ALGORITHM OPTIMIZATION FOR SOLVING CREW SCHEDULING PROBLEMS

Author: Christine A. Oketch
Director: Dr. Ángel A Juan
The Institute of International Graduate in Computers, multimedia and telecommunications, Universitat Oberta de Catalunya, 2013

ABSTRACT

Airline industry is one of the most competitive industries existing today and both efficient and effective crew schedule is crucial as it has a big impact on the airline's costing. Crew Scheduling Problem involves the process of assigning crew to operate a designated route. Lots of literature exist in regards to try to solve the Crew Scheduling Problems.

In this paper, we are proposing a methodology to determine the most efficient and least costly way of crew pairing optimization. We are developing a methodology based on algorithm optimization on Eclipse open-source IDE using the Java programming language to solve the crew scheduling problems.

The solution derived from this algorithm has been tested and the results show that the execution time obtained by the algorithm is quite fast and it can be relied on to produce the best set balanced crew routes.

Key Words: Crew Scheduling, Crew Pairing, algorithm optimization, balanced crew routes.

1. INTRODUCTION

Airlines main objective is cost cutting on its day to day operations. Several research and studies have been done on airlines cost structure and the findings have been that, after the cost of fuel which cannot be controlled by the airline, the second highest expense is the crew scheduling cost. Crew cost basically covers the salaries, layover stipends when out of the base, hotel, transport etc. These costs are probably the most important area for potential airlines savings and when not planned properly, it can cost the airline and therefore becoming a huge problem. With this in mind, airlines are striving to reduce or save on the costs by implementing efficient and effective solutions to solve the crew scheduling problems.

Crew Scheduling problems can be defined as those factors that affects the crew operations. The factors can be flight cancellations due to weather, equipment unavailability, flight delays, political factor such as union politics, crew unavailability such as sickness. It is divided into Crew Pairing and Crew Rostering.

Crew pairing is a sequence of flight legs or segments that begin and end at a crew base such that in a sequence the arrival city of a flight leg coincides with the departure city of the next flight leg and wrong or incomplete crew pairing can be very costly to an airline, Deng, G. F., & Lin, W. T. (2010).
Crew pairing problems is divided into two categories; cock-pit crew pairing and cabin crew pairing. Each category has its own rules, as explained in Shangyao Yan, Jei-Chi Chang (2002), cock-pit crew pairing generation is more complicated than cabin crew because each pilot is qualified to fly only one type of aircraft. Crew Pairing problem is usually approached by first generating a huge number of pairings. From this huge collection of legal pairings, a set of pairings that has minimal cost and which ensures that each flight is manned by exactly one crew is determined.

Once crew pairs have been generated, each single crew is assigned to the pair generated, this is called Crew Rostering Problem. while assigning the crew to flight pairs, several factors have to be taken into consideration depending on airlines, that is, number of days off per week, number of hours per week, number of nights per week in the case of layover flights, age restriction on operating some legs in the case of pilots, crew dead heading etc.

The purpose of this paper is to propose an algorithm which automatically controls for all lurking variables, thereby getting feasible and optimal solutions which can be useful in solving the aforementioned crew scheduling problems.

The rest of the document is organized as follows: Section 2 reviews some of existing research done by other authors in regards to crew scheduling problems. Section 3 describes the analysis of the proposed algorithm. Section 4 reports the computational experiments for the proposed models. Finally, in Section 5, we conclude and outline possible research extensions.

2. BACKGROUND AND RELATED WORK

Airline operations can be very costly especially if not managed and controlled properly. The airline operations includes flight operations which consists of flight planning, air craft route planning, crew scheduling, et cetera. In this research we are going to focus on crew scheduling and crew scheduling problem.

There are many articles researched on and written in regards to solving crew scheduling problems and in our case, we are applying an algorithm optimization to solve the crew scheduling problems.

One of the crew scheduling problems which we are seeking to solve is the Crew pairing issues. Crew cost is the biggest expenditure an airline can control, therefore effective assignment of crews to flight is a very important aspect of airline planning(see Gopalakrishnan et al.(2005)). Wrong or incomplete crew pairing can be very costly to an airline. In Deng et al. (2010), Ant Colony Optimization-based algorithm that was proposed by Dorigo in 1992, has been applied to solve the crew scheduling problems. The ACO idea was inspired by the behavior of ant colonies that find the shortest route between ant's nest and a source of food, and this was applied in crew scheduling problems by applying a flight based scheduling to build the shortest path with minimum cost and use flights as nodes of paths and the connecting edges to conform to the constraints between two consecutive flights

In Shangyao Yan et al. (2002), a column generation approach model was applied to help in minimize crew cost and to plan for proper cockpit crew pairing. Unlike cabin crew pairing where a crew can be qualified in more than 2 aircraft types, cockpit crew can only be qualified to fly one type of aircraft, thus making the generation of the cockpit crew scheduling more complicated. Using the column generation model in Ahuja et al.,(1993), the flight timetable and the work rules stated in the article are used to develop two scheduling networks in order to generate feasibly pairings. The networks are called Standard crew network and Augmented crew network. Based on the model we are trying to see if we can apply it on our research to solve the crew pairing problems

For day to day flight operations, there are factors that disrupt the smooth flow of operations, these can be weather, flight cancellation, delays etc and this results to crew scheduling disruptions which are costly. Schaefer, Andrew J., et al. (2005), applied a deterministic crew scheduling model operating under uncertainty under the assumption that all pairings will operate as planned, this model is the SimAir, a Monte Carlo Simulation of airline operations based on Rosenberger et al.(2000a).

The column generation approach has also been mentioned in AhmadBeygi, S. et al. (2009), where an Integer programming approach is applied to generate airline crew pairings

In Ralf Borndöfer et al. (2005), a column generation approach is applied for solving airline crew scheduling problems that is based on a set partitioning model, considering algorithmic aspects such as the use of bundle techniques for the fast, approximate solution of linear programs, a pairing generator that combines Lagrangean shortest path and callback techniques, and a novel “rapid branching” IP heuristic

A stochastic crew scheduling model and a solution methodology for integrating disruptions in the evaluation of crew schedules was devised in Yen, J. W., & Birge, J. R. (2006). The goal was to use the information to find a robust solution that can withstand disruptions. A stochastic integer programming model was used to develop a branching algorithm to identify expensive flight connections and finding alternative sources.

