On Large-Scale Airline Crew Pairing Generation

Divyam Aggarwal
Mechanical & Industrial Engineering Department
Indian Institute of Technology Roorkee
Roorkee, Uttarakhand- 247667, India
daggarwal@me.iitr.ac.in

Dhish Kumar Saxena
Mechanical & Industrial Engineering Department
Indian Institute of Technology Roorkee
Roorkee, Uttarakhand- 247667, India
dhishfme@iitr.ac.in

Michael Emmerich
Leiden Institute of Advanced Computer Science
Leiden University
Niels Bohrweg 1, 2333 CA Leiden, The Netherlands
m.t.m.emmerich@liacs.leidenuniv.nl

Saaju Paulose
Digital Solutions
General Electric (GE) Aviation
Fort Worth. Texas, USA
saaju.paulose@ge.com

***Abstract*—Crew operating cost is the second largest cost component of an airlines' total operating cost (second only to the fuel cost) and even marginal cost savings here may amount to millions of dollars, annually. Towards it, a crew needs to be efficiently assigned a sequence of flights starting and ending at the same crew base (a crew pairing). The challenge for an airline is to generate crew pairings which completely cover a finite set of flights over a particular time window, with minimum cost, while satisfying multiple *legality* constraints linked to airlines' own regulations, labor laws, and government safety rules etc. The success in solving the associated constrained optimization problem largely depends on solving NP-complete subproblems, linked to the generation of a feasible solution (legal crew pairings covering all flights) and generation of pairings with reasonable cost-quality. In an attempt to address these subproblems in a computationally- and time-efficient manner, the contributions of this paper relate to the use and characterization of network structure for pairing generation; enhancement of the graph traversal algorithm, namely Depth-first Search (DFS); its parallel implementation on multiple processors of a single computer; and embedding of cost considerations during pairing generation itself towards boosting the optimizer's performance subsequently. The utility of the cited contributions is demonstrated on medium (∼2000 flights) and large (∼4000 flights) data sets provided by GE Aviation.**

***Index Terms*—Airline Scheduling, Crew Scheduling, Crew Pairing, Crew Pairing Generation, Parallelization, Depth-first Search.**

I. Introduction

Airline crew scheduling is one of the most important step of the overall airline scheduling process. This relates to the fact that crew operating cost is the second largest cost component of the total operating cost, next to the fuel cost. Unlike fuel cost, a major part of the crew costs is controllable and can be minimized by optimizing the crew scheduling process. Traditionally, crew scheduling process is carried out by solving two subproblems sequentially, namely, crew pairing problem, and crew rostering problem. The former deals with the generation of a set of crew pairings (a crew pairing is a sequence of flights to be flown by a crew that starts and ends at the same crew base) to cover a finite number of flights of an airline in minimum cost possible, and the latter deals with the assignment of these pairings to specific crew members. The focus of this paper is on the crew pairing problem and the readers are referred to [1] for a comprehensive discussion on crew scheduling process. Crew pairing problem is an extremely complex combinatorial optimization problem and belongs to the class of NP-hard problems. Since 1980s, airliners and researchers from all over the globe have given a considerable amount of attention towards this problem because of the two reasons: a small percentage of cost improvement translates to millions of dollars annually, and presence of the suboptimal state-of-the-practices. However, in the last two decades, the number of flight legs and airports to be scheduled have increased tremendously, leaving the state-of-the-practice obsolete. With the recent technological advancements, data handling capacities and the speed of computations have improved significantly, leaving the scope for further improvement in the current state-of-the-practices in crew pairing problem.

In a crew pairing problem, crew pairings have to satisfy multiple legality constraints linked to FAA rules and regulations, labor agreements, government safety regulations etc. These legality constraints, to some extent, have been discussed in section III-A in detail. Owing to the presence of these legality rules, it is required to develop a *legal crew pairing generation approach* in order to facilitate *legal crew pairings* to the optimization problem. The presence of a hub-and-spoke type of flight network (used in this research) causes an explosion in the flight connections at hubs, leading to billions/trillions of legal pairings. This brings tractability issues in the pairing generation problem, making it a computationally expensive and time-consuming process. The exact amount of legal pairings possible for a large-scale problem (say 4000 flight legs)

This research is partly supported by MEITY in India (under the grant MIT-861-MID) & NWO in Netherlands, and partly by General Electric (GE) Aviation

is not known and may take non-deterministic polynomial time to complete, making it an NP problem. Given the NP-nature of the large-scale pairing generation problem, the generation of an initial feasible solution (covering all flights atleast once) and the generation of pairings with reasonable cost-quality are NP-complete problems as it is hard to find the associated solutions but easy to verify in polynomial time. After developing a legal pairing generation process, the legal pairings are facilitated to the optimization problem which is sometimes modeled as a set-partition problem where each flight is covered only once, and sometimes as a set-covering problem where flights are allowed to be covered more than once. There are two types of optimization approaches: one approach is to produce all legal pairings 'offline' (before proceeding with optimization phase) and selecting the best subset from these pre-generated pairings, and the other approach is to produce the pairings dynamically during the optimization phase. The scope of this paper is limited to the crew pairing generation process, and due to the presence of enormous literature associated with these approaches, readers are referred to the most recent ones, [2] & [3], for each of these approaches respectively.

