining Hours**

This step only occurs in the algorithm for undersupplied regions. As we have restricted the max shift length we have many left over hours and plenty of undersupply to cover, and the key is to assign them efficiently to cover as many undersupply hours without incurring excessive oversupply hours. In this phase we process the patrols sequentially and try to extend their shifts. An intuitive way is to look at the ratio of undersupply hours to oversupply hours covered when extending the patrol’s shift. We will extend shifts with higher ratios first. We will continue until no more hours can be added to cover undersupply or when the unallocated hours are used up. The pseudocode is as follows.

1. Order the patrols in order of unallocated hours, with the one with most unallocated hours being considered first. Consider the patrols in this order.
2. Start with the first patrol and look at days on which he is assigned shifts. Determine the days on which it is possible to extend his shift. Consider the first such day.
3. Try extending the patrol’s shift on that day till it an undersupplied hour is encountered. If the patrol’s maximum shift length is reached without encountering an undersupply hour stop. Otherwise (i.e., if an undersupplied hour is encountered) continue trying to extend the shift till an oversupply hour is encountered or the patrol’s maximum shift length is reached. Now examine the extended hours and calculate the ratio of undersupply hours to oversupply hours.
among these additional hours. Suppose his shift is extended to the maximum length possible on a given day. We will check whether each additional hour worked would be an oversupply or undersupply hour. Compute the ratio of undersupply hours to oversupply hours among the additional hours.

(Example 1): If the original shift length is 7 hours (700h-1400h) and the maximum shift length for that day is 10 hours. We will check the demand at 1500h, 1600h and 1700h (Feasible extension slots). If 1500h is an oversupply hour and 1600 and 1700h are undersupply hours, the ratio is 2/1 = 2. If all are undersupply, the day will be given a score of 10. If all are oversupply, the day will be given a score of 0.

Example 2: Suppose the original shift was 5 hours (700h to 1200h) and maximum possible shift length is 10 hours meaning that the possible extension are from 1300h to 1900h.

1st Case
1300 Undersupply
1400 Undersupply
1500 Oversupply
1600 Undersupply
1700 Oversupply
The first undersupply block is from 1300 to 1400h and the checking stops at 1500h where it is an oversupply. The extension for that day will be from 1200h to 1400h (Ratio given will be 10.)

2nd Case
1300h Oversupply
1400 Oversupply
1500 Undersupply
1600h Oversupply
1700 Undersupply
We will check from 1300h to 1600h in this case and assign a ratio of ½. 2 oversupply needed to cover 1 undersupply.

Comment 1: Note that the patrol’s shift is not actually extended in this step. We merely look at possible extensions in order to identify the best day for extending the shift.

Comment 2: If all the hours encountered while trying to extend the shift are undersupply hours the day will be given a score of 10. If all are oversupply hours the day will be given a score of 0 and not considered.

4. Choose the day for which the calculated ratio of undersupply hours to oversupply hours is highest. Extend the patrol’s shift to the end of the block of undersupply hours identified in Step 3.
5. Repeat Step 2.
6. Continue till the patrol has no unallocated hour or it is not possible to reduce the undersupply by extending the working shift on any day.
7. Continue till all patrols have been processed.

(Comment: Shift extensions are done by adding hours at the end. We could add hours at the beginning but this is currently not done).

Implementation Detail: Starting Times
The quality of the starting times is essential to the performance of the algorithms. Therefore, it may be a good idea to restrict the starting time to an even smaller range than the allowed feasible starting times. Generally, starting time range is from 700h (depends on capping hours) to 1800h. We may, e.g., be able to improve algorithm performance by restricting the start time to (say) 1500h at the latest so that no patrol may start after 1500h (although they are legally allowed to do so). The benefit is that the algorithm will not be much affected by the start time variance constraint. If there are sufficient hours, it may be possible to have all shifts start before 1500h as a shift that starts at 1500h can cover the same hours as a shift that starts after 1500h. Although we may waste a few hours, we will ensure that in future moves, we will be less likely to come across a situation whereby we are unable to find a patrol to start at a time we need. Different values of the latest allowed start times can be tested to optimize algorithm performance.