Algorithms similar to the SR-GCWS-CS that combines Monte Carlo simulation as explained in AA Juan et al.(2011), can be used to solve combinatorial type problems like the Crew Scheduling Problems whereby some biased random behaviour within the CWS heuristic to perform a search process inside the space of feasible solutions are introduced. These feasible solutions consists of a set of round trip routes from the depot (in our case, the airport base) that, altogether, satisfy all demands of the nodes by visiting and serving all of them exactly once. Different geometric statistical distributions during the randomized CWS solution-construction process: every time a new edge is selected from the list of available edges, a value α is randomly selected from a uniform distribution. This value is then used to assign exponentially diminishing probabilities to each eligible edge according to its position inside the sorted savings list. That way, edges with higher savings values are always more likely to be selected from the list, but the exact probabilities assigned are variable and they depend upon the concrete distribution selected at each step.

In Vance et al. (1997), a new formulation and decomposition approach for solving a new model based on selecting a set of duty periods that cover the flights in schedule and building pairings using he duty periods is suggested. The formulation is based on linear programming relaxation which provides a stronger bound in the optimal integer programming. Just like the algorithm we are trying to solve in our case, in this model, to initialize the column generation procedure, there must be a feasible solution to the LP relaxation of the master problem. An initial solution can be obtained by constructing an artificial solution using the duty periods in any duty period set to build pairings. The other approach would be to start with some known feasible solution.

Crew scheduling problems can also be solved using Differential Evolution (DE) method as discussed in Santosa, B. et al. (2010). In the paper, the DE algorithm is proven to be able to find the near optimal solution accurately for the optimization problem, focusing on developing differential evolution algorithm applied on intelligent airline crew rostering system. Differential evolution is an evolutionary population-based algorithm proposed by R. Storn and K. Price (1995). Still with evolution method, Marchiori, E. et al. (2000) proposes an adaptive heuristic based evolutionary algorithm whose main ingredient is a mechanism for selecting a small core sub problem which is dynamically updated during the execution. The mechanism allows the algorithm to find covers of good quality in short time.

A network model approach has been used in Yan, S., & Tu, Y. P. (2002) to efficiently and effectively solve crew scheduling problems for a Taiwan airline using real constraints. The network allows for the drafted flight timetable, and the average cost and number of cabin attendants required for each flight, to formulate crew scheduling for a single home base. The model was formulated as a pure network flow problem to solve the problem, and a flow decomposition algorithm was applied to obtain the pairings, from the optimal integer solutions. The model has a source node and a sink node that represents the same home base, the other nodes indicates the location and the time for a duty departure and arrival. The network model is also represented by arcs; specifically starting or ending arcs, duty arcs, deadhead arcs, rest arcs, and a cyclic arc

In Levine, D. (1996), Application of a hybrid genetic algorithm to airline crew scheduling has been developed and compared to the traditional approaches. The hybrid algorithm consists of a steady-state genetic algorithm. The algorithm works with a population of candidate solutions. As can be seen in the original Genetic Algorithm of Holland J(1975), each candidate solution is represented as a string of bits where the interpretation of the bit string is problem specific. The strings then recombines by using the crossover and mutation operators to produce a new generation of strings.

In Juan, A. A. et al. (2011), the use of probabilistic or randomized algorithm is used for solving vehicle routing problems with non-smooth objective functions. Just like the problem we are trying to solve, we are seeking to apply similar approach of biased randomization to solve crew scheduling problems. The approach employs a non-uniform probability distributions to add a biased random behavior to well known savings heuristic in order to sample out the best feasible solution.

3. THE METHODOLOGY RESOLUTION

This methodology is almost similar to the ones discussed in the literature. It involves creating an algorithm to balance crew routes.

The major actors in this methodology are

- A pair of Legs where; legs=(leg_1, leg_2, ...leg_n) and a leg being a connection between one airport to the airport, for example BCN-AGP-BCN

- Airports where; Aiports=(aiport$_1$, airport$_2$....airport$_n$)

When creating the algorithms solutions, we will take the following factors into considerations

- Numbers of flight sectors → There can be routes that have more than 4 sectors, however with proper route creation, they can be reduced to only 3 sectors. Lets take an example where we have crew route containing BCN-SVQ-BCN-MAD-BCN, this route has 5 sectors but it can be reduced to 4 sectors by doing a triangle route generation to get BCN-SVQ-MAD-BCN
- Duty time period → Pilots and cabin crew have different duty time period and when generating the crew routes, this has to be taken into consideration to avoid bursting out the required hours. Depending on airlines and Collective Bargain Agreements (CBA) each type of crew has a minimum number of hours that they can operate in a day, in some airlines the pilots can do up to 12 hours and cabin crew up to 15 hours

Starting with the first solution, we will generate the Basic routes of paired legs to obtain the leg pairings. To do this, we analyze the pseudo-code of the algorithm below

```
procedure generateBasicRoutesOfPairedLegs(legs)

% This procedure generates basic roundtrip routes
% composed of pairs of inverse legs.
% E.g.: (BCN-JFK) + (JFK-BCN) = BCN-JFK-BCN

01 basicRoutes <- emptyList
02 legs <- sortByStartingTime(legs)
03 while {legs is not empty} do
04   newRoute <- emptyRoute
05   leg <- extractNextLeg(legs)
06   invLeg <- extractInverseLeg(leg, legs)
07   newRoute <- merge(leg, invLeg)
08   basicRoutes <- add(newRoute)
09 end while
10 return basicRoutes

end procedure
```

The algorithm is trying to generate basic crew routes composed of number of legs

In line 1, we do not have any values in our list yet, thus, our *basicRoutes* list is empty.
As we can see in line 2, there are available legs, and they will be sorted by their starting time to get the *sortByStartingTime*. This start time is generated from the origin airport leg time and date and the arrival time and date of the destination airport.

In line 3, We will verify if the leg is not empty and in line 4, new empty route will be created, the next leg will be extracted in line 5, the inverse leg will be extracted in line 6 and in line 7, the next leg and the inverse leg will be merged to create the new route. This new route will then be added to the basic route on line 8 and the new basic route created will be added to the list of paired legs.

