Optimal Dispatcher Workload Distribution

Cipriano A. Santos, Alex Zhang, Ivan Lopez, Glenn Herrick, Mark Raper

HP Laboratories
HPL-2011-195

Keyword(s):

Transportation; Airline Operations; Optimization; Modeling

Abstract:

The Workload Distribution (WKLD) problem is to assign Flight Legs to available Dispatcher Work Positions for a period of time. The primary objective is to balance the workload for Dispatcher Work Position at each hour of the day. Typically, the planning horizon is 30 days, there are approximately 1000 Flight Legs each day, and there are between 20 and 30 Positions open at various times of the day and week. Some Positions are open 24 hours a day. Our solution approach is based on a Mixed Integer Programming (MIP) model to optimally balance the workload for dispatcher work position at each hour of the day. The MIP formulation can be very large - with millions of binary variables; therefore we decomposed the problem into smaller and tractable sub-problems. We have built a Lab prototype to tackle the WKLD problem. The WKLD Lab prototype enables the optimization of the allocation of dispatcher positions to flight legs effort requirements. The WKLD Lab prototype increases safety since by optimizing the dispatcher workload distribution less stressed dispatcher reduces human error. In addition, less stressed dispatcher increases airline operations efficiency.

External Posting Date: October 6, 2011 [Fulltext] Approved for External Publication
Internal Posting Date: October 6, 2011 [Fulltext]

Optimal Dispatcher Workload Distribution

Pano Santos, Alex Zhang, Ivan Lopez, Glenn Herrick,

and Mark Raper

Creation: 06/09/11.
Revision: 08/03/11,08/21/11, 09/29/11

Abstract

The Workload Distribution (WKLD) problem is to assign Flight Legs to available Dispatcher Work Positions for a period of time. The primary objective is to balance the workload for Dispatcher Work Position at each hour of the day.

Typically, the planning horizon is 30 days, there are approximately 1000 Flight Legs each day, and there are between 20 and 30 Positions open at various times of the day and week. Some Positions are open 24 hours a day.

Our solution approach is based on a Mixed Integer Programming (MIP) model to optimally balance the workload for dispatcher work position at each hour of the day. The MIP formulation can be very large –with millions of binary variables; therefore we decomposed the problem into smaller and tractable sub-problems.

We have built a Lab prototype to tackle the WKLD problem. The WKLD Lab prototype enables the optimization of the allocation of dispatcher positions to flight legs effort requirements. The WKLD Lab prototype increases safety since by optimizing the dispatcher workload distribution less stressed dispatcher reduces human error. In addition, less stressed dispatcher increases airline operations efficiency.

1- Introduction

The Flight Planning product, an asset of HPES-Agilaire, contains a component called Workload Distribution (WKLD). This component is responsible for distributing flight assignments across a number of dispatcher positions as efficiently as possible. The distribution criteria are complex, and clients desire an "optimal" solution.

The Workload Distribution problem is to assign Flight Legs to available Dispatcher Work Positions for a period of time. The primary objective is to balance the workload for Dispatcher Work Position at each hour of the day.

Typically simple heuristics and rules of thumb are used for workload balancing. However, the HPES-Agilaire customers are requesting mathematical optimization technology, but HPES-Agilaire does not have a go-to-market mathematical optimization capability, and mathematical optimization contractors are expensive; e.g. $3k USD / FTE / day.

A custom development is unusually risky because it is difficult to know if we have agreement on requirements before we build a proof-of-concept -or possibly the entire system. Using an optimization engine mitigates the risk above; because the client knows it is the right tool for the job and is already biased to accept it.

The challenges described earlier present an opportunity for collaboration between HPES-Agilaire and HP Labs to drive better business performance. Flight Planning has a well-scoped problem; however, it lacks a delivery capability for a mathematical optimization model. The Services Research Laboratory (SRL) has an advanced technical capability and is always searching for opportunities to prove the value of this technical capability across HP, including HP Enterprise Services. Flight Planning providing the delivery context for SRL to develop production-grade technical content is synergistic. Flight Planning meets its commitments on time and on budget, while SRL receives an excellent success story to prove the value of their research. Together, HP is able to deliver on a workload distribution optimization solution with efficiency not possible without this internal cooperation.

2- Problem description

Typically, the planning horizon is 30 days, there are approximately 1000 Flight Legs each day, and there are between 20 and 30 Positions open at various times of the day and week. Some Positions are open 24 hours a day.

The collection of Flight Legs for the planning horizon is predetermined, and is an input to the model.

Each Flight Leg has the following attributes:

- FlightNumber
- FlightStartDate
- DepartureStationCode
- ArrivalStationCode
- InternationalFlightLegInd (International or Domestic)
- StartOfPlanningTime
- ScheduledReleaseTime
- ScheduledArrivalTime

Each individual Flight Leg has an hourly calendar built that represents the life of the flight. The life of the Flight Leg has two main periods; the Planning Period and the Following Period. The Planning Period is from the Start of Planning Time to the Scheduled Dispatcher Release Time. The Following Period is from the Scheduled Dispatcher Release Time to the Scheduled Arrival Time. The effort requirement of a Flight Leg is the total working effort (number of minutes) for each particular hour of the flight (actual calendar date and hour). There are three types of effort requirements: Planning, PlanningAndFollowing –i.e. there are planning and following effort requirements at the same hour, or Following.

The determination of effort needed for each specific flight leg is predetermined. That determination is based upon flight duration, international/domestic type, station characteristics, and seasonal considerations. The effort for the planning and following periods can be different for a flight leg.

The collection of Dispatcher Positions (working schedule and number of Positions) for the planning horizon is predetermined.

Each Position has the following attributes:

- PositionNumber
- OpenTime
- CloseTime
- Frequency (Days of the Week Position is Open)
- InternationalPositionIndicator (International or Domestic)
- UtilizationPercent (Normally 100%, 60 minutes per hour, but can be less for a dispatch position that has other responsibilities.)
- OverUtilizationPercent (Percentage a position can be over-utilized in any hour.)

Each individual Position has an hourly calendar built that represents the working schedule for the Position. Each hourly calendar contains the maximum number of minutes per hour that the Position is available to perform work as normal utilization during each actual day and hour of the distribution period. The Position can be allowed to be over utilized up to a maximum number of minutes.

The following key constraints and rules must be followed when allocating flight leg effort requirements to a set of open positions throughout the life of the flight.

- The amount of work assigned to each Position for each hour should not vary significantly from Position to Position.
- All effort required for a Flight Leg during the life of the flight should be assigned to a single Position –as much as possible.
- If a single Position cannot be identified to accommodate all of the effort for the Flight Leg, the Flight Leg effort can be split between 2 or 3 positions, at different hours during the life of the flight, according to the following rules:
    - The entire Planning Period -where the effort type is Planning or Planning and Following, must be assigned to the 1st Position.
    - The 1st Position is closing during the Following Period of the flight. The 1st Position is closing if there is a sequential break in the working hours of the Position.
        - The Following Period can be assigned to a 2nd or 3rd Position in this situation.
    - All Flight Legs for a specific 1st Position that are assigned to a 2nd or 3rd Position when the 1st Position closes should use the same 2nd or 3rd Positions -as consistently as possible.
- The number of unassigned Flight Legs should be minimized.