Comment: This step is only used for undersupplied regions. It may be necessary to investigate alternative strategies for stabilizing start time variance for oversupplied regions.
Appendix C – OPL Code

//Script

main {
    thisOplModel.generate();
    var Yap = thisOplModel;
    var NbStage = 61;
    var PeriodIncre = 0;
    var PatrolHourUsed = 0;
    var TotalHourUsed = 0;
    var y = new IloOplOutputFile("InitSolution.txt");
    while (Yap.Stage <= NbStage) {
        writeln(Yap.Stage);
        if (cplex.solve()) {
            //Prepare for next iteration
            var def = Yap.modelDefinition;
            var data = Yap.dataElements;
            //Postprocess
            for (var d in Yap.Day)
                for (var t in Yap.Time)
                    for (var p in Yap.Patrol)
                        data.StartTime[p][d+PeriodIncre][t] = Yap.Start[p][d][t];
                        data.EndTime[p][d+PeriodIncre][t] = Yap.End[p][d][t];
            for (p in Yap.Patrol) {
                PatrolHourUsed = 0;
                for (var b=1;b<=data.MaxBlock;b++) {
                    data.UBBlock[p][b] = Yap.Hour[p][b];
                    PatrolHourUsed += Yap.Hour[p][b];
                }
                if (PatrolHourUsed < data.UBHour[p][data.Stage] &&
                    data.Stage != NbStage) {
                    data.UBHour[p][data.Stage+1] +=
                        (data.UBHour[p][data.Stage]-PatrolHourUsed);
                }
            }
            PeriodIncre += data.NbDay;
            data.Stage++;
            if (data.Stage == NbStage + 1) break;
            data.Period = PeriodIncre;
            Yap = new IloOplModel(def,cplex);
            Yap.addDataSource(data);
            Yap.generate();
        } else {
            writeln("Error!");
            writeln(Yap.dataElements.UBBlock);
            writeln(Yap.dataElements.StartTime);
            writeln(Yap.dataElements.UBHour);
            break;
        }
    }
    //Print Output File & Convert shifts with 11 hours
}
for (p in Yap.Patrol) {
    y.write(p, " ");
    for (d=1;d<=183;d++) {
        if (Yap.Work[p][d] == 1) {
            for (t in Yap.Time) {
                if (Yap.StartTime[p][d][t] == 1) {
                    y.write(t+6,":00 ");
                    Start=t+6;
                }
                if (Yap.EndTime[p][d][t] == 1) {
                    if (t+6+1-Start==11) {
                        y.write(t+6,":45 ");
                    } else {
                        y.write(t+7,":00 ");
                    }
                }
                y.write("0:00 0:00 ");
            } else {
                y.write("NIL ");
            }
            y.writeln();
        }
        y.writeln("EOF");
    }
}

//The Model

int Period =...;
int NbDay =...;
int NbPatrol =...;
int NbTime =...;
int Stage =...;
int MaxBlock =...;
rangle Day = 1..NbDay;
rangle Patrol = 1..NbPatrol;
rangle Time = 1..NbTime;

int Maxshift[Patrol] =...;
int Minshift[Patrol] =...;
int Work[Patrol][1..183] =...;
int Demand[1..183][Time] =...;
int Block[Patrol][1..183] =...;
int UBBBlock[Patrol][1..MaxBlock]=...;
int StartTime[Patrol][1..183][Time]=...;
int EndTime[Patrol][1..183][Time]=...;
int UBHour[Patrol][1..61]=...;
int WeekDay[1..183]=...;
int IStart[Patrol][1..183][Time]=...;
int IZ[Patrol][1..183][Time]=...;
int WkDayVar[Patrol] =...;
int WkEndVar[Patrol]=...;
dvar int+ Start[Patrol][Day][Time] in 0..1;
dvar int+ End[Patrol][Day][Time] in 0..1;
dvar int+ Flow[Patrol][Day][Time][Time] in 0..1;
dvar int+ z[Patrol][Day][Time] in 0..1;
dvar int+ y[Day][Time];
dvar int+ Hour[Patrol][1..MaxBlock];