Mostly, in crew pairing literature, pairing generation has been performed using either row approach, column approach, or network approach. Some literature studies that used row approach for pairing generation are [4], [5] and [6]. There are two types of network-based approaches for pairing generation- one uses a flight-based network and the other uses a duty-based network. The flight-based network approach is used in [7] and [8]. Reference [9] was the first approach to use a duty-based network for pairing generation. The other approaches linked to duty-based network, are [10], [11], [12], [3]. Researchers in [13] presented experiments with both the network structures, however, they used the duty-based network for a problem of upto 449 flights and used the flight-based network for a problem of upto 1744 flights. In [14], a new approach, called as Knowledge-Based Random Algorithm (KBRA), is presented to generate pairings. This approach randomly selects flight legs to start with and then uses the knowledge of candidate flights of the predecessor flight in order to generate pairings. A typical solutioning when implemented on mainframe computers, consumes days of computing in order to construct acceptable crew schedules and most of this computing time (25-50%) is consumed in pairing generation process (e.g. [6], [11]). Hence, it is imperative to speed up the pairing generation process in order to improve the crew scheduling time. With the technological advancements such as the introduction of parallel computing architectures, researchers have started developing parallel pairing generation approaches. In [15], authors have presented a master/worker parallel pairing generation algorithm, where a master processor is used to distribute new flights to idle worker processors. In [16], authors have developed a new load balancing scheme to distribute the pairing generation process among different machines of a cluster and among multiple processors within the same machine without using a master processor. This parallel pairing generation algorithm is employed in [17] and [18]. With these parallel approaches, an expensive computational environment is required which is not in the interest of our sponsors. Moreover, the efforts required to implement such complex concepts in a crew scheduling framework may prevent the researchers from testing and developing new ideas, serving as a serious obstacle-to-entry in this line of research. Hence, instead of a complicated parallel environment, a straightforward parallelization is proposed in this paper which may motivate the new researchers to develop and test new ideas in a simple, yet faster, pairing generation environment.

The major contributions of this paper are:

- a parallel implementation of the pairing generation approach by decomposing it into independent subprocesses w.r.t. each crew base and distributing them on multiple processors of a single computer;
- a comparative study of the pairing generation process using two network structures, highlighting their uses and characteristics; and
- two modifications in the pairing generation approach- an enhancement of the graph traversal algorithm (DFS) to generate a feasible solution for warm-starting[1], and embedding of cost considerations to generate pairings with reasonable cost-quality.

Section II outlines the basic terminology associated with the crew pairing problem, a comprehensive discussion about the network structures used for pairing generation, and the concepts and pseudo-codes of the developed pairing generation approach. Section III presents the used test-cases and the computational experiments for the developed pairing generation approach. Section IV concludes this paper by summarizing the use and characterization of the developed pairing generation approach.

II. Crew Pairing Generation

A. Basic Terminology

In this section, the basic terminology of crew pairing problem is discussed. In crew pairing problem, a flight schedule along with its fleet-type is given as input. This flight schedule contains all the non-stop flight legs and their associated attributes such as flights' departure (arrival) time-stamp[2], departure (arrival) airport, block-time, fleet-type. *Block-time* of a flight leg is its actual flying time and is measured as the time an aircraft takes from the departure gate to the arrival gate of the airport. In a work day, the sequence of flight legs flown by a crew member is called a *duty period* or a *duty* in short. In a duty, two flight legs are always separated by a time interval, called *connection-time* or *sit-time*, which is restricted by a maximum and minimum limit. The sit-time accounts for the time taken by a crew in changing the aircraft

[1]It means starting the optimization framework using a feasible solution, covering all flights, instead of a random solution which may or may not cover all flights.

[2]It gives information about the time and day of the event.

for its next flight, if required, or a short break. A rest period also called an *overnight rest*, is provided at the end of a duty. This overnight rest is always longer than the sit-time and is restricted by a minimum and maximum limit. A *crew base* or a *crew domicile* is the home airport of the crew. A *crew pairing* is the sequence of flights that starts and ends at the same crew base. Sometimes, due to flight cancellations or missing flight connections, the crew needs to be transported to an airport where they are needed to fly the next scheduled flight, or sometimes they may end up at an airport which is not their crew base and needs to be brought back to their crew base to end their pairing. The crew can be transported between airports in the same city using road transportation but if not possible, then the crew is repositioned by traveling as passengers in a flight and such flights are called *deadhead flights* or *deadheads* in short. Deadheading affects the total revenue of the airline in two-ways, one by reducing the profits earned from the passenger seat being wasted, and the other by paying the crew for the time they did not fly any plane. Hence, it is desirable for airlines to reduce the number of deadheads as much as possible in order to increase their total revenue. The time for which a crew is away from its base station is called as *time away from base* (TAFB). An example of a crew pairing with five flights and two duties is given in Fig. 1.

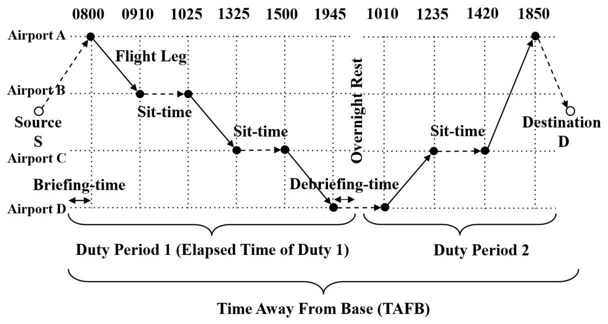

Fig. 1. An example of a legal crew pairing using flight-based network

B. Network Structure

Conventionally, in the crew scheduling literature, two types of network structure are used to carry out crew pairing generation. The first network design is called a flight-based network (FN) and the other is called a duty-based network (DN).

