Journal of the Korean
OR/ MS Society
Vol. 17, No. 3, December 1992


# Iterative Time-constraint Addition Algorithm for the Crew Scheduling Problem

Gwan-Ho Paek*

## Abstract

The size of time constraints is the critical bottleneck of the Crew Scheduling Problem (CSP). This paper deals with a method to extract the minimum required time constraints by k-shortest path algorithm. These time constraints are added as the "insurance constraints" to avoid the unnecessary tree search, which are very time-consuming procedures, for the integer solutions. The computational results show that the problem size in LP formulation could be reduced by our method.

## 1. Introduction

The scheduling of crews is a well-known combinatorial problem of the NP-complete family [2] which has attracted considerable attention from both industries and the operational research community. A tremendous amount of man-hours has been devoted to this classical problem due to its mathematical importance and economic impact. The problem, however, has refuted all the efforts so far to find a good algorithm for it.

The Crew Scheduling Problem (CSP) is defined as a cost minimization scheduling problem on the time constrained graph [1]. The objective is to find a set of paths of the least cost covering every one of a given set of tasks (flights) only once within the given work-duty-time. The work-duty-time is the maximum allowable duration of a path (set of flights) which is usually 8 hours a day according to the union regulations.

The CSP can be formulated as the following Integer Programming (IP) format [11]. The objective function (1) is to minimize the transition cost between flights and the crew costs at the same time. The subscript 0 is used to refer to both super-source S and super-sink T, when no confusion can arise. The Constraints (2) are the typical network flow constraints which define

* Electronics and Telecommunications Research Institute

the out-flow to be exactly the same as the in-flow at each vertex. The Constraints (3) are time constraints derived from the work-duty-time feasibility to accept only the path whose time duration is not greater than the work-duty-time. The Constraints (4) are the binary constraints which guarantee the connection between flights must be 0 (connected) or 1 (disconnected). Finally the Constraints (5) are the integer constraints which limit the number of crews should be the same as the out-flow form the super-source and the in-flow to the super-sink as well.

$$\text{Minimize} \quad z=\sum_{(i,j)\in\bar{N}} C_{ij}X_{ij}+B_kY_k \tag{1}$$

$$\text{subject to} \quad \left.\begin{aligned} &\sum_{j=1}^{n} X_{0j}=Y_k \\ &\sum_{j=0}^{n} X_{ij}=1, \qquad \text{for } i=1,\cdots,n \\ &\sum_{i=0}^{n} X_{ij}=1, \qquad \text{for } j=1,\cdots,n \\ &\sum_{i=1}^{n} X_{i0}=Y_k \end{aligned}\right\} \tag{2}$$

$$\sum_{(i,j)\in P^-} X_{ij}\le |P^-|-1 \qquad \forall P^-\in F^- \tag{3}$$

$$X_{ij}\in\{0,1\} \qquad \text{for } 1\le i<j\le n \tag{4}$$

$$Y_k\ge 0 \tag{5}$$

where (i,j) is the transition arc from flight i to flight j

$C_{ij}$=linking cost between flights i and j [cost of arc(i,j)]

$X_{ij}$=1, if (i,j) is in the optimal solution,

=0, otherwise

$B_k$=crew cost per unit crew

$Y_k$=the number of crew to work in the flight schedule

$\bar{N}$ ={set of all feasible transitions}

n is the number of total flights

$P^-$ is an infeasible path whose time duration exceeds the work-duty-time. We consider also $P^-$ to be the set of all transition arcs or vertices in such a path, when no confusion arises.

$|P^-|$ is the cardinality of $P^-$

$F^-$ is the set of all paths from the super-source to the super-sink (which are infeasible because of the work-duty-time restrictions).

The usual IP approaches have been trying to formulate the CSP with all the constraints included "once and for all" in the beginning, and then algorithmic methods are used to solve the problem ([6], [7], [8]). In most cases, however, since the number of Constraints (3) explodes astronomically, the problem size becomes unmanageable soon after starting the algorithm.

In this paper we develop an algorithm to get the optimal solution to the CSP with the "minimum" number of constraints by adding the time constraints step by step. In case the solution is not integer, we have decided to add a set of "insurance constraints" prior to entering the tree search, in order to minimize the likelihood that the integer solution becomes infeasible. These "insurance" constraints are generated by enumerating paths from S to T in order of increasing cardinality (this idea of the k-shortest path algorithm can be found in [5], [10] and [12]) and testing these paths for feasibility to determine whether we use them to generate an added constraint or not.

For the realistic understanding of the CSP, consider the following example problem which has 7 flights and 5 airports to be served. We assume that the work-duty-time is 300 minutes and that the crew cost is 50. the flights are arranged in the ascending order of their starting times.

Table 1. Flights of Example Problem

| Flight No | Starting Time | Finishing Time | Starting Place | Finishing Place |
|---|---|---|---|---|
| 1 | 30 | 120 | A | C |
| 2 | 100 | 150 | A | B |
| 3 | 160 | 200 | B | C |
| 4 | 280 | 330 | C | D |
| 5 | 300 | 450 | C | E |
| 6 | 420 | 570 | D | A |
| 7 | 450 | 600 | E | D |

The problem involves 5 airports. The transition times and costs between the airports are given as follows. We assume these transition times and costs are not symmetric.