- Flight Legs that are repeated on different dates with the same flight number, arrival and departure station (daily flight legs) should be assigned to the same Position as often as possible over the Distribution period.
- Some Flight Legs may be pre-assigned to specific Positions prior to general distribution of Flight Legs. This can occur because of a specific preference by the user.
- The maximum Position utilization (normal plus over capacity) during any hour should not be exceeded.

These rules can be summarized into the following goals.

1- Evenness: *Minimize the Deviation of all Work Positions assigned effort for each hour of the day. The amount of work assigned to each Position should not vary significantly from Position to Position with consideration to the normal capacity of the position*. That is, ideally all active positions at each hour of the day should have the same effort allocated to them.

2-Consistency: *Assign Daily Flight Legs to the Position(s) with open times corresponding to the Daily Flight Leg the greatest number of times over the distribution period*. That is, ideally all replicas/duplicates of a Flight-leg should be allocated to the same first position for each day of the distribution period; and all closing positions should complete all the effort requirements of all the flight-legs they are allocated to.

3-Minimize Assignments: *Minimize number of positions assigned to a flight leg, i.e. limit the usage of Second and Third Position values*. Ideally the Planning & Following periods of a flight-leg should be assigned to a single position, the Planning period cannot be split, and there should be at most 3 positions allocated to a flight-leg.

4-Completeness: *Minimize the remaining Flight Leg Effort of unassigned Flight Legs*. Ideally all Flight-leg efforts should be filled by an open position, i.e. there should be no remaining unassigned Flight Legs.

There are conflicts and therefore trade-offs among these optimization goals. Specifically, there is a trade-off between the Evenness goal and the Consistency/Assignment goals. The Consistency/Assignment goals will tend to consume capacity from positions with large capacity, typically 24 hours positions, leaving positions with little capacity idle, consequently hurting the Evenness goal. Conversely, the Evenness goal will try to split flight-legs in order to consume the capacity of all positions –with large and little capacity, hence hurting the Consistency/Assignment goals.

3- Solution approach

The main purpose of the Workload Distribution (WKLD) problem is to assign Flight Legs to available Dispatcher Work Positions for a period of time.

The main goal is to balance the workload for dispatcher work position at each hour of the day. Several rules and constraints need to be satisfied, for example: 1) flights that are repeated on different dates, should be consistently assigned to the same position, 2) flight allocation can be split between 2 or 3 positions if the 1st position can accommodate all the effort required during the planning period and the 1st position is closing during any of the hours of the following period.

Our solution approach is based on a Mixed Integer Programming (MIP) model to optimally balance the workload for dispatcher work position at each hour of the day.

The main decision variable of the MIP model is a binary variable that determines if a position fills completely the effort requirement of a flight at a given hour during the life of the flight.

The objective function is to minimize the summation, over each hour in the planning horizon, of the absolute value of the difference between the utilization of a position open at an hour and the average utilization of all open positions at such hour.

The constraints of the MIP formulation are as follows.

1. Satisfy the effort requirements of all flight legs during the planning horizon. For each hour during the life of the flight allocate a position to fill effort requirements, otherwise declare the effort requirements at such hours as unfilled.
2. Compute the total effort allocated at each open position for each hour of the planning horizon.
3. For each open position at each hour of the planning horizon, the total effort allocated should be less than or equal to the regular capacity plus overtime.
4. For each open position at each hour of the planning horizon, the overtime allocated to satisfy total effort allocated should be less than or equal to a maximum limit of over capacity.
5. For each open position at each hour of the planning horizon, compute the absolute value of the difference between the utilization of the open position at the hour and the average utilization of all open positions at that hour.
6. For each open position and a flight leg, this constraint indicates if such position has been allocated to the flight –at any hour during the life of the flight.
7. For each flight leg assign at most three positions to fill the effort requirements during each hour of the life of the flight.
    a. For each open position and a flight leg, this constraint ensures that all the effort requirements during hours of the planning period are filled by the same single position.

b. For each open position and a flight leg, this constraint indicates if a single position has been used to fill all the effort requirements during the life of the flight.

The rest of the constraints are logical constraints that ensure that the Flight-leg splitting rules are properly considered.

The details of the WKLD-MIP model are described in appendix A.

The MIP formulation can be very large –with millions of binary variables; therefore we need to decompose the problem. We note that there are two main types of flights: cycle flights and extraordinary flights. Cycle flights repeat at several days during the planning horizon -with exactly the same planning and following hours and the same effort requirement at each hour. Extraordinary flights are non-cycle flights, i.e. flights that do not repeat.

For the cycle flights we develop the concept of meta-week to aggregate -or fold, all the flights that repeat at several days during the planning horizon into a single week –the meta-week. In this way we define the Monday flights as the flights that are identical at all Mondays during the planning horizon. Similarly we define the Tuesday, Wednesday… and Sunday flights. Hence one flight partition will be the flights that repeat at all days of the meta-week, plus all the extra days in the planning horizon.

In Figure 1 we illustrate the concept of meta-week. In this example we have a planning horizon starting on Sunday the 9^{th} and finishing on Saturday the 22^{nd}, a Pre-processing phase takes the flight-legs that are identical on each day of the meta-week (Sunday, Monday… Friday, Saturday). Then the Sunday flights includes the flight-legs that are identical on Sunday the 9^{th} and 16^{th}, the Monday flights includes the flight-legs that are identical on Monday the 10^{th} and 17^{th}, and so on and so forth.

For each of these partitions we consider a sub-partition of flights that have the same planning and following hours. For example a partition F5_7 means cycle flights that repeats 5 days of the meta-week and have a planning and following duration of 7 hours.

Partitions are ordered by higher repetition to lower repetition, and for each of these partitions, the sub-partitions are ordered by larger duration to lower duration.

As a heuristic, we first solve a series of MIP problems for the cycle flights considering the flight partitions described above. We solve one flight partition at a time in the order described above. Note that the planning horizon for the cycle flights MIP problem is a meta-week. The main output of the MIP is a flight-position allocation at each hour during the life of the flight and flight effort requirements that could not be filled at some hours during the life of the flight. Given the flight-position allocations we can compute the total effort allocated at each open position and hour of the meta-week.