This analysis is represented in the flow chart diagram below (Figure I)

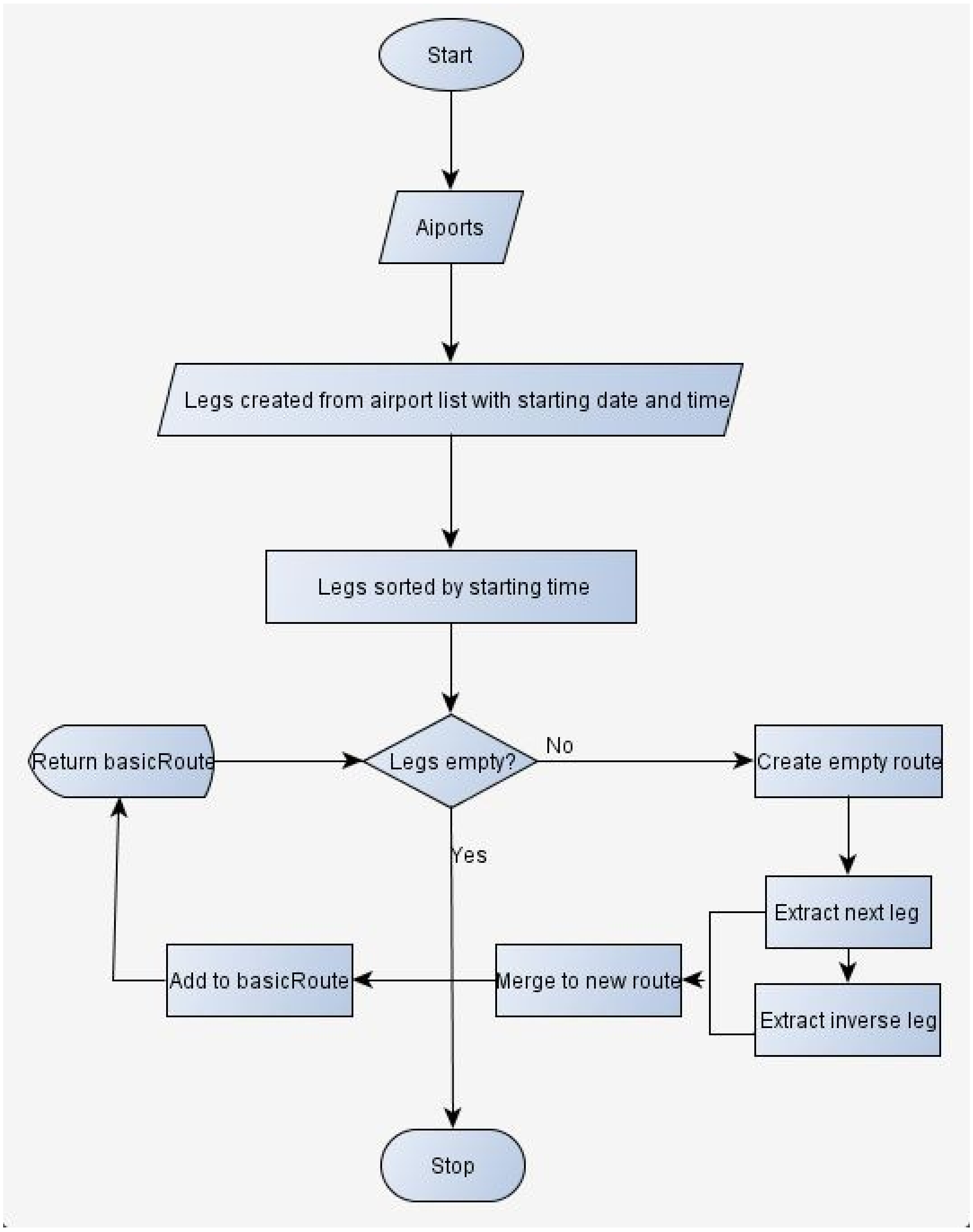


Figure I: Flow chart diagram for creating basic Route pair

Once the basic routes have been generated, we will create the initial solution by merging the basic Route solution above starting at each airport. The idea behind this would be to balance the crew routes and return a working and feasible crew route combinations.

The pseudo code for generating the initial solution is as shown below

```
procedure generateInitialSol(legs, airports)

% This procedure generates an inital solution by merging
% basic routes (pairs of legs) starting at each airport.
% E.g.: (BCN-MAD-BCN) + (BCN-VAL-BCN) = BCN-MAD-BCN-VAL-BCN

01 globalSol <- emptySol
02 basicRoutes <- generateBasicRoutesOfPairedLegs(legs)
03 for each {airport in airports} do
04   basicRoutes(airport) <- getBasicRoutes(basicRoutes, airport)
05   airportSol <- emptySol
06   while {basicRoutes(airport) is not empty} do
07     newRoute <- extractNextStartingRoute(basicRoutes(airport))
08     for each {route in basicRoutes(airport)} do
09       tentativeRoute <- merge(newRoute, route)
10       if {tentativeRoute is feasible} then
11         basicRoutes(airport) <- delete(route, basicRoutes(airport))
12         newRoute <- tentativeRoute
13       end if
14     end for
15     airportSol <- add(newRoute, airportSol)
16   end while
17   globalSol <- add(airportSol, globalSol)
18 end for
19 return globalSol

end procedure
```

The algorithm is trying to generate a feasible pairing solution from the legs and airports (basicRoute), this solution is going to be called **globalSol**.

To start with we do not have a globalSol yet, therefore at line 1 our global solution will be null.
We already have basic routes of paired legs generated from the previous algorithm, therefore in line 2 from the generated Basic routes of paired legs we get the **basicRoutes**. At line 3, we check in each airports an array of basicRoutes and aiports to generate the basicRoutes(Airports) in line 4.
There is still no solution generated yet the aiport, therefore at line 5, we have the aiportSol will be null.

At line 6, we will check if the basicRoute above is empty and if it is empty, we go back to getting the Basic routes again, otherwise we extract the next starting route (line 7) from the airport's basicRoute and create a new route called **newRoute**

For each route in the basicRoute in line 8, we will merge it with the newRoute to create a tentative route on line 9 called **tentativeRoute**.