1) *Flight-based Network*: In this network, flights are the fundamental units of the network and can be represented in two ways, either using a node or an arc. Traditionally, an arc-based representation is used for simplifying the network construction. In this representation of this network, there are two distinct nodes corresponding to each flight leg, a departure node, and an arrival node. These nodes are connected using an arc representing the flight leg. There are two other nodes present in the network, one is a source node (shown by S) and the other is a destination node (shown by D). The departure node of each flight leg that departs from a particular crew base, is connected to the S node and the arrival node of each flight leg that ends at the same crew base is connected to the D node. For a legal flight connection in a pairing, the arrival airport of the first flight leg should be same as the departure airport of the second flight leg and the time interval between both the legs should be within the minimum and maximum limits of either the sit-time in a duty period or the overnight rest between the duties. A connection-arc is used to create a legal flight connection in the network. As mentioned in [13], it is to be noted that the overnight rest is a function of multiple attributes of the preceding duty and the whole pairing. Hence, an overnight connection-arc is only inserted if the overnight rest is $\geq$ and $\leq$ the minimum and maximum bounds of the permissible layover time respectively. This overnight connection-arc is an approximation (not a tight bound) and it may or may not be legal depending on the other pairing rules. Fig. 1 shows a sample flight-based network.

2) *Duty-based Network:* In a duty-based network, a duty period can also be represented in two ways, either by a node or an arc. Conventionally, in a duty-based network, departure time and station, and arrival time and station are represented by nodes and the arc between them represents a duty-period. Connection-arcs are inserted between those nodes which satisfy the conditions for a legal overnight rest i.e. arrival station of the last flight of the first duty should be same as the departure station of the first flight of the second duty and the time interval between two duties should be within the permissible bound of the overnight rest. It is known that the overnight rest is a function of the various attributes of the preceding duty and the overall pairing. In comparison to the flight-based network, duty rules are explicitly built in the duty-based network and hence the latter is superior to the former in terms of estimating a real overnight connection. Fig. 2 shows a sample duty-based network.

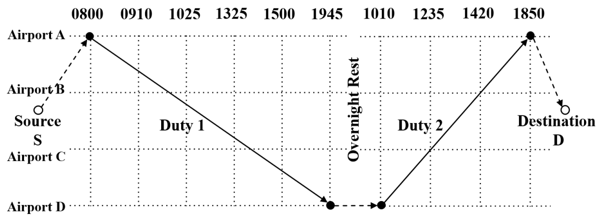

Fig. 2. An example of a legal crew pairing using duty-based network

A legal crew pairing is nothing but some S-D path in both networks. The flight-based network guarantees the flight connection within a duty period but it does not promise the legality of other pairing rules such as maximum duty flying hours and many more. Hence, some of the S-D paths in the flight-based network may not be legal and the other pairing rules are required to be checked dynamically during pairing generation. On the contrary, all duty rules and a majority of the pairing rules are satisfied by the S-D paths in the duty-based network, hence, overcoming the limitations of the flight-based network to a certain extent. This improvement comes at a cost of high memory-usage due to the presence of a much larger arcs set. However, some of the rules such as 8-in-24 rule[3], are not addressed even in the duty-based network. Authors in [13] suggested that such rules may never be enforced using these network structures unless a multi-labeling procedure or a constrained shortest path method [20] is used in order to keep track of the costing and legality rules, hence, leading to only legal paths/pairings.

In a duty-based network, a duty period can also be represented with a node and a legal overnight rest with a connection-arc between two duty nodes. Reference [16] adopted this type of representation for generating crew pairings from a duty-based network structure. First, all of the legal duties are enumerated and then the duty-based network is built using these legal duties from which legal pairings are generated. Reference [21] defined a new approach, called a *duty tree* approach, to generate all legal duties and to store them in a memory-efficient way. A compact storage thread is defined by them in order to store all of the duties, that starts from the same root node, in a single thread (less memory). In a duty tree structure, all the flights are represented using a node and the connections are included between those nodes that satisfy the condition of a legal sit-time connection within a duty period. There are multiple properties associated with this duty tree approach, such as the height of the duty tree, completeness, exclusiveness, tail-off and many more. Using the same type of structure, authors in [21] also built a *pairing tree* structure, that contains these duties as nodes and overnight rest period as the connection between these nodes, to generate legal pairings. Readers are referred to the work in [21] for acquiring more comprehensive information about the duty tree and pairing tree structures.

C. Crew Pairing Generation Approach and Enhancements

In this research, pairing generation approaches using both the network structures is developed.