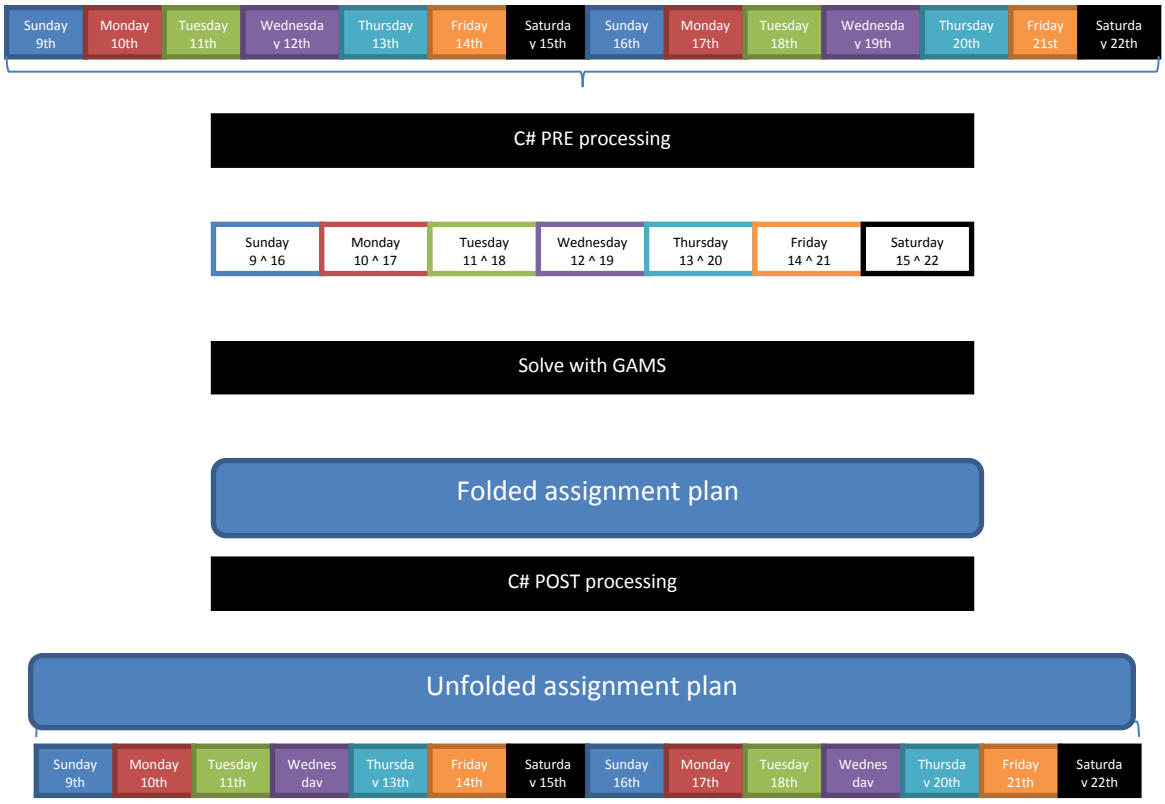

Figure 1: Meta-week concept.

Then we unfold the total effort allocated at each open position and hour of the meta-week into the hours of the original planning horizon, and compute the remaining capacity available (regular and over capacity) at each open position and hour of the original planning horizon. This remaining capacity available is considered now to tackle the extraordinary flights problem, and for this problem we create flight partitions based on the duration of the flights –i.e. number of planning and following hours.

Finally we solve a series of MIP problems for the extraordinary flights considering flight partitions from larger duration to smaller duration.

A MIP model -for cycle and extraordinary flights, is implemented in GAMS –GAMS[1] is a commercial mathematical modeling language. GAMS calls a commercial solver - Gurobi[2], to solve the MIP model. The pre-processing and post-processing phases are coded in c#.

[1] http://www.gams.com/
[2] http://www.gurobi.com/html/products.html

4- Lab prototype for the Workload Distribution (WKLD) problem

We have built a Lab prototype to tackle the WKLD problem. The following screen shot – Figure 2, describes the GUI of the Lab prototype. In the Directory Path field the user specifies the directory that contains the data and code. The user by pressing the Detect XML Files button activates the search for the XML files containing Flight Leg and Position Schedule data. The user then set the mathematical optimization model to use in the list from the Model to use field. Finally, the user presses the Run Optimization button to start the process. The screen at the bottom of the GUI reports the date and start time of each of the main components of the WKLD prototype. The details of the Lab prototype are described in Appendix B.

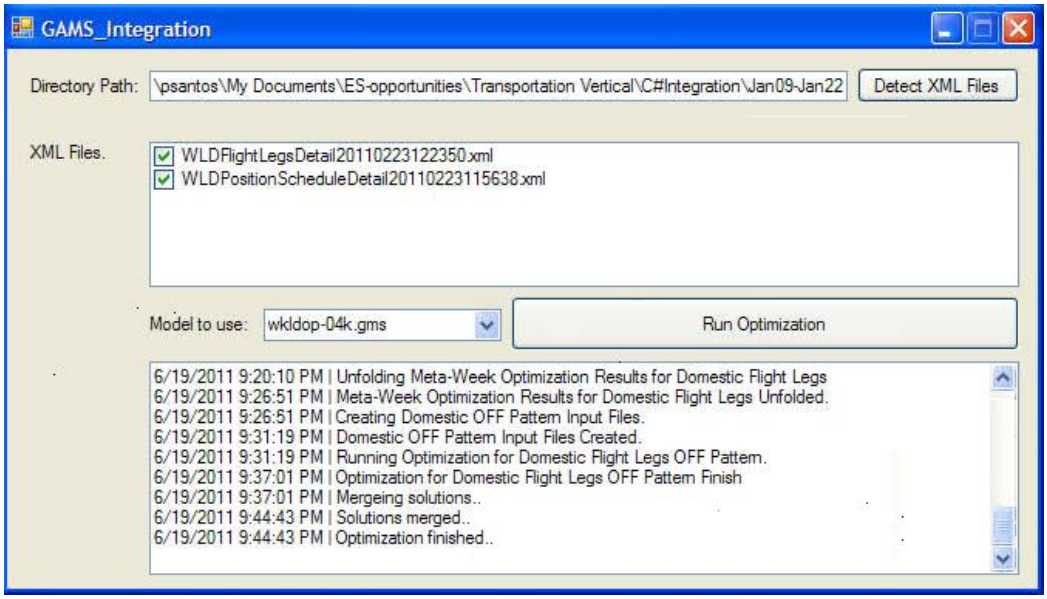

Figure 2: GUI of the Lab prototype.

The main steps of WKLD Lab prototype are

Step 1: Read XML data with flight leg and dispatcher positions information.

Step 2: Identify cycle and extraordinary flights.

Step 3: Aggregate (fold) cycle flight data into a meta-week.

Step 4: Generate text (.txt) files as an input for the cycle flight MIP problem.
(Run input data validation test and report issues when found.)

Step 5: Run GAMS/Gurobi to determine the flight-position allocation that minimize the total evenness deviations of cycle flights.
(Run output results validation test and report issues when found.)

Step 6: Compute total effort allocated at each open position and hour of the meta-week based on the flight-position allocations determined by GAMS/Gurobi.

Step 7: Unfold the total effort allocated at each open position and each hour of the original planning horizon.

Step 8: Compute remaining capacity available (regular and over capacity) at each open position and each hour of the original planning horizon.

Step 9: Generate text (.txt) files as an input for the extraordinary flight MIP problem.
(Run input data validation test and report issues when found.)

Step 10: Run GAMS/Gurobi to determine the flight-position allocation that minimizes the total evenness deviations of extraordinary flights.
(Run output results validation test and report issues when found.)

Step 11: Generate flight-position allocation reports.

5- Tests of WKLD Lab prototype

In this section, we describe a test of the WKLD Lab prototype considering COA real data from Jan09-2011 to Jan22-2011.