Table 2. Transition Times(Costs) of Example Problem

| Place | A | B | C | D | E |
|---|---|---|---|---|---|
| A | 0 (0) | 50(60) | 90 (90) | 120(110) | 100(100) |
| B | 80 (80) | 0 (0) | 40 (50) | 60 (60) | 80 (90) |
| C | 100 (90) | 60(70) | 0 (0) | 50 (70) | 150(140) |
| D | 150(130) | 70(80) | 80 (60) | 0 (0) | 130(150) |
| E | 120(130) | 90(80) | 110(120) | 150(140) | 0 (0) |

The network presentation of the example problem is shown in Figure 1. For simplicity, the starting and finishing vertices are omitted and the flights are shown only by the circled numbers. The objective is finding the combination of flights (path) with the minimum cost to cover each flight only once. Some of these combinations are infeasible. For example, flight 1 (the arriving airport C) cannot be connected to flights 2 (the departing airport A) and 3 (the departing airport B) because of the transition time. The work-duty-time permits flight 1 to be connected only to flight 4. We do not need to consider the other connections between flight 1 and flights 5, 6 and 7 whose time durations are greater than the work-duty-time.

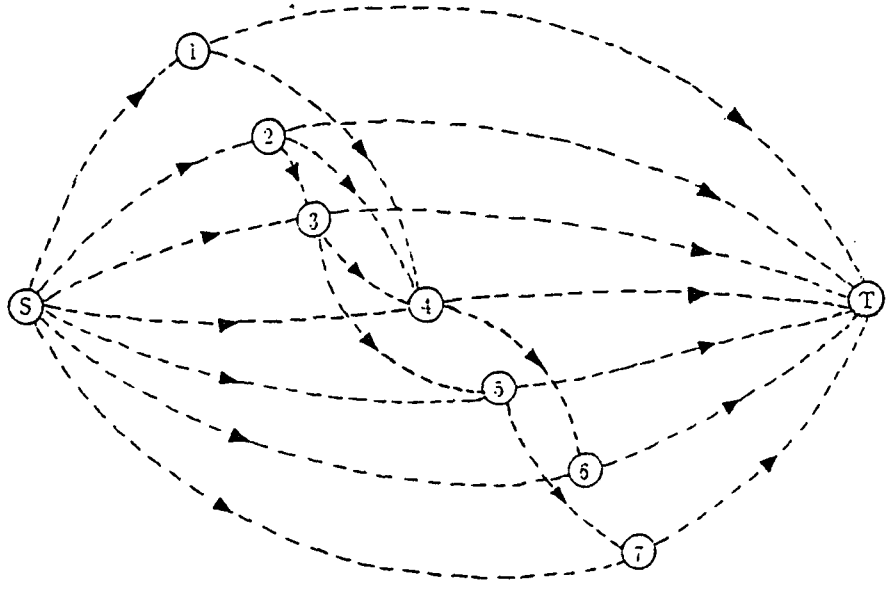


Figure 1. Network Presentation of Example Problem

## 2. Variables in the ITA Algorithm

The initial step of our formulation is to extract the necessary variables from the combination of flights and to reduce the initial problem size by removing the unnecessary time constraints for any feasible solution [3].

First, we remove variables $X_{ij}$ for $i \geq j$ because all the variables are arranged in the ascending

order of their starting times. These variables cannot be included in the feasible solutions.

We can also remove the variables corresponding to transitions between flights which cannot satisfy the time constraints even as a single transition. Variable $X_{ij}$ in the CSP represents a link of flight i to flight j. We will extract only variable $X_{ij}$ in the CSP, if it satisfies Conditions (6) and (7). The other variables can be ignored because they cannot be included in the feasible solution. The first condition says that only two flights whose transition time is less than the time gap between them are to generate a meaningful variable. The second condition says that the time duration from the starting time of one flight to the finishing time of the other must be less than the work-duty-time.

$$FT_i + T_{ij} \leq ST_j \tag{6}$$

$$FT_j + ST_i \leq WT \tag{7}$$

where $FT_i$ is the finishing time of flight i
$ST_i$ is the starting time of flight i
$T_{ij}$ is the transition time from flight i to flight j
WT is the work-duty-time.

## 3. Objective Function of the CSP

In the practial CSP there are three kinds of costs to be considered for obtaining an optimal solution. They are the flight cost, the crew cost and the transition cost as shown in the following Figure 2.