We will then check the feasibility of the tentativeRoute line 10 and if feasible, we go ahead and delete the basicRoute (line 11) and make the tentativeRoute (line 12) as the **newRoute** and add the newRoute to the globalSol list and then the best feasible solution is returned.

The analysis can be summarized in the flow chart below (Figure II)

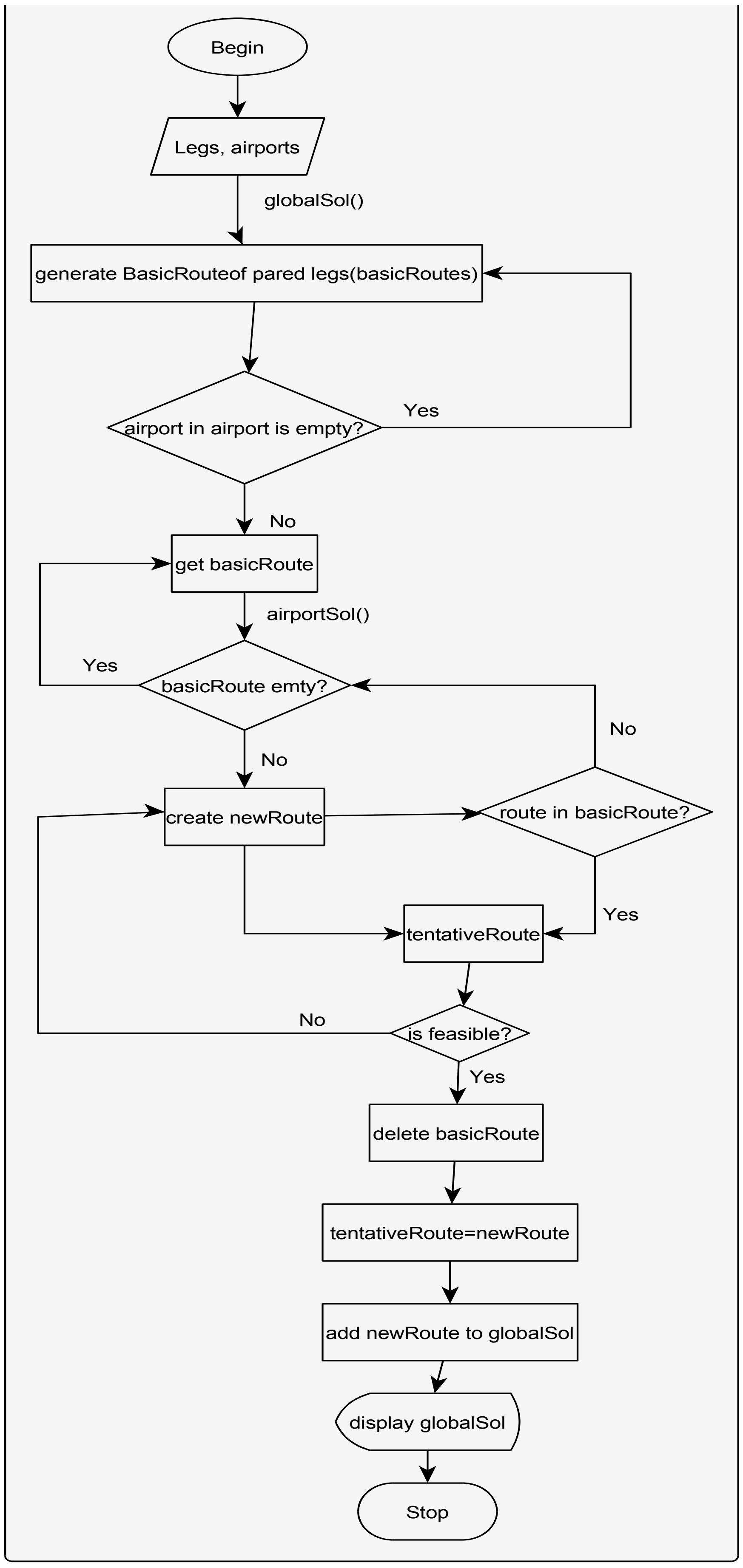


Figure II: Flow chart for the InitialSolution

4. RESULTS ACHIEVED FROM THE ALGORITHM.

The algorithm was developed by applying the Free Software on an Eclipse environment using java version 1.6.0_37 and was tested on a simulated schedule that can handle an average of 100 flights per day, both domestic and international. Assuming that the crew pairing generation is done manually, it can take lots of hours to find a feasible and optimized solution. Once the algorithm was developed successfully, we put it under different types of test to evaluate its efficiency and effectiveness.

The tests that we did were

- Performance and reliability test to check the approximate time it takes to execute and return the output using different types of operating system and processors and different numbers of flights per day
- Functional test to check the accuracy of the results returned, possible values that can be derived and exceptions

The Table1:Flight Schedules below has a list of possible scheduled flights in a day,

Flight number	Origin	Destination	Start time	Finish Time	Date
1015	MAD	AGP	1015	1115	02/01/2013
1016	AGP	MAD	1215	1315	02/01/2013
1017	MAD	AGP	1415	1515	02/01/2013
1018	AGP	MAD	1615	1715	02/01/2013
1111	SVQ	MAN	0900	1130	02/01/2013
1112	MAN	SVQ	1230	1500	02/01/2013
1115	AGP	CDG	0715	0945	02/01/2013
1116	CDG	AGP	1045	1315	02/01/2013
1201	SVQ	BCN	1000	1115	02/01/2013
1202	BCN	AMS	1215	1415	02/01/2013
1203	AMS	SVQ	1515	1815	02/01/2013
1800	MAD	LHR	0600	0800	02/01/2013
1801	LHR	MAD	0900	1100	02/01/2013
1802	AGP	FCO	0600	0800	02/01/2013
1803	FCO	AGP	0900	1100	02/01/2013
1804	AGP	LTN	0845	1045	02/01/2013
1805	LTN	AGP	1145	1345	02/01/2013
2000	MAD	JFK	1000	1835	02/01/2013
2001	JFK	MAD	0015	0850	03/01/2013
2960	AGP	CDG	0600	0715	03/01/2013
2961	ORY	CDG	0815	1015	03/01/2013
2962	CDG	ORY	1115	1215	03/01/2013