The following is the report created by the WKLD Lab prototype-GUI from the test data, describing the start time of the main components of the Lab prototype.

```
============================================================
6/28/2011 9:37:06 PM | Detecting XML Files in C:\Documents and Settings\psantos\My
Documents\ES-opportunities\Transportation Vertical\C#Integration\Jan09-Jan22.
6/28/2011 9:37:06 PM | Adding file: WLDFlightLegsDetail20110223122350.xml.
6/28/2011 9:37:06 PM | Adding file: WLDPositionScheduleDetail20110223115638.xml.
6/28/2011 9:37:15 PM | Starting data structures load.
6/28/2011 9:37:15 PM | Loading file: WLDFlightLegsDetail20110223122350.xml.
6/28/2011 9:37:22 PM | File: WLDFlightLegsDetail20110223122350.xml loaded.
6/28/2011 9:37:22 PM | Loading file: WLDPositionScheduleDetail20110223115638.xml.
6/28/2011 9:37:23 PM | File: WLDPositionScheduleDetail20110223115638.xml loaded.
6/28/2011 9:37:23 PM | Data structures loaded.
6/28/2011 9:37:23 PM | Starting Classification Domestic - International.
6/28/2011 9:37:23 PM | Classification Domestic - International Ended.
6/28/2011 9:37:23 PM | Creating International Meta-Week Input Files.
6/28/2011 9:44:35 PM | International Meta-Week Input Files Created.
6/28/2011 9:44:35 PM | Running Optimization for International Flight Legs Meta-Week.
6/28/2011 10:40:29 PM | Optimization for International Flight Legs Meta-Week Finish.
6/28/2011 10:40:29 PM | Unfolding Meta-Week Optimization Results for International Flight Legs.
6/28/2011 10:43:24 PM | Meta-Week Optimization Results for International Flight Legs Unfolded.
6/28/2011 10:43:24 PM | Creating International OFF Pattern Input Files.
6/28/2011 10:45:58 PM | International OFF Pattern Input Files Created.
6/28/2011 10:45:58 PM | Running Optimization for International Flight Legs OFF Pattern.
6/28/2011 11:03:51 PM | Optimization for International Flight Legs OFF Pattern Finish.
6/28/2011 11:03:51 PM | Merging solutions..
6/28/2011 11:10:28 PM | Solutions merged..
6/28/2011 11:10:28 PM | Solving Domestic Meta-Week.
6/28/2011 11:10:28 PM | Creating Domestic Meta-Week Input Files.
6/28/2011 11:27:07 PM | Domestic Meta-Week Input Files Created.
6/28/2011 11:27:07 PM | Running Optimization for Domestic Flight Legs Meta-Week.
6/29/2011 2:30:38 AM | Optimization for Domestic Flight Legs Meta-Week Finish.
6/29/2011 2:30:39 AM | Unfolding Meta-Week Optimization Results for Domestic Flight Legs.
6/29/2011 2:36:17 AM | Meta-Week Optimization Results for Domestic Flight Legs Unfolded.
6/29/2011 2:36:17 AM | Creating Domestic OFF Pattern Input Files.
6/29/2011 2:40:00 AM | Domestic OFF Pattern Input Files Created.
6/29/2011 2:40:00 AM | Running Optimization for Domestic Flight Legs OFF Pattern.
6/29/2011 3:15:16 AM | Optimization for Domestic Flight Legs OFF Pattern Finish.
6/29/2011 3:15:16 AM | Merging solutions..
6/29/2011 3:26:35 AM | Solutions merged..
6/29/2011 3:26:35 AM | Optimization finished..
============================================================
```

Notice that the WKLD Lab prototype took 5 hours and 49 minutes approximately, running in an HP desktop Intel(R) Xeon(R) CPU X5450 @3.00GHz 3.00GB of RAM.

The main output of the WKLD Lab prototype is an allocation of positions to flight legs at a day and hour of the planning horizon. For example, in Table 1 we have a report that says that dispatcher position # 12 has been allocated to Flight Leg CO0001IAHHNL on Tuesday January 18, 2011 from the 1500 hours to the 2100 hours and the effort requirement at each hour is of 1 minute.

Position-id	Flight-Leg-id	Day	hour-id	Effort requirements (min)
12	CO0001IAHHNL	Tue	201101181500	1.00
12	CO0001IAHHNL	Tue	201101181600	1.00
12	CO0001IAHHNL	Tue	201101181700	1.00
12	CO0001IAHHNL	Tue	201101181800	1.00
12	CO0001IAHHNL	Tue	201101181900	1.00
12	CO0001IAHHNL	Tue	201101182000	1.00
12	CO0001IAHHNL	Tue	201101182100	1.00

Table 1: Allocation of a position to a flight-leg.

Another important output report that summarizes the overall Flight-Position Allocation is presented in Figure 3. The x-axis has the hours-id in the planning horizon, and the y-axis is in effort minutes. The blue line shows the total **capacity** at each hour-id of the planning horizon. The red line shows the **effort requirements**. The purple line shows **effort requirements** that have been **filled** –this line covers filled demand. Finally, the green line shows the **effort requirements** that could **not** be **filled**.

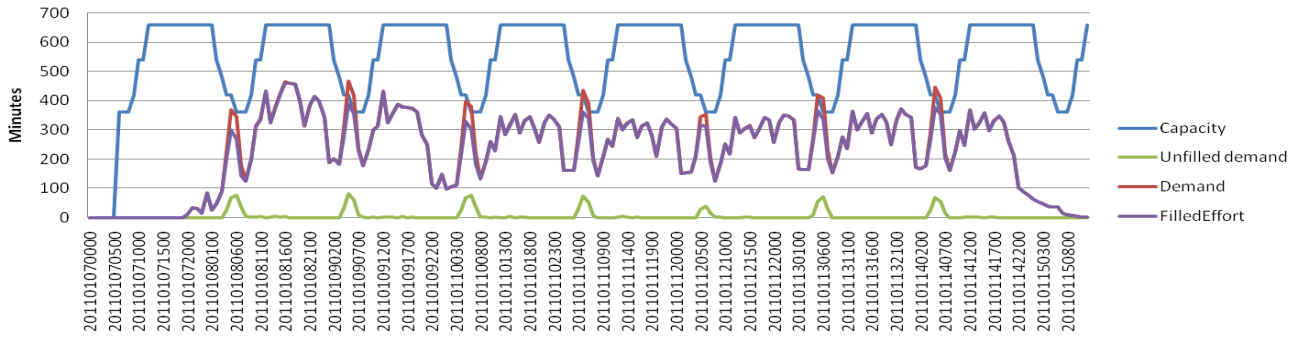

Figure 3: Flight-Position Allocation

Unfilled effort requirements are caused by lack of capacity (regular and overtime) at the hour, or because some splitting rule was violated.

6- Conclusions and future research

The perceived benefits of the WKLD Lab prototype follow.

1) The WKLD Lab prototype enables the optimization of the allocation of dispatcher positions to flight legs effort requirements.

2) The WKLD Lab prototype minimizes the total deviation of the utilization of an open position at an hour and the average utilization of all open positions at such hour, for all open position and hours of the planning horizon. Consequently this prototype solution balances and optimizes the workload distribution of open positions at each hour during the planning horizon, i.e. the amount of work assigned to each Position should not vary significantly from Position to Position.

3) The WKLD Lab prototype enables the rule that says all effort requirement of a Flight Leg at an hour during the life of the flight should be assigned to a single Position.

4) The WKLD Lab prototype enables the rule that says if a single Position cannot be identified to accommodate all of the effort for the Flight Leg, the Flight Leg effort can be split between 2 or 3 positions, at different hours during the life of the flight, according to the following rules.

> 4.1) The entire Planning Period -where the effort type is Planning or PlanningAndFollowing, must be assigned to the 1st Position.
>
> 4.2) The 1st Position is closing during the Following Period of the flight. The 1st Position is closing if there is a sequential break in the working hours of the Position.
>
> > 4.2.1) The Following Period can be assigned to a 2nd or 3rd Position in this situation.