For the flight-based network structure, a tree-like representation (flights as nodes and legal connections as arcs with no cycle present) is used to generate legal pairings. Several attributes of the flight legs are necessary in order to evaluate a pairing's legality and to calculate its cost. A class object is used to store a flight leg and its associated attributes as class labels. There are two types of search algorithms for traversing a network, namely, *depth-first search* (DFS) and *breadth-first search* (BFS). The former traverses along each branch of the tree whereas the latter searches along the breadth of the tree and the choice of selection depends upon the characteristics of the network formed. In the above-mentioned representation of the flight-based network, a tree structure is formed and a legal pairing can only be generated by traversing along the branch (depth) of this tree. Hence, the DFS algorithm is used for traversing this network. In DFS, starting from a root node, all the nodes along each branch (depth) of the tree are traversed before backtracking[4] to the previous node and traversing the next branch of that node. The pseudo code of a DFS algorithm, adapted for searching a pairing tree, is shown in Algorithm 1.

Let C_S denotes sit-time constraints, C_R denotes overnight rest constraints, C_D denotes all duty constraints, C_P denotes all pairing constraints, and S_{CB} denotes the set of all crew bases. A root node in a flight-based network (or duty-based network) is a flight leg (or duty period) that starts from any airport in S_{CB} respectively. Let R_a and DR_a be the set of all flight nodes and duty nodes that departs from an airport a respectively. The pseudo code of the sequential pairing generation algorithm using the flight-based network is shown in Algorithm 2. First, a flight-based network is constructed (lines 1-10), and then all legal pairings starting from all root nodes and for all crew bases are sequentially generated (line 12-25) using the DFS algorithm (line 19).

Algorithm 1: Depth-first Search Algorithm

```
   Input: ParentFlight, PF; Flight Graph, G; PairingStack; and pairing
          constraints, C
1  Output: Set of legal pairings, $S_{LP}$
2  $S_{LP} = \phi$
3  for Child Flight of $PF \in G$ do
4      Add $CF$ to Stack    /* CF is Child Flight of PF  */
5
6      if —($PF \rightarrow CF$) satisfy all constraints $\in C$— then
7          if —PairingStack $\equiv$ Complete Sequence— then
8              Add PairingStack to $S_{LP}$
9              Remove $CF$ from PairingStack
10             Backtrack to a previous flight in the PairingStack
11         else
12             DFS($CF$, G, PairingStack, C)
13         end
14     else
15         Remove $CF$ from PairingStack
16         Backtrack to a previous flight in the PairingStack
17     end
18 end
```

Similarly, for the duty-based network, a duty tree and a pairing tree structure, similar to the ones in [21], is used to generate legal duties and pairings, and the remaining pairing rules are dynamically checked. A DFS algorithm is used to traverse this network to search for legal pairings. First, all legal duties are enumerated using a flight-based network. This is performed using the same method as given in Algorithm 2 and the only difference here is the use of duty rules instead of all pairing rules as done in Algorithm 2. All the generated legal duties are stored in a set, S_{LCD}, which is then used to construct a duty-based network with a tree-like representation (duties as nodes and overnight rest as connection-arcs; no cycles present) along with the overnight rest constraints as connection-arcs. The pseudo code of this procedure is given in lines 1-10 of Algorithm 3. The pseudo code for traversing this duty-based network using DFS is given in lines 12-36 of Algorithm 3. The legality of the pairings for

[3]In this rule, the crew is not permitted to fly more than 8 hrs in a 24-hour window unless an extra rest is provided in the end.

[4]It means going back to the parent node as soon as the child node is inspected and further traversal is not possible either because of an invalid solution or the end of the branch has reached.

Algorithm 2: Sequential pairing generation using DFS in flight-based network

```
/* Generation of flight-based network */
Input: Flight schedule, $F$; $C_S$; and $C_R$
Output: Flight-based Network, $FN$
for $f \in F$ do
    for $f' \in F \backslash f$ do
        if —($f \rightarrow f'$) satisfy constraints $\in (C_S \cup C_R)$— then
            Add a directed edge from $f$ to $f'$
        else
            continue
        end
    end
end
/* Generation of all legal pairings using DFS */
Input: $S_{CB}$; $R_a$; $FN$; and set of other pairing constraints,
       $C' \equiv (C_P \cup C_D) \backslash (C_S \cup C_R)$
Output: All legal crew pairings, $S_{LCP}$
$S_{LCP} = \phi$
for $a \in S_{CB}$ do
    for $f \in R_a$ do
        PairingStack $= \phi$
        if —$f$ satisfy pairing-start constraints— then
            Add $f$ to PairingStack
            DFS($f$, $FN$, PairingStack, $C'$)
            Add legal pairings to $S_{LCP}$
        else
            continue
        end
    end
end
```

Algorithm 3: Sequential pairing generation using DFS in duty-based network

```
/* Generation of duty-based network */
Input: $S_{LCD}$; and $C_R$
Output: Duty-based network, $DN$
for $d \in S_{LCD}$ do
    for $d' \in S_{LCD} \backslash d$ do
        if —($d \rightarrow d'$) satisfy constraints in $C_R$— then
            Add a directed edge from d to d'
        else
            continue
        end
    end
end
/* Generation of all legal pairings using DFS */
Input: $S_{LCD}$; $DR_a$; $DN$; and set of other pairing rules
       $C' \equiv C_P \backslash (C_D \cup C_S \cup C_R)$
Output: All legal crew pairings, $S_{LCP}$
$S_{LCP} = \phi$
for $a \in S_{CB}$ do
    for $d \in DR_a$ do
        PairingStack $= \phi$
        if —$d$ satisfy pairing-start constraints— then
            Add $d$ to PairingStack
            DFS(d, DN, PairingStack, $C'$)
            Add legal pairings to $S_{LCP}$
        else
            continue
        end
    end
end
```

the remaining pairing rules is examined dynamically inside the DFS algorithm (line 6 & 7 of Algorithm 1).