Table1:FlightSchedules

In the first algorithm of creating basic routes pairs, the algorithm check the possible pairs it can create using the flight number, origin and destination to have produce the output in the figure III below

Pair1: AGP =>CDG
Pair2: CDG => AGP

Figure III

In the second algorithm for generating initial solution, the algorithm uses the crew pairs generated in Figure III to find the best combination of crew routes and balance them off to produce the output in figure IV

AGP => CDG => AGP

Figure IV

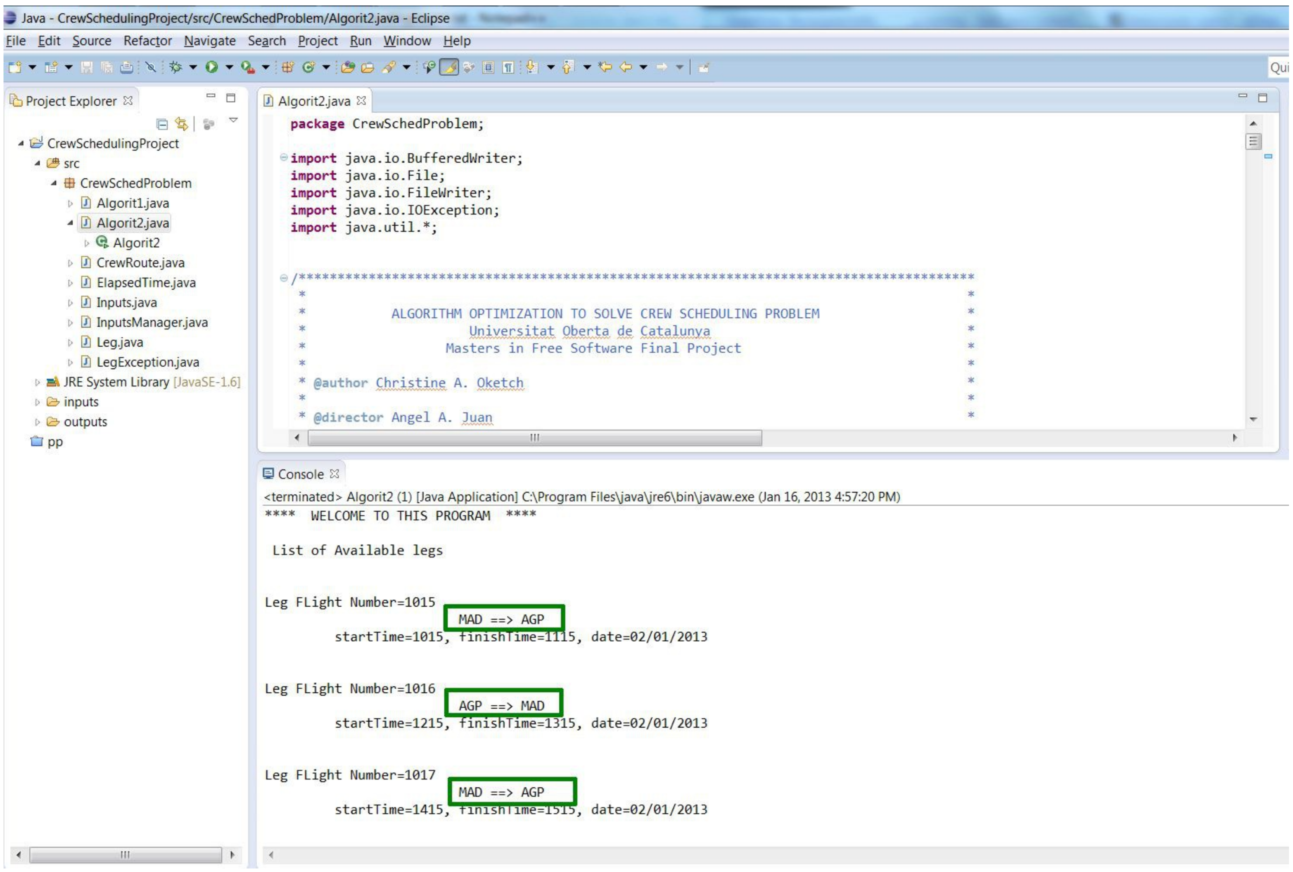
We executed the algorithm using different numbers of flights per day and the Table II below summarizes the execution time it took to provide the output in terms of number of flights per day

Table II: Execution time for a number of flights legs paired in a day

Equipment	50 flights	100 flights	200 flights
Windows 7 Intel(R) Core(TM)2 Duo CPU P8700 @2.53Hz 2.53GHz RAM: 4,00GB	Total elapsed time is 0h 0m 0s	Total elapsed time is 0h 0m 0s	Total elapsed time is 0h 0m 0s
openSUSE 12.2, AMD Athlon(tm) 64 X2 Dual Core Processor 4400+ , RAM 7,00GB	Total elapsed time is 0h 0m 0s	Total elapsed time is 0h 0m 0s	Total elapsed time is 0h 0m 0s
Windows XP Genuine Intel(R) CPU T2400 @1.83GHz 1.83GHz, RAM: 1,99GB	Total elapsed time is 0h 0m 1s	Total elapsed time is 0h 0m 1s	Total elapsed time is 0h 0m 1s

As can be seen from the table, depending on the number of flights per day and different equipments with different operating systems and processors, the time spent to execute the algorithm and produce results comes to an average of 1 second.

From the list of flight schedules we have in Table1, the algorithm for generating the basic crew routes/pairs was executed to test its functionality. The output results are as shown in Figure V.