minimize
    sum (d in Day, t in Time) y[d][t];

subject to 
    //Flow Conservation
    forall (p in Patrol, d in Day)
        sum (t in Time) Start[p][d][t] === Work[p][d+Period];
    forall (p in Patrol, d in Day, t in Time)
        Start[p][d][t] === sum(s in t..NbTime) Flow[p][d][t][s];
    forall (p in Patrol, d in Day, t in Time)
        End[p][d][t] === sum(s in 1..t) Flow[p][d][s][t];
    forall (p in Patrol, d in Day)
        sum (t in Time) End[p][d][t] === Work[p][d+Period];

    //z Computation
    forall (p in Patrol, d in Day, t in Time)
        z[p][d][t] == sum (e in 1..t, s in t..NbTime) Flow[p][d][e][s];

    //Min/Max shift
    forall (p in Patrol, d in Day) 
        sum (t in Time) z[p][d][t] <= Maxshift[p]*Work[p][d+Period];
        sum (t in Time) z[p][d][t] >= Minshift[p]*Work[p][d+Period];
    
    //Start Time Variance
    forall (p in Patrol)
        forall (d in Day, e in d+1..NbDay: Block[p][d+Period]===Block[p][e+Period]
            && WeekDay[d+Period]==0 && WeekDay[e+Period]==0) 
            { 
            sum (t in Time) t*Start[p][d][t] - sum(v in Time)v*Start[p][e][v] <= WkDayVar[p];
            sum (t in Time) t*Start[p][d][t] - sum(v in Time)v*Start[p][e][v] >= -WkDayVar[p];
            };
    forall (p in Patrol)
        forall (d in Day, e in 1..Period: Block[p][d+Period]===Block[p][e]
            && WeekDay[d+Period]==0 && WeekDay[e]==0) 
            { 
            sum (t in Time) t*Start[p][d][t] - sum(v in Time)v*StartTime[p][e][v] <= WkEndVar[p];
            sum (t in Time) t*Start[p][d][t] - sum(v in Time)v*StartTime[p][e][v] >= -WkEndVar[p];
            };
    forall (p in Patrol)
        forall (d in Day, e in d+1..NbDay: Block[p][d+Period]===Block[p][e+Period]) 
        { 
        sum (t in Time) t*Start[p][d][t] - sum(v in Time)v*Start[p][e][v] <= WkEndVar[p];
        sum (t in Time) t*Start[p][d][t] - sum(v in Time)v*Start[p][e][v] >= -WkEndVar[p];
        };
    forall (p in Patrol)
        forall (d in Day, e in 1..Period: Block[p][d+Period]===Block[p][e]) 
        { 
        sum (t in Time) t*Start[p][d][t] - sum(v in Time)v*StartTime[p][e][v] <= WkEndVar[p];
        sum (t in Time) t*Start[p][d][t] - sum(v in Time)v*StartTime[p][e][v] >= -WkEndVar[p];
        };

83
// Block hours constraint
forall (p in Patrol, b in 1..MaxBlock)
  Hour[p, b] == sum (t in Time, d in Day: Block[p, d + Period] == b) z[p, d, t];

// for all (p in Patrol, b in 1..MaxBlock)
  // Hour[p, b] <= UBlock[p, b];

// Individual Hours constraint
forall (p in Patrol)
  sum (d in Day, t in Time) z[p, d, t] <= UHour[p, Stage];

// Withheld Hours Constraint
forall (p in Patrol, d in Day, t in Time) {
  if (IZ[p][d + Period][t] == 1) z[p][d][t] == 0;
  if (IStart[p][d + Period][t] == 1) Start[p][d][t] == 0;
};

// y Computation
forall (d in Day, t in Time) {
  y[d][t] >= 1 * (sum (p in Patrol) z[p][d][t] - Demand[d + Period][t]);
  y[d][t] >= 6 * (Demand[d + Period][t] - sum (p in Patrol) z[p][d][t]);
};