This case study is carried out on a real-world large-scale problem (containing up to 4212 flights). Out of this problem, two test-cases are developed and are discussed in section III-A in detail. The number of possible legal duties for TC-2 is estimated to be 737,184 and the number of possible legal pairings is expected to be more than a billion (exact amount not known); leading to tractability issues. Towards this effect, some serious challenges arise such as how to speed up the pairing generation process, how to create a full coverage initial feasible solution, and how to generate only quality pairings from such a huge search-space? To address these challenges, following three modifications in the pairing generation approach are proposed:

Mod. 1: Parallel Pairing Generation

With the technological advancements, the computational hardware is now capable of multiprocessing (using more than one CPU in parallel). From the understanding of pairing rules and network structure, it is inferred that the overall pairing generation process can be decomposed into independent subprocesses on the basis of crew base they are starting from. In this case study, 15 crew bases are present and some of the pairing rules are a function of the crew base from which the pairing starts. Hence, the overall pairing generation process could be decomposed into independent subprocesses on the basis of each crew base. Using this decomposition and the multiprocessing capability of the computational machine used, the legal duty/pairing generation process is sped up by running 15 independent processes in parallel. Each process returns a set of legal crew duties/pairings starting from a particular crew base. The pseudo code for parallelization of the sequential algorithms (Algorithms 2& 3) is shown in Algorithm 4.

Algorithm 4: Parallel Algorithm

```
Input: $S_{CB}$; Independent subprocesses, $SubP \equiv \{SubP_a \forall a \in S_{CB}\}$
Output: Combined Set, $CS$
$CS = \phi$
for $a \in S_{CB}$ in parallel do
    Send $SubP_a$ to a randomly chosen idle thread
    Add resultant sets from each $SubP_a$ to $CS$
end
```

Mod. 2: Variable-Backtrack Modification in DFS

An offline[5] enumeration of all legal pairings for TC-2 (4212 flight problem, given in section III-A) is not advisable as it may not terminate in a finite time. Hence, it is not possible to generate a feasible solution (a set of pairings that covers all the flights atleast once) in a short time using the original-DFS. But sometimes, this feasible solution is required to warm-start the optimization process. To generate a feasible solution in a short time, original-DFS is enhanced by varying the step-length of backtrack so as to cover more number of unique flight nodes. To explain the working of the enhanced-DFS, an example is presented in fig. 3. Let us

[5]It means generating legal pairings before proceeding with the optimization process.

assume that the *PairingStack* has a legal pairing sequence: $1 \rightarrow 2 \rightarrow 3 \rightarrow 5 \rightarrow 8$ before backtracking and it is not possible to traverse further as the branch end is reached. Now,

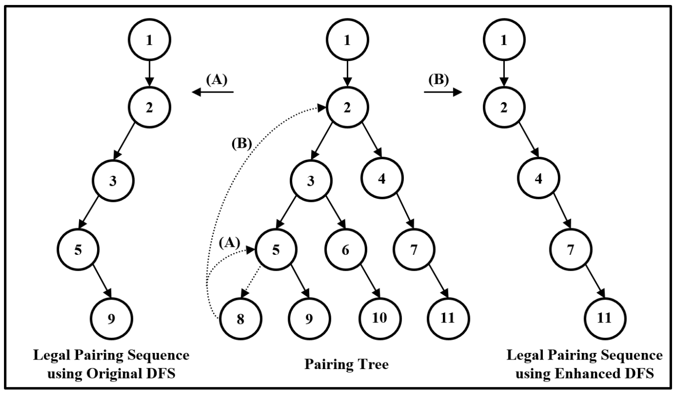
(A)
(B)
(B)
(A)
Legal Pairing Sequence using Original DFS
Pairing Tree
Legal Pairing Sequence using Enhanced DFS

Fig. 3. Enhanced-DFS Algorithm

with a backtrack step-length $S_{BT} = 2$, the enhanced-DFS (B) backtracks to the flight node 2 instead of flight node 5 which happens in case of original-DFS (A), as shown in fig. 3. After backtracking and further traversal, the next legal pairing sequences obtained in both the DFSs' are as follows:

- Original-DFS (right pairing): $1 \rightarrow 2 \rightarrow 3 \rightarrow 5 \rightarrow 9$
- Enhanced-DFS (left pairing): $1 \rightarrow 2 \rightarrow 4 \rightarrow 7 \rightarrow 11$

On comparing these legal pairings with the previous legal pairing, it is clear that new legal pairing generated using enhanced-DFS covers more unique nodes than the new legal pairing generated using original-DFS. Note that the backtrack step-length is measured from the start of the *Stack* i.e. if $S_{BT} = 2$ then the DFS will backtrack to the node 2 and will explore its child branches. In addition to this, the pairing generation process is repeated by varying the backtrack step-length S_{BT} from 0 to a finite limit (say 5) until all unique flights are covered atleast once.