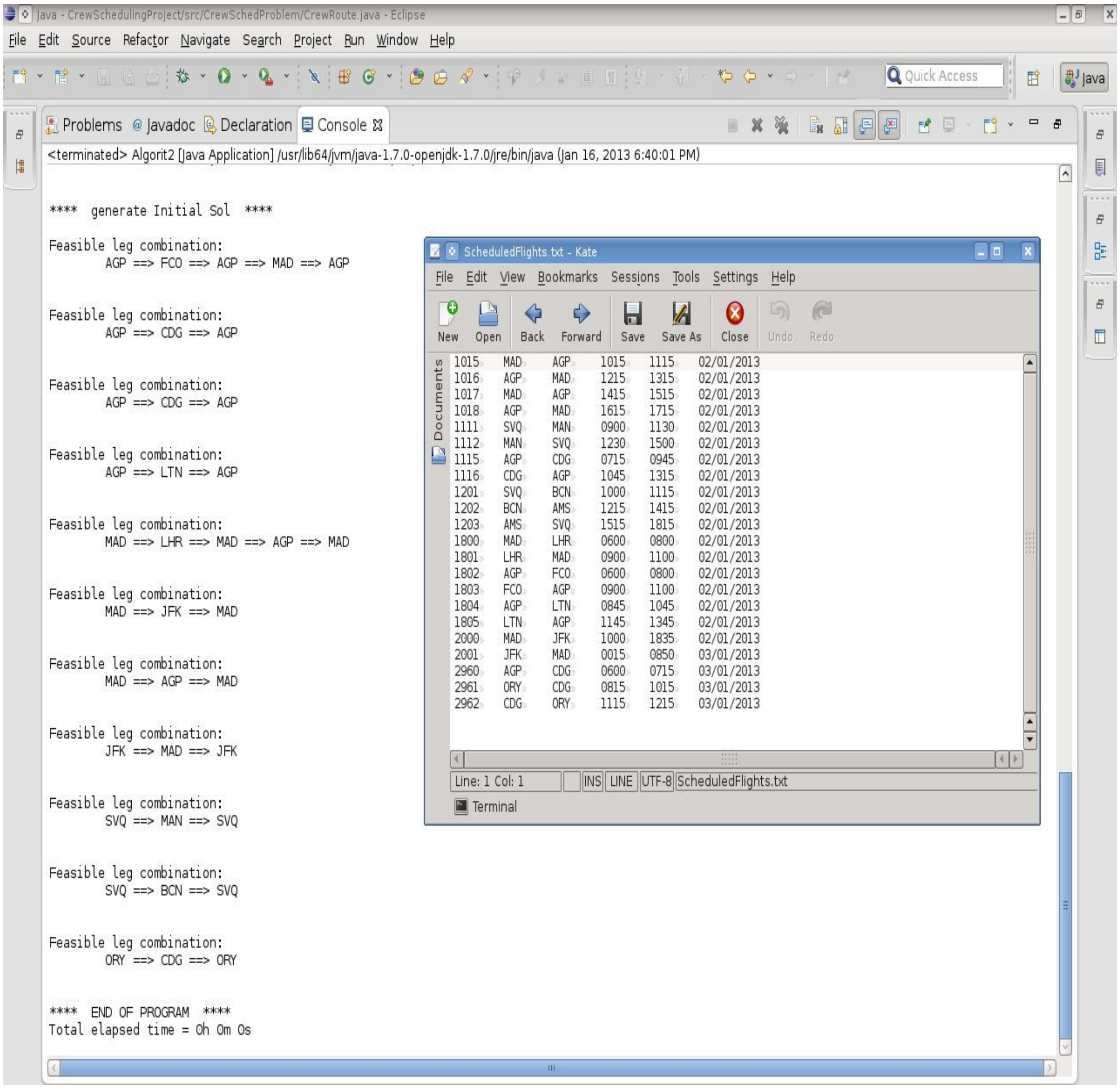


```
package CrewSchedProblem;

import java.io.BufferedWriter;
import java.io.File;
import java.io.FileWriter;
import java.io.IOException;
import java.util.*;


/*****************************************************************************************
 *                                                                                       *
 *         ALGORITHM OPTIMIZATION TO SOLVE CREW SCHEDULING PROBLEM                       *
 *                    Universitat Oberta de Catalunya                                    *
 *                 Masters in Free Software Final Project                                *
 *                                                                                       *
 * @author Christine A. Oketch                                                           *
 *                                                                                       *
 * @director Angel A. Juan                                                               *
```

```
<terminated> Algorit2 (1) [Java Application] C:\Program Files\java\jre6\bin\javaw.exe (Jan 16, 2013 4:57:20 PM)
****  WELCOME TO THIS PROGRAM  ****

 List of Available legs


Leg FLight Number=1015
                         MAD ==> AGP
         startTime=1015, finishTime=1115, date=02/01/2013


Leg FLight Number=1016
                         AGP ==> MAD
         startTime=1215, finishTime=1315, date=02/01/2013


Leg FLight Number=1017
                         MAD ==> AGP
         startTime=1415, finishTime=1515, date=02/01/2013
```

Figure V: Output of available legs

As can be seen in the diagram, the program has generated for us a list of available flight legs, the start time of the origin leg, the finish time of the destination leg and the date.
These flight legs are now used to create feasible crew route pattern, which are returned when the algorithm *generateInitialSol* is executed. See figure V: Crew routes balancing


Java - CrewSchedulingProject/src/CrewSchedProblem/CrewRoute.java - Eclipse
File Edit Source Refactor Navigate Search Project Run Window Help
Quick Access
Java
Problems
@ Javadoc
Declaration
Console
ScheduledFlights.txt - Kate
File Edit View Bookmarks Sessions Tools Settings Help
New
Open
Back
Forward
Save
Save As
Close
Undo
Redo
Documents
Line: 1 Col: 1
INS
LINE
UTF-8
ScheduledFlights.txt
Terminal