4.3) All Flight Legs for a specific 1st Position that are assigned to a 2nd or 3rd Position when the 1st Position closes should use the same 2nd or 3rd Positions as consistently as possible.

5) The WKLD Lab prototype enables the rule that says the number of unassigned Flight Legs should be minimized.

6) The WKLD Lab prototype enables the rule that says Flight Legs repeated on different dates with the same flight number, arrival and departure station (daily flight legs) should be assigned to the same Position over the Distribution period as much as possible.

7) The WKLD Lab prototype enables the rule that says some Flight Legs may be pre-assigned to specific Positions prior to general distribution of Flight Legs.

8) The WKLD Lab prototype enables the rule that says the maximum Position utilization (regular plus over capacity) during any hour should not be exceeded.

9) The WKLD Lab prototype increases safety since by optimizing the dispatcher workload distribution less stressed dispatcher reduce human error. In addition, less stressed dispatcher increases airline operations efficiency.

The future research includes the following areas for investigation.

1) We need to investigate more efficient approaches to find a 'good' flight-leg and position allocation. These approaches may include *column generation* based algorithms, or approximation algorithms based on *Extended Bin Packing* problems. Our research goal is to solve effectively the Workload Distribution problem in minutes. Approximation algorithms or efficient heuristics for this problem are appropriate, since there is no clear definition of the objective function to optimize, consequently we cannot really talk about the "optimal" solution.

2) The solution of the WKLD problem provides an allocation of flight-legs to positions; consequently this allocation can be interpreted as a forecast of flight-legs effort requirements at each position. Hence, an underlying problem is a dispatcher rostering problem that assigns dispatchers to positions-flights effort requirements considering labor rules for airline dispatchers, dispatchers' utilization, and dispatchers' preferences.

3) During the daily airline dispatcher operations there are many uncertain events that modify the effort requirements of flight-legs. Weather is the main factor that modifies the effort requirements of flight-legs; hence there is the need of real time reassignment and redeployment of dispatchers to fill new effort requirements of flight-legs. Therefore, there is a need to develop effective and very fast algorithms to reassign dispatchers in real time.

Appendix A: Mixed Integer Programming Formulation of WKLD problem

Indices

$f \in F$: Index and set of flights

$i, j \in I$: Index and set of positions

$h \in H$: Index and set of hours during planning horizon (1 month)

r : Index for replicas of flight leg f. A replica of a flight leg is really the day that the flight departs.

Parameters

$effor(f,r,h)$: This is the effort requirements (in min) of replicate r of flight leg f during each hour h of planning horizon.

$plan(f,r,h)$: $plan(f,r,h) = 1$ indicates if hour h is a planning period of replica r of flight leg f.

$follow(f,r,h)$: $follow(f,r,h) = 1$ indicates if hour h is a following period of replica r of flight leg f. This set is also called FOLLOWSET.

$regcap(i,h) \geq 0$: This is the regular capacity available (in min) of position i during each hour h of planning horizon.

$overcap(i,h) \geq 0$: This is the over capacity available (in min) of position i during each hour h of planning horizon.

maxpos(f): Maximum number of positions allowed for flight leg f. Typically 3 positions

Computed Parameters

$m(f,r,h)$: indicates if replica r of flight leg f requires an effort during hour h.

$m(f,r,h) = 1$ if $effor(f,r,h) > 0$, and 0 otherwise.

$replica(f,r)$: indicates that r is a replica of flight leg f.

$replica(f,r) = 1$ if $\sum_h m(f,r,h) > 0$, and 0 otherwise.

$numrep(f)$: Number of replicas of flight leg f.

$numrep(f) = \sum_r replica(f,r)$.

$numpos(h) = \sum_{i \in I} 1_{regcap(i,h)>0}$, i.e. number of open positions at hour h.

$frh = \{(f,r,h) : effor(f,r,h) > 0\}$: Set of replicas of flights that required effort at each hour of planning horizon.

$ih = \{(i,h) : regcap(i,h) > 0\}$: Set of open positions at each hour of planning horizon.

$ifrh = \{(i,f,r,h) : regcap(i,h) * effor(f,r,h) > 0\}$: Set of positions that can fill effort requirements of replicas of flight legs at each hour of planning horizon.

$ifr = \{(i,f,r) : \bigcup_{h \in H} ifrh(h)\}$: Set of valid position, flight leg, and replica.

$pf = \{(i, f) : \bigcup_r ifr(r)\}$: Set of valid position, and flight leg.

$(i, j, f, r, h-1, h) \in ijfrh1h$: For a flight leg f at a replica r this is the set of feasible combinations of switching from position i at hour h-1 to position j at hour h.

$(i, j, f, r, h) \in ijfrh$: This set (table) is derived from set $ijfrh1h$ by removing the column associated with hour h-1.

The calculation of these data structures ($ijfrh1h$, $ijfrh$) are explained in Appendix C of this document.

Decision Variables

$z_{i,f,r,h} = 1$ iff effort requirement $effor(f, r, h) > 0$ is allocated to position i.

$xdz_{f,r,h} = 1$ iff effort requirement $effor(f, r, h) > 0$ cannot be filled by any open position at hour h.

$xtoteff_{i,h} \geq 0$: Total effort allocated to position i at hour h.

$0 \leq xovertime_{i,h} \leq overcap(i, h)$: Overtime allocated to position i during hour h

$xover_{i,h} \geq 0$: Over-allocation of capacity at position i an hour h. That is, more capacity than the average utilization was allocated.

$xunder_{i,h} \geq 0$: Under-allocation of capacity at position i an hour h. That is, less capacity than the average utilization was allocated.

$0 \leq xd_{f,r,h} \leq effor(f, r, h)$: Artificial variable to allow that the effort requirement of replica r of flight leg f is not fulfilled at hour h.

$zh_{i,f,r}$ = 1 if position i is allocated to replica r of flight leg f at any hour of planning horizon, and 0 otherwise.

$zhr_{i,f}$ = 1 if position i is allocated to flight leg f at any hour of planning horizon, and 0 otherwise.

$up_{i,f}$ = 1 if position i is allocated to leg f at all hours of the planning period for all identical replicas r, and 0 otherwise.

$zr_{i,f,h}$ =1 if position i fills the effort requirements of flight f at hour h for all identical replicas r.

$z1_{i,f,r} = 1$ indicates that only position i has been allocated throughout the whole life of the flight f at replica r.