Mod. 3: Cost constrained Pairing Generation
To promote an efficient search in a huge search-space, it is imperative to reduce it to a manageable size. For search-space reduction, it is required to identify a pairing's feature that could help in quantifying its quality. From the understanding of the costing rules, it is found that the excess-pay cost component (explained in section III-A) is the most important KPI [6]. It is the cost of a crew's non-productive hours (other than the flying, hotel and meal costs) and is required to be as minimum as possible (ideally 0). In this module, legal pairings being generated are filtered using a constraint on excess-pay during the DFS search (after line 7 in Algorithm 1). Pairings, satisfying excess-pay constraint, are added to the final set of legal pairings. The constraint upper bound could also be varied in order to ensure the presence of pairings with various cost quality.

[6]Key Performance Indicator.

III. Computational Experiments

A. Airline Data

In this paper, all computations are demonstrated using a real-world airline crew pairing problem (defined by GE Aviation). From the given weekly flight schedule, two test-cases are developed- a medium-scale test-case (TC-1, 1820 flights) and a large-scale test-case (TC-2, 4212 flights), and this client's operations involve 15 crew bases. In the provided flight schedule, each flight leg is associated with following attributes: departure/arrival date and time, departure/arrival city, aircraft fleet type, and flight block-time. These test-cases involve several legality constraints which must be satisfied by a crew pairing to be 'Operational' or 'Legal', and some of these are as follows:

- *Start-city and End-city Constraints*: 1^{st} flight of a pairing should start from a crew-base only and last flight of the pairing should end at the same crew-base.
- *Connection-city Constraint*: For connection between flights, departure city of an outgoing flight should be same as arrival city of the incoming flight.
- *Sit-time & Rest-time Constraints*: Sit-time between two consecutive flights in a crew duty and the rest-time between two consecutive crew duties in a pairing should be restricted by minimum and maximum limits.
- *Duty Constraints*: Number of duties in a pairing is restricted by a maximum limit. Also, number of flights in a duty, duty elapsed-time, and duty flying-time should be restricted by minimum and maximum limits, and these limits vary according to the duty start-time.
- *Special Constraints*: Some special rules such as pairing starting from a crew-base cannot overnight in the same-city airports etc., are used to optimize the crew utilization.

A complicated set of costing rules is defined by GE Aviation to calculate a pairing's cost. These costing rules are as follows:

- *Excess-Pay*: It is the pay of minimum guaranteed hours minus the cost of actual flying hours. It is the cost of non-productive hours of a crew in a pairing. The guaranteed pay of the pairing is calculated as the maximum of various individual guarantees which depends on duty elapsed-time, TAFB, deadhead hours, and so forth. In order to optimize the crew-utilization, it is imperative to minimize excess-pay of a set of pairings (ideally $= 0$).
- *Soft Cost*: It is the penalty cost incurred on changing an aircraft within a duty of a pairing, and on dead-heading a flight within a pairing.
- *Hotel and Meal Costs*: It constitutes the hotel costs incurred in the overnight rests and the entire trip's meal cost.

B. Results of Crew Pairing Generation Methods

Results of computational experiments are demonstrated in this section. Implementation of all pairing generation modules is performed in Python 3.6, and an HP Z640 Workstation, having 32 cores @3.2 GHz (capable of multiprocessing) is used for all computations. First, a comparison is drawn in between pairing generation approaches using flight-based and duty-based network structures, and their respective sequential and parallel algorithms (SA & PA). These computational results are summarized in Table I. The prime aim of crew pairing generation problem is to generate a large number of legal pairings in minimum runtime possible, resulting in two sole criteria for drawing the comparison. First criterion is the runtime of pairing generation (for TC-1 and similar small/medium-scale problems) and the other is the number of pairings generated in a fixed time (for TC-2 and similar large-scale problems where $\geq$ billion legal pairings are possible). For TC-1, it is observed that 1,681,056 legal pairings are possible and among its flight-based network algorithms, PA is $\sim$5 times faster than SA whereas among its duty-based network algorithms, PA is $\sim$8 times faster than SA. Also, for TC-1, the duty-network based PA is $\sim$2 times faster than the flight-network based PA. For TC-2, all algorithms are allowed to run for an hour (3600 sec) and it is observed that among its flight-based network algorithms, PA generated $\sim$9 times the pairings generated by SA whereas among its duty-based network algorithms, PA generated $\sim$19 times the pairings generated by SA in the given time. Also, for TC-2, the duty-network based PA generated $\sim$29 times the pairings generated by the flight-network based PA. From these results, it is evident that PA is faster than SA by an average factor of $\sim$10 and among the algorithms based on both the network structures, the duty-network based algorithms are extremely faster than the ones based on flight-network. As discussed in section II-C, for generating pairings using a duty-based network, it is imperative to pre-process all legal duties. These duties are generated using a flight-based network and the results of duty generation using SA & PA are summarized in Table II. It is observed that the parallel enumeration of all legal duties takes <60 seconds which is negligible in comparison to the number of duty rules already pre-processed, resulting in a faster search for legal pairings in duty-network based algorithms. However, in duty-based network, the number of duty nodes (737,184 for TC-2) present is much more than the number of flight nodes (4212 for TC-2) involved in flight-based network, and the entire duty-network has to be stored in memory. To this effect, duty-based network has high-memory requirements but these storage needs could be justified by the fact that memory requirement per pairing is significantly smaller for the duty-based network than the flight-based network.