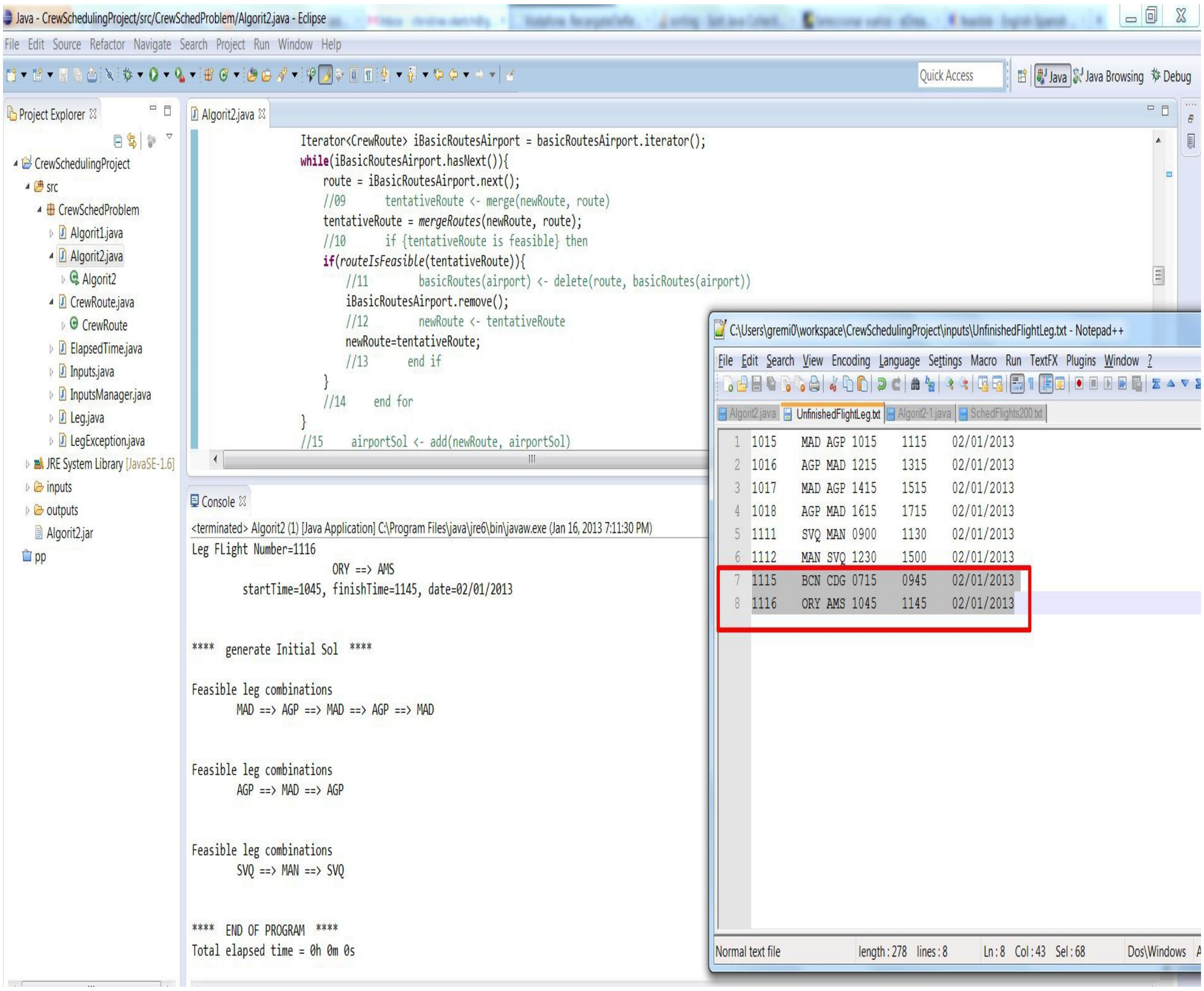
```
<terminated> Algorit2 [Java Application] /usr/lib64/jvm/java-1.7.0-openjdk-1.7.0/jre/bin/java (Jan 16, 2013 6:40:01 PM)

****  generate Initial Sol  ****

Feasible leg combination:
        AGP ==> FCO ==> AGP ==> MAD ==> AGP


Feasible leg combination:
        AGP ==> CDG ==> AGP


Feasible leg combination:
        AGP ==> CDG ==> AGP


Feasible leg combination:
        AGP ==> LTN ==> AGP


Feasible leg combination:
        MAD ==> LHR ==> MAD ==> AGP ==> MAD


Feasible leg combination:
        MAD ==> JFK ==> MAD


Feasible leg combination:
        MAD ==> AGP ==> MAD


Feasible leg combination:
        JFK ==> MAD ==> JFK


Feasible leg combination:
        SVQ ==> MAN ==> SVQ


Feasible leg combination:
        SVQ ==> BCN ==> SVQ


Feasible leg combination:
        ORY ==> CDG ==> ORY


****  END OF PROGRAM  ****
Total elapsed time = 0h 0m 0s
```

```
1015  MAD  AGP  1015  1115  02/01/2013
1016  AGP  MAD  1215  1315  02/01/2013
1017  MAD  AGP  1415  1515  02/01/2013
1018  AGP  MAD  1615  1715  02/01/2013
1111  SVQ  MAN  0900  1130  02/01/2013
1112  MAN  SVQ  1230  1500  02/01/2013
1115  AGP  CDG  0715  0945  02/01/2013
1116  CDG  AGP  1045  1315  02/01/2013
1201  SVQ  BCN  1000  1115  02/01/2013
1202  BCN  AMS  1215  1415  02/01/2013
1203  AMS  SVQ  1515  1815  02/01/2013
1800  MAD  LHR  0600  0800  02/01/2013
1801  LHR  MAD  0900  1100  02/01/2013
1802  AGP  FCO  0600  0800  02/01/2013
1803  FCO  AGP  0900  1100  02/01/2013
1804  AGP  LTN  0845  1045  02/01/2013
1805  LTN  AGP  1145  1345  02/01/2013
2000  MAD  JFK  1000  1835  02/01/2013
2001  JFK  MAD  0015  0850  03/01/2013
2960  AGP  CDG  0600  0715  03/01/2013
2961  ORY  CDG  0815  1015  03/01/2013
2962  CDG  ORY  1115  1215  03/01/2013
```

Figure V: Crew routes balancing

The algorithm does a loop to check for a complete flight leg, when it finds the feasible combination, it gives a list of the possible combinations.