WKLD MIP Formulation

0) Objective function: Minimize penalties reflecting violating the various goals of the WKLD problem.

$$Min\ \ penalties = evencst * \sum_{(i,h)\in ih}\left(xover_{i,h} + xunder_{i,h}\right) +$$

$$splitcst * \sum_{(i,f,r)\in ifr}\ (\sum_{h\in ifrh} effor(f,r,h) * z_{i,f,r,h} - z1_{i,f,r} * \sum_{h\in frh} effor(f,r,h)) +$$

$$bigm * \sum_{(f,r,h)\in frh} xdz_{f,r,h}$$

1) Satisfy demand

Satisfy effort requirements using bin variables z. Notice that the equation will set a variable z = 1 to indicate that one position fills the effort (f, r, h), and if not possible an artificial variable for unfilled requirements will be set to one. For each effort requirement $effor(f,r,h) > 0$ allocate a position i to fulfill the effort requirement. For each $(f,r,h) \in frh$

$$\sum_{i\in ifrh} z_{i,f,r,h} = 1 - xdz_{f,r,h}$$

2) Total effort requirement at position

This equation computes total effort as a function of bin variables z. For each position i and open hour such that total capacity > 0, compute total effort requirement that is fulfilled. For each $(i,h) \in ih$

$$\sum_{(f,r)\in ifrh} effor(f,r,h) * z_{i,f,r,h} = xtoteff_{i,h}$$

3) Satisfy position capacity

For each position i and open hour such that $regcap(i,h) > 0$, satisfy position capacity constraints. For each $(i,h) \in ih$

$$xtoteff_{i,h} \le regcap(i,h) + xovertime_{i,h}$$

4) Overtime constraints

For each $(i,h) \in ih$

$$xovertime_{i,h} \le overcap_{i,h}$$

5) Deviation of total effort respect to regular capacity

For each position i and open hour h such that $(i,h) \in ih$, compute over or under deviation of position utilization respect to average utilization.

$$xover_{i,h} - xunder_{i,h} = {xtoteff_{i,h}}\Big/{totcap(i,h)} - \left({1}\Big/{numpos(h)}\right)\sum_{j\in ih} {xtoteff_{j,h}}\Big/{totcap(j,h)}$$

6) Indicates if position has been assigned to a replica of a flight leg

This constraint for zh = 1 indicates whether i is allocated to f-r or not. For each replica r of a flight leg f, indicate if position i has been allocated to fulfill effort requirements at some hour during planning horizon. For each $(i, f, r) \in ifr$

$$\sum_{h \in ifrh} effor(f, r, h) * z_{i,f,r,h} \leq zh_{i,f,r} * \sum_{h \in frh} effor(f, r, h)$$

7) Maximum number of positions allocated per replica of flight leg

For each replica r of flight leg f, at most assign *maxpos(f)* positions to it. Then, for each (f, r) such that replica(f, r) = 1 satisfy

$$\sum_{i \in ifr} zh_{i,f,r} \leq maxpos(f).$$

8) Planning period cannot be split

For each replicate r of flight leg f and effort hour h of the planning period, ensure all planning period is fulfilled by <u>same</u> position i. For each $(i, f, r) \in ifr$

$$\sum_{h \in ifrh} plan(f, r, h) * effor(f, r, h) * z_{i,f,r,h} = u_{i,f,r} * \sum_{h \in frh} plan(f, r, h) * effor(f, r, h)$$

9) Only ONE position is considered during planning period

We need to add a constraint that ensures that only ONE position is allocated during the planning period of f-r. Then for each $(f, r) \in fr$ satisfy

$$\sum_{i \in ifr} u_{i,f,r} \leq 1$$

10) indicate that only 1 position has been allocated to flight leg

For each position i and replica r of flight f indicate if only 1 position has been used at all hours of the planning and following periods. For each $(i, f, r) \in ifr$

$$\sum_{h \in ifh} effor(f, r, h) * z_{i,f,r,h} \geq z1_{i,f,r} * \sum_{h \in frh} effor(f, r, h)$$

11) Consider switching penalties from one position to another

This constraint flags whenever a flight leg f at replica r switches from position i at hour h-1 to position j at hour h. For each $(i, j, f, r, h-1, h) \in ijfrh1h$

$$z_{i,f,r,h-1} + z_{j,f,r,h} \leq 1 + v_{i,j,f,r,h}$$

12) At most 2 switches can happen per flight-leg during following period

For each $(f, r) \in fr$ satisfy

$$\sum_{i,j,h \in ijfrh} v_{i,j,f,r,h} \leq 2$$

13) At most 1 switch from position i to j during following period of f-r

For each $(i, j, f, r) \in ijfr$ satisfy

$$\sum_{h \in ijfrh} v_{i,j,f,r,h} \leq 1$$

14) If i switch to j then i CANNOT be allocated again to f-r

For each $(i, j, f, r, h) \in ijfrh$ satisfy

$$\sum_{hh \geq h} z_{i,f,r,hh} \leq Dur_{f,r} * (1 - v_{i,j,f,r,h})$$

15) If i is NOT allocated to f-r at h-1 then we cannot switch from i to j at h

The idea of this constraint is that if pos i has not been allocated to f-r at hour h-1, then there cannot be a switching from i to j at hour h. For each $(i, j, f, r, h) \in ijfrh \cap ifrh$ satisfy

$$z_{i,f,r,h-1} \geq v_{i,j,f,r,h}$$

16) Ensure flight f-r is completely filled or not

This equation ensures that a flight f-r is either completely filled or not filled at all. For each $(f, r) \in fr$ satisfy

$$\sum_{i,h \in ifrh} z_{i,f,r,h} = Dur_{f,r} * y_{f,r}$$

17) Force y = 0 whenever xdz = 1

If flight f-r cannot be filled at some period h then the complete flight is not filled at all. For each $(f, r, h) \in frh$ satisfy

$$1 - xdz_{f,r,h} \geq y_{f,r}$$

Appendix B: Detailed description of WKLD Lab prototype

In this appendix, we provide a detailed description of the Lab prototype to tackle the WKLD problem.

The flow diagram of the WKLD Lab prototype and a description of the main components of the code follow.