TABLE I
LEGAL PAIRING GENERATION USING USING SEQUENTIAL ALGORITHM (SA) AND PARALLEL ALGORITHM (PA) FOR TC-1 & TC-2

Test Case	Measures	Flight-based Network		Duty-based Network	
		SA	PA	SA	PA
TC-1	Amount	1,681,056	1,681,056	1,681,056	1,681,056
	Runtime (sec)	2101	436	1629	199
TC-2	Amount	378,572	3,573,492	5,485,255	103,183,858
	Runtime (sec)	3600	3600	3600	3600

TABLE II
LEGAL DUTY GENERATION USING SEQUENTIAL ALGORITHM (SA) AND PARALLEL ALGORITHM (PA) FOR TC-1 & TC-2

Measures	TC-1		TC-2	
	SA	PA	SA	PA
Amount	331,455	331,455	737,184	737,184
Runtime (sec)	78	14	237	56

The crew pairing optimization problem for TC-2 and similar large-scale problems is initiated using an initial feasible solution. Due to presence of large number of possible legal pairings in such test-cases, it is not advisable to generate all legal pairings, making it difficult to find a feasible solution in short runtime. In this paper, an enhanced-DFS algorithm (Mod.2 in section II-C) is presented for generation of a feasible solution in very short runtime and its comparison with the original-DFS is shown in Table III, using a sequential algorithm built on flight-based network. It is observed that the enhanced-DFS covers all 4212 flights (of TC-2) in a runtime of $\sim$150 seconds whereas the original-DFS only covered 980 flights in 150 seconds (results shown under Column "Setting 1" in Table III). Moreover, the original-DFS is allowed to run for an hour (3600 seconds) and still it covered only 1565 flights (results shown under Column "Setting 2" in Table III). This proves that enhanced-DFS is highly effective in generating an initial feasible solution for TC-2 and similar large-scale problems, which is imperative to start the crew pairing optimization problem. Fig. 4 shows the variation of unique flight coverage with the variation in backtrack step-length of the enhanced-DFS.

TABLE III
COMPARISON BETWEEN ENHANCED-DFS AND ORIGINAL-DFS ALGORITHMS TO GENERATE A FEASIBLE SOLUTION FOR TC-2

Measures	Enhanced-DFS	Original-DFS	
		Setting 1	Setting 2
Amount	77,728	664	1,872
Runtime (sec)	146	146	3600
Uncovered Flights	0	3232	2647

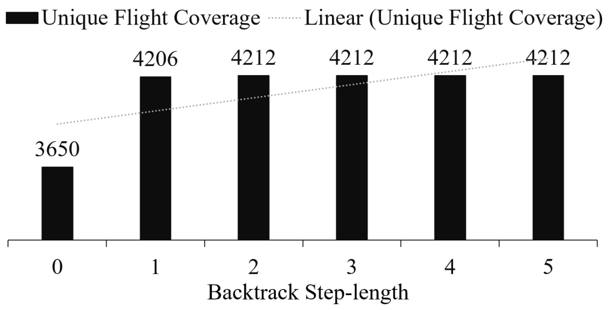

Fig. 4. Unique Flight coverage versus Backtrack Step-length for TC-2

Given the large number of possible legal pairings in TC-2, the search efficiency of the subsequent crew pairing optimization problem becomes low. In this paper, Modification 3 of pairing generation method is presented in section II-C which helps in reducing the large search-space to pairings with reasonable cost quality. Under this modification, cost constraints are embedded in the pairing generation and only those pairings are generated which have '0' excess pay (excluding the deadhead excess pay component; ideally excess pay = 0). Results of this module are shown in Table IV. With this modification, it is possible to narrow down TC-2's huge search-space (billions/trillions; not known) to a quality pairing set of size ∼612 million (still huge but manageable).

TABLE IV
RESULTS OF MODIFICATION-3 PARALLEL PAIRING GENERATION ALGORITHM USING DUTY-BASED NETWORK FOR TC-2

Experiment	Measures	Duty-based Network
Legal Duty Generation	Amount	633,485
	Runtime (sec)	37
	Uncovered Flights	5
Legal Pairing Generation	Amount	611,274,334
	Runtime (sec)	31,230
	Uncovered Flights	5

IV. Conclusion and Future Scope

In this paper, a computationally- and time-efficient legal crew pairing generation approach is presented for a real-world airline crew pairing problem (with ≤4212 flight nodes). Firstly, a parallel pairing generation approach is presented by distributing the pairing generation with-respect-to each crew base (15 in total) among 15 idle threads of a processor. From computational experiments, it is observed that the parallel approach is ∼10 times faster than the sequential approach. However, it should be equivalent to the number of crew bases (i.e. 15) ideally. The inefficiencies of Python's *Multiprocessing* library, used for parallelization, is accountable for this less-than-expected speed improvement. Secondly, the uses and characteristics of pairing generation approach for both network structures are explored and to guarantee a fair competition, exactly same experimental settings are used. It is observed that, computationally, the duty-network based algorithms are faster than the flight-network based algorithms. Finally, it is proved that Modification-2 (Enhanced-DFS) & Modification-3 (Cost-constrained pairing generation) are highly effective in solving the associated NP-complete subproblems (generating an initial feasible solution and generating pairings with reasonable cost-quality) which builds the rationale for solving the subsequent NP-hard crew pairing optimization problem.