In the case that it cannot find a feasible solution in a list of available flight legs, then it does not return the unfinished crew route. See figure VI: Unfinished legs below



```
Iterator<CrewRoute> iBasicRoutesAirport = basicRoutesAirport.iterator();
while(iBasicRoutesAirport.hasNext()){
    route = iBasicRoutesAirport.next();
    //09      tentativeRoute <- merge(newRoute, route)
    tentativeRoute = mergeRoutes(newRoute, route);
    //10      if {tentativeRoute is feasible} then
    if(routeIsFeasible(tentativeRoute)){
        //11        basicRoutes(airport) <- delete(route, basicRoutes(airport))
        iBasicRoutesAirport.remove();
        //12        newRoute <- tentativeRoute
        newRoute=tentativeRoute;
        //13      end if
    }
    //14    end for
}
//15    airportSol <- add(newRoute, airportSol)
```

```
<terminated> Algorit2 (1) [Java Application] C:\Program Files\java\jre6\bin\javaw.exe (Jan 16, 2013 7:11:30 PM)
Leg FLight Number=1116
                        ORY ==> AMS
        startTime=1045, finishTime=1145, date=02/01/2013


****  generate Initial Sol  ****

Feasible leg combinations
        MAD ==> AGP ==> MAD ==> AGP ==> MAD


Feasible leg combinations
        AGP ==> MAD ==> AGP


Feasible leg combinations
        SVQ ==> MAN ==> SVQ


****  END OF PROGRAM  ****
Total elapsed time = 0h 0m 0s
```

```
1015    MAD AGP 1015    1115    02/01/2013
1016    AGP MAD 1215    1315    02/01/2013
1017    MAD AGP 1415    1515    02/01/2013
1018    AGP MAD 1615    1715    02/01/2013
1111    SVQ MAN 0900    1130    02/01/2013
1112    MAN SVQ 1230    1500    02/01/2013
1115    BCN CDG 0715    0945    02/01/2013
1116    ORY AMS 1045    1145    02/01/2013
```

Figure VI: Unfinished legs

In the figure VI, the flight legs 1115 and 1116 are not displayed in the initial solution, this is because the algorithm did not find a feasible complete balanced crew route. A complete crew pattern has to start at base airport and end at base airport, but the mentioned flight legs are not finishing at base airport.

5. CONCLUSIONS

In this paper, we proposed an algorithm optimization models that is used to solve the airline crew scheduling problems by generating feasible combination of crew routes and balancing them out. The performance of the proposed algorithm was tested for its efficiency and it is worth mentioning that it was fast in comparison with manually crew pair generation, having an instance run in an average of 1 second. We also checked its functionality of generating a feasible balanced crew route patterns, we got different pattern combinations from the available flight schedules and the incomplete flight legs were not displayed as per the functionality of the algorithm. We also discussed about other methods that have been used before to solve crew scheduling problems and how it can apply to our methodology.

Although the algorithm was able to solve the main problem that is related to crew route generation and balancing, it still needs further improvement to enable it generate petal-solutions and local search. We also recommend a further study on other algorithms like using the biased randomization to obtain a random and feasible crew pairs can be applied as a future work.

ACKNOWLEDGEMENTS
I hereby express my sincere gratitude to Dr. Angel A. Juan for his invaluable
guidance, help and for providing me with the necessary materials that were required for the successful preparation and completion of this paper.

I would also like to thank to my family and friends for their endless support and help.

REFERENCES

1) AhmadBeygi, S., Cohn, A., & Weir, M. (2009). An integer programming approach to generating airline crew pairings. *Computers & Operations Research*, *36*(4), 1284-1298.

2) Beasley, J. E., & Cao, B. (1996). A tree search algorithm for the crew scheduling problem. European Journal of Operational Research, Volume 94 Issue 3, 517-526.

3) Borndörfer, R., Schelten, U., Schlechte, T., & Weider, S. (2006). A column generation approach to airline crew scheduling. Operations Research Proceedings 2005, 343-348.

4) Deng, G. F., & Lin, W. T. (2010). Ant colony optimization-based algorithm for airline crew scheduling problem. *Expert Systems with Applications*. Volume38 Issue 5 Pages 5787-5793

5) Holland, J. H. (1975). Adaptation in natural and artificial systems, University of Michigan press. Ann Arbor, MI, 1(97), 5.

6) Juan, A. A., Faulin, J., Ferrer, A., Lourenço, H. R., & Barrios, B. (2011). MIRHA: multi-start biased randomization of heuristics with adaptive local search for solving non-smooth routing problems. *TOP*, 1-24.

7) Juan, A. A., Faulin, J., Jorba, J., Riera, D., & Masip, D. (2011). On the Use of Monte Carlo Simulation, Cache and Splitting Techniques to Improve the Clarke and Wright Savings Heuristics. Journal of the Operational Research Society, 62(6), 1085-1097.

8) Levine, D. (1996). Application of a hybrid genetic algorithm to airline crew scheduling. Computers & Operations Research, 23(6), 547-558.

9) Marchiori, E., & Steenbeek, A. (2000). An evolutionary algorithm for large scale set covering problems with application to airline crew scheduling. Real-World Applications of Evolutionary Computing, 370-384.

10) Rosenberger, Jay M., et al. (2002) "A stochastic model of airline operations." Transportation science Volume 36.issue 4 : Pages 357-377.

11) Santosa, B., Sunarto, A., & Rahman, A. (2010). Using Differential Evolution Method to Solve Crew Rostering Problem. Applied Mathematics, 1(4), 316-325.

12) Schaefer, Andrew J., et al. (2005) "Airline crew scheduling under uncertainty." *Transportation Science* Volume 39 Issue 3 : 340-348.

13) Shangyao Yan, Jei-Chi Chang (2002), Airline cockpit crew scheduling, European Journal of Operational Research, Volume 136, Issue 3, Pages 501-511, ISSN 0377-2217.

14) Storn, R., & Price, K. (1995). Differential evolution-a simple and efficient adaptive scheme for global optimisation over continuous spaces. International Computer Science Institute, Berkley, CA, Tech. Rep. TR, 95-012

15) Vance, P. H., Barnhart, C., Johnson, E. L., & Nemhauser, G. L. (1997). Airline crew scheduling: A new formulation and decomposition algorithm. Operations Research, 45(2), 188-200.

16) Yan, S., & Tu, Y. P. (2002). A network model for airline cabin crew scheduling. European Journal of Operational Research, 140(3), 531-540.

17) Yan, S., & Chang, J. C. (2002). Airline cockpit crew scheduling. European Journal of Operational Research, 136(3), 501-511.

18) Yen, J. W., & Birge, J. R. (2006). A stochastic programming approach to the airline crew scheduling problem. Transportation Science, Volume 40(1), 3-14.

19) Kornilakis, H., & Stamatopoulos, P. (2002). Crew pairing optimization with genetic algorithms. *Methods and Applications of Artificial Intelligence*, Pages 109-120.