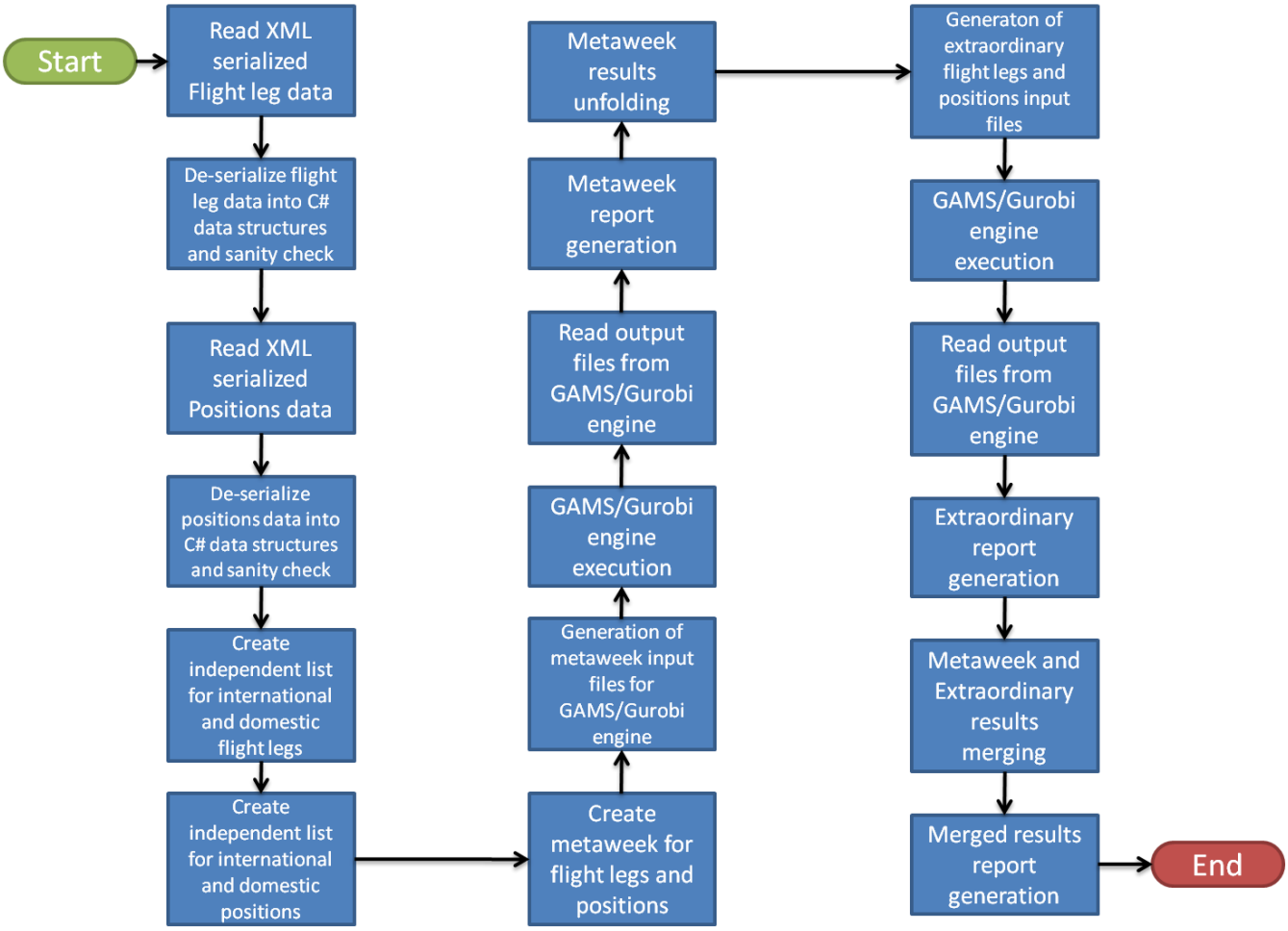

Figure 1: Flow Diagram of WKLD Lab prototype.

Read XML serialized Flight leg data and De-serialize flight leg data into C# data structures and sanity check

In these steps the XML files containing the flight legs information is de-serialized into the proper data structures. The functions to use are:

a) `public void LoadXMLFiles()`
b) `protected void DeserializeFlight(File f)`
c) `protected List<WLDFlightLeg> SanityCheck(List<WLDFlightLeg> flights)`

Read XML serialized Positions data and De-serialize positions data into C# data structures and sanity check

In these steps the XML files containing the positions information is de-serialized into the proper data structures. The functions to use are:

a) `public void LoadXMLFile(File f)`
b) `protected void DeserializePosition(File f)`

Create independent list for international and domestic flight legs and Create independent list for international and domestic positions

In these steps the data structures are classified in two groups INTERNATIONAL or DOMESTIC. The functions used are:

a) `public void SeparateInternational_Domestic()`

Create metaweek for flight legs and positions and Generation of metaweek input files for GAMS/Gurobi engine

In this step the data structures are prepared to create the metaweek, fill the data structures with the metaweek information and create the input files to start solving the problem. The functions to be used are:

1. **International**
    a. `public void PrepareDataToSolveInternational()`
    b. `public void CreateInputFiles()`
        i. `this.CreateParameterFiles(this.flight)`
        ii. `this.CreatePositionFiles()`
        iii. `this.CreateInputStatisticsFiles(this.flight)`
        iv. `this.CalculateEffortDistribution(this.flight)`
        v. `this.ReduceFlightsToPattern()`
        vi. `this.CreateParameterFiles(this.flightPattern)`
        vii. `this.CreateFlightFiles(this.flightPattern)`
        viii. `this.CreatePositionFiles()`
        ix. `this.CreateInputStatisticsFiles(this.flightPattern)`
        x. `this.CreateQualIFRH(this.flightPattern)`
        xi. `this.CreateDISCSPAN(this.flightPattern)`
        xii. `this.CreateCOMBPOS()`
        xiii. `this.CreateIJFRH1H()`
        xiv. `this.CreateIJJIFRH()`
2. **Domestic**
    a. `public void PrepareDataToSolveDomestic()`

b. `public void CreateInputFiles()`
    i. `this.CreateParameterFiles(this.flight)`
    ii. `this.CreatePositionFiles()`
    iii. `this.CreateInputStatisticsFiles(this.flight)`
    iv. `this.CalculateEffortDistribution(this.flight)`
    v. `this.ReduceFlightsToPattern()`
    vi. `this.CreateParameterFiles(this.flightPattern)`
    vii. `this.CreateFlightFiles(this.flightPattern)`
    viii. `this.CreatePositionFiles()`
    ix. `this.CreateInputStatisticsFiles(this.flightPattern)`
    x. `this.CreateQualIFRH(this.flightPattern)`
    xi. `this.CreateDISCSPAN(this.flightPattern)`
    xii. `this.CreateCOMBPOS()`
    xiii. `this.CreateIJFRH1H()`
    xiv. `this.CreateIJJIFRH()`

GAMS/Gurobi engine execution

In this step you will be able to use your preferred method; in this case we are using a C# Process object to run a .BAT file containing the MS-DOS instructions needed to execute GAMS.

```
Process gurobi = new Process();
gurobi.StartInfo.FileName = converter.DirectoryPath + "wkldop-04k.bat";
gurobi.StartInfo.WorkingDirectory = converter.DirectoryPath;
gurobi.StartInfo.UseShellExecute = false;
gurobi.StartInfo.RedirectStandardInput = false;
gurobi.Start();
gurobi.WaitForExit();
gurobi.Close();
```

Metaweek report generation and Metaweek results unfolding

In this step, the output files generated by GAMS/Gurobi are read and the output reports are generated. In order to achieve this, you need to execute the following functions.

1. International or Domestic
    a. `public void ReadPatternOutputFiles()`
        i. `this.ReadVarOverTime()`
        ii. `this.ReadVarTotEff()`
        iii. `this.ReadVarX()`
        iv. `this.ReadVarXD()`
        v. `this.ReadVarZH()`
        vi. `this.ReadVarZHR()`
        vii. `this.CreateEffortTestPerHour() [Optional]`
        viii. `this.CalculateEvennessMetric()`
        ix. `this.CalculatePositionFreqPerLeg()`
        x. `this.CalculateConsistencyMetric()`
        xi. `this.CheckFlighLegContinuity(this.flightPattern)`
        xii. `this.CreateOuputStatistics()`

xiii. `this.CalculateDifferentPosPerLeg(this.flightPattern)`
xiv. `this.UncollapseVarX()`
xv. `this.UncollapseVarXD()`
xvi. `this.CreateEffortMatrix4Surface()`
xvii. `this.CreateEvenessMatrix4Surface()`
xviii. `this.IdentifyGaps()`

Generaton of extraordinary flight legs and positions input files

In this step the positions capacity are updated given the results obtained in the metaweek processing, also the input files for the extraordinary flights and positions are generated. The functions needed are:

1. **International or Domestic**
    a. `public void CreateOFFPatternInputFlights()`
        i. `public void CreateOFFPatternInputFlights()`
        ii. `this.UpdatePositionCapacity()`
        iii. `this.CreateParameterFiles(this.flightNotPattern)`
        iv. `this.CreateFlightFiles(this.flightNotPattern)`
        v. `this.CreateInputStatisticsFiles(this.flightNotPattern)`
        vi. `this.CreateQualIFRH(this.flightNotPattern)`
        vii. `this.CreateDISCSPAN(this.flightNotPattern)`
        viii. `this.CreateCOMBPOS()`
        ix. `this.CreateIJFRH1H()`
        x. `this.CreateIJJIFRH()`
        xi. `this.CreatePositionFiles()`

GAMS/Gurobi engine execution

In this step you will be able to use your preferred method; in this case we are using a C# Process object to run a .BAT file containing the MS-DOS instructions needed to execute GAMS.

```
Process gurobi = new Process();
gurobi.StartInfo.FileName = converter.DirectoryPath + "wkldop-04k-
offptrn.bat";
gurobi.StartInfo.WorkingDirectory = converter.DirectoryPath;
gurobi.StartInfo.UseShellExecute = false;
gurobi.StartInfo.RedirectStandardInput = false;
gurobi.Start();
gurobi.WaitForExit();
gurobi.Close();
```

Read output files from GAMS/Gurobi engine - Extraordinary report generation - Metaweek and Extraordinary results merging - Merged results report generation

In these steps the off_pattern files are read and the extraordinary output reports are generated. As a second stage the metaweek results are merged with the extraordinary results and the proper output reports are generated

1. International or Domestic
    a. `public void ReadOFFPatternOutputFiles(bool kind, string path)`
        i. `this.ReadVarOverTime()`
        ii. `this.ReadVarTotEff()`
        iii. `this.ReadVarX()`
        iv. `this.ReadVarXD()`
        v. `this.ReadVarZH()`
        vi. `this.ReadVarZHR()`
        vii. `this.CalculateEvennessMetric()`
        viii. `this.CalculatePositionFreqPerLeg()`
        ix. `this.CalculateConsistencyMetric()`
        x. `this.CheckFlighLegContinuity(this.flightNotPattern)`
        xi. `this.CreateOuputStatistics()`
        xii. `this.CalculateDifferentPosPerLeg(this.flightNotPattern)`
        xiii. `this.CreateEffortMatrix4Surface()`
        xiv. `this.CreateEvenessMatrix4Surface()`
        xv. `this.IdentifyGaps()`
        xvi. `this.MergeSolutions(kind,path)`

Appendix C: Special data structures in MIP formulation

In this appendix we describe an issue of the WKLD model related to the allocation discontinuities of positions to a flight leg f-r. Consider the following allocation.

Position	Flight	Day	Hour
i1	**f1**	**d**	**h1**
i1	**f1**	**d**	**h2**
i1	**f1**	**d**	**h3**
i2	**f1**	**d**	**h4**
i3	**f1**	**d**	**h5**
i2	**f1**	**d**	**h6**
i3	**f1**	**d**	**h7**
i3	**f1**	**d**	**h8**

Figure C1: Flight leg f-d discontinuous allocation

The planning period of f1-d is h1…h3, and the following period is h4…h8.

This type of allocation might happen due to the minimization of evenness deviations, although the question is if this type of allocation is feasible.

If this allocation is infeasible, then we need to develop a set of constraints that removes this type of discontinuous allocation, but at the same time they allow continuous allocations.

We now propose a set of data structures that will facilitate the implementation of the set of constraints to avoid allocation discontinuities

For each flight leg f-r we need to define the span of possible discontinuity hours throughout the life of the flight leg. Notice that there is a constraint that guarantees that only ONE position can be allocated during the planning period; hence the span of discontinuity hours of flight f-r is determined by the last hour of the planning period and all the hours during the following period –except the last hour of the following period. Let's define DISCSPAN (f, r, h) as the set of hours in the discontinuity span of flight f-r.

We use the example in Figure C1 to illustrate our concepts. Hence, DISCSPAN set will look like this

f1, d, h3
f1, d, h4
f1, d, h5
f1, d, h6
f1, d, h7

Note that the number of records per flight f1-d is the number of following hours and in this example is 5 records

We also need to generate all the permutations of two open positions for flight f-d. Let's define COMBPOS (i, j, f, r) as the set of permutations of feasible positions for f-r. In terms of the example in Figure 1, the set COMBPOS looks like follows, assuming the set of feasible (open) positions for f1-d is {i1, i2, i3}:

i1, i2, f1, d
i1, i3, f1, d
i2, i1, f1, d
...
i3, i1, f1, d
i3, i2, f1, d

Let n be the number of feasible positions for f1-d, then the number of records for f1-d in the set COMBPOS is n*(n-1).

Let's consider now the set of following of hours of flight leg f-r called FOLLOWSET (f, r, h), in terms of our example this set looks as follows:

f1, d, h4
f1, d, h5
f1, d, h6
f1, d, h7
f1, d, h8

Now we generate another data structure we require for our set of constraints that eliminates discontinue allocations. This data structure includes the DISCSPAN (f, r, h) FOLLOWSET (f, r, h) and COMBPOS (i, j, f, r) sets, and the idea of this set is to enumerate the possibility of switching from one position i at hour $h-1 \in DISCSPAN$ to another position j at hour $h \in FOLLOWSET$.

The name of this set is IJFRH1H (i, j, f, r, h1, h)*;* the fields of this set (table) are position1-id, position2-id, flight-id, replica-id, hour1-id, hour2-id, where $(pos1, pos2) \in COMBPOS$, and $hour1 \in DISCSPAN \; and \; hour2 \in FOLLOWSET$. In terms of the example for flight f1-d it looks as follows:

COMBPOS POS	COMBPOS POS	Flight	Replica	DISCSPAN	FOLLOWSET
i1	i2	f1	d	h3	h4
i1	i2	f1	d	h4	h5
i1	i2	f1	d	h5	h6
i1	i2	f1	d	h6	h7
i1	i2	f1	d	h7	h8
i1	i3	f1	d	h3	h4
i1	i3	f1	d	h4	h5
i1	i3	f1	d	h5	h6
i1	i3	f1	d	h6	h7
i1	i3	f1	d	h7	h8
i2	i1	f1	d	h3	h4
i2	i1	f1	d	h4	h5
i2	i1	f1	d	h5	h6
i2	i1	f1	d	h6	h7
i2	i1	f1	d	h7	h8
i2	i3	f1	d	h3	h4
i2	i3	f1	d	h4	h5
i2	i3	f1	d	h5	h6
i2	i3	f1	d	h6	h7
i2	i3	f1	d	h7	h8
i3	i1	f1	d	h3	h4
i3	i1	f1	d	h4	h5
i3	i1	f1	d	h5	h6
i3	i1	f1	d	h6	h7
i3	i1	f1	d	h7	h8
i3	i2	f1	d	h3	h4
i3	i2	f1	d	h4	h5
i3	i2	f1	d	h5	h6
i3	i2	f1	d	h6	h7
i3	i2	f1	d	h7	h8

Content of file *ijfrh1h.txt*

Now we generate a data structure called IJFRH (i, j, f, r, h) derived from the set IJFRH1H (i, j, f, r, h1, h) by removing the field hour1-id. Also we generate a set IJFR (i, j, f, r) derived from the set IJFRH1H (i, j, f, r, h1, h) by removing the field hour1-id and the filed hour2-id. Note that IJFR is identical to COMBPOS.