The parallel approach, presented in this paper, could be improved further by involving the concepts of load-balancing among the multiple threads used. However, these advanced concepts of parallelization may increase the complexity of the pairing generation environment, which may prevent researchers from testing and developing new ideas for the subsequent crew pairing optimization problem.

Acknowledgment

Arioli Arumugam (Director- Data & Analytics, GE Aviation), and Alla Rajesh (Staff Data Scientist, GE Aviation), thanks for problem definition and discussions on the crew pairing generation problem.

References

[1] C. Barnhart, A. Cohn, E. Johnson, D. Klabjan, G. Nemhauser, and P. Vance, "Airline crew scheduling", In: Handbook of transportation science, Kluwers International Series, Dordrecht: Kluwer Academic Publishers, 2003.
[2] B. Zeren, and İ. Özkol, "An Improved Genetic Algorithm for Crew Pairing Optimization", Journal of Intelligent Learning Systems and Applications, Vol. 4 No. 1, pp. 70-80, 2012.
[3] B. Zeren, and I. zkol, "A novel column generation strategy for large scale airline crew pairing problems", Expert Systems with Applications, 55:133-44, 15 Aug 2016.
[4] M.M. Etschmaier, and D.F. Mathaisel, "Airline scheduling: An overview", Transportation Science, 19(2):127-38, May 1985.
[5] R. Anbil, E. Gelman, B. Patty, and R. Tanga, "Recent advances in crew-pairing optimization at American Airlines", Interfaces, 21(1):62-74, Feb 1991.
[6] G.W. Graves, R.D. McBride, I. Gershkoff, D. Anderson, and D. Mahidhara, "Flight crew scheduling", Management science, 39(6):736-45, Jun 1993.
[7] C. Barnhart, E.L. Johnson, R. Anbil, and L. Hatay, "A column-generation technique for the long-haul crew-assignment problem", InOptimization in industry 2, (pp. 7-24). John Wiley & Sons, Inc., 4 Oct 1994.
[8] G. Desaulniers, J. Desrosiers, Y. Dumas, S. Marc, B. Rioux, M.M. Solomon, and F. Soumis, "Crew pairing at air france", European journal of operational research, 97(2):245-59, 1 Mar 1997.
[9] S. Lavoie, M. Minoux, and E. Odier, "A new approach for crew pairing problems by column generation with an application to air transportation", European Journal of Operational Research, 35(1):45-58, 1 April 1988.
[10] J. Desrosiers, Y. Dumas, M. Desrochers, F. Soumis, B. Sanso, and P. Trudeau, "A breakthrough in airline crew scheduling", Cahiers du GERAD, G-91-11, 1991.
[11] H.D. Chu, E. Gelman, and E.L. Johnson, "Solving large scale crew scheduling problems", InInterfaces in Computer Science and Operations Research, (pp. 183-194), Springer, Boston, MA, 1997.
[12] R. Anbil, J.J. Forrest, and W.R. Pulleyblank, "Column generation and the airline crew pairing problem", Documenta Mathematica, 3(1):677, Aug 1998.
[13] P.H. Vance, A. Atamturk, C. Barnhart, E. Gelman, E. L. Johnson, A. Krishna, D. Mahidhara, G.L. Nemhauser, and R. Rebello, "A heuristic branch-and-price approach for the airline crew pairing problem", Technical report TLI/LEC-97-06, Georgia, Institute of Technology, Atlanta, GA, 23 June 1997.
[14] A. Aydemir-Karadag, B. Dengiz, and A. Bolat, "Crew pairing optimization based on hybrid approaches", Computers & Industrial Engineering, 65(1):87-96, 1 May 2013.
[15] C. Goumopoulos, E. Housos, and O. Liljenzin, "Parallel crew scheduling on workstation networks using PVM", InEuropean Parallel Virtual Machine/Message Passing Interface Users Group Meeting, (pp. 470-477), Springer, Berlin, Heidelberg, 3 Nov 1997.
[16] D. Klabjan, and K. Schwan, "Airline Crew Pairing Generation in Parallel," InPPSC, 2001.
[17] D. Klabjan, E.L. Johnson, G.L. Nemhauser, E. Gelman, and S. Ramaswamy, "Solving large airline crew scheduling problems: Random pairing generation and strong branching", Computational Optimization and Applications, 20(1):73-91, 1 Oct 2001.
[18] A.J. Schaefer, E.L. Johnson, A.J. Kleywegt, and G.L. Nemhauser, "Airline crew scheduling under uncertainty", Transportation science, 39(3):340-8, Aug 2005.
[19] G. Eason, B. Noble, and I. N. Sneddon, "On certain integrals of Lipschitz-Hankel type involving products of Bessel functions," Phil. Trans. Roy. Soc. London, vol. A247, pp. 529–551, April 1955.
[20] M. Desrochers, and F. Soumis, "A generalized permanent labelling algorithm for the shortest path problem with time windows," INFOR: Information Systems and Operational Research, 26(3):191-212, 1 Jan 1988.
[21] S. Qiu, "Airline crew pairing optimization problems and capacitated vehicle routing problems," PhD diss., Georgia Institute of Technology, 2012.