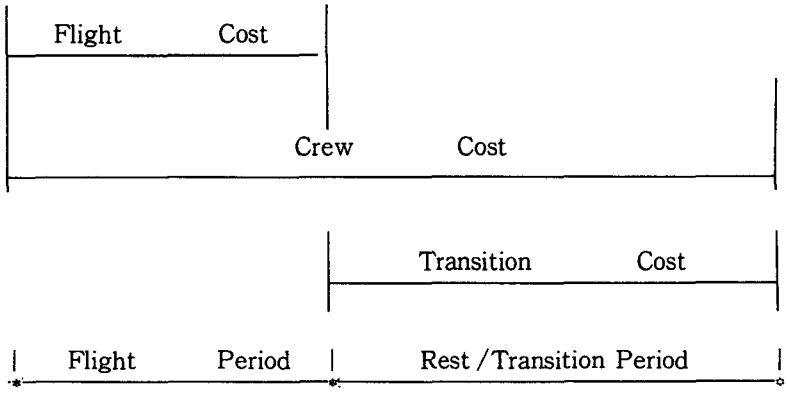


Figure 2. Cost Structure of CSP

The flight cost is the cost related to the flight itself, such as the cost for fuel and insurance of the plane and so on. This cost usually is approximately proportional to the total flight time of the airplane. In the CSP formulation we do not need to consider this cost since the summation of these flight costs is always constant, because all the flights are covered once and only once.

Crew cost $X_k$ is the cost for the crew such as the salary, bonus etc. In fact this cost has nothing to do with the flight of the airplanes. In other words this cost must be paid even if the crew is not working at all. This cost is assumed to be directly proportional to the number of crews.

Lastly transition cost $X_{ij}$ is the cost for moving the crews from one airport to another in order to enable them to work on the next flight. This cost, which is usually proportional to the distance and transition time between two airports, includes the transportation cost, the cost of lodging and so on.

# 4. Time Constraints in the ITA Algorithm

In this section we deal with the time Constraints (3) which are derived from the work-duty-time. From the infeasible path, whose time duration is greater than the work-duty-time, we can get the feasible path just by breaking the path into smaller sub-paths whose time duration is not greater than the work-duty-time. If the infeasible path is marginal, in other words if the path becomes infeasible to be added with one more fligt, this infeasible path could be excluded from the optimal solution through reducing its cardinality by one. This means at least one flight must be removed from the infeasible path to become feasible. We adopt these Constraints (3), even though they are the same as the Condition (7), because their formats are simple and are likely to provide the integer results.

Generally this kind of constraints may cause the most difficult bottleneck in the CSP since the work-duty-time regulations usually generate such numerous constraints that even a small-sized CSP cannot be handled in the apriori formulation of this type. We, therefore, propose a method to manage the numerous time constraints step by step.

## 4.1 Generation of Initial Time Constraints

Since the family $F^-$ in Constraints (3) is huge we describe the iterative generation of these constraints in our ITA algorithm. Let us consider the problem defined by (1), (2), a subset of (3) and the linear relaxation of Constraints (4) as problem P. The solution to P is, generally fractional, given by $[\bar{X}_{ij}]$. Construct subgraph G(P) of graph G describing problem P as follows [4]:

Let G=(V, A) and G(P)=(V, B)
where V is the set of all vertices of G and G(P)
A is the set of all arcs of G
B is the set of all arcs of G(P).

Define G(P) by set B as follows :

$$B=\{(i,j)\,|\,(i,j)\in A,\ \bar{X}_{ij}\neq 0\}. \tag{8}$$

For example, if Figure 1 is supposed to be G, then partial graph G(P), which is in fact one of the possible solutions (but not always integer or feasible), is derived as the following Figure 3. In our example, variables $X_{2T}$, $X_{23}$, $X_{35}$, $X_{55}$, $X_{4T}$ and $X_{46}$ are fractional and path S-2-3-5-7-T is infeasible.

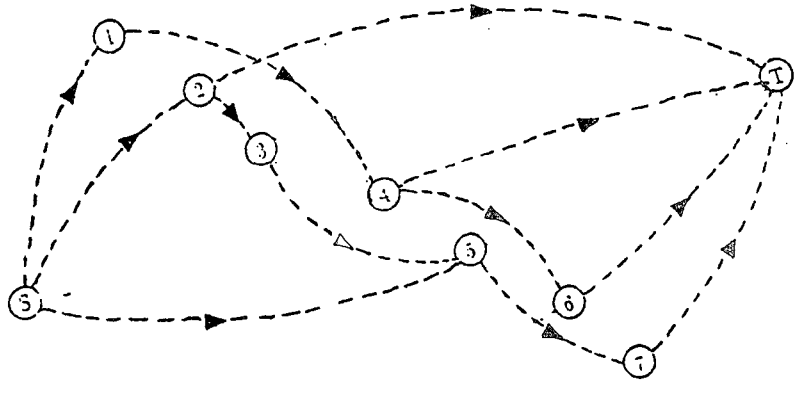


Figure 3. Partial Graph G(P) of Example Problem

We now wish to find S to T paths in G(P) and check them for the work-duty-time infeasibility. For a small non-dense graph G(P), this is an easy task. However, for a large graph, and especially one where the size of B is large, (e.g. if the LP solution is 'very' fractional), the complete set of paths from S to T in G(P) is not easy to obtain.

In our procedure, we overcame this problem by enumerating the necessary paths systematically as follows :

**Step 1** : Firstly find the 'longest' paths in G(P) from S to T. This is only a linear complexity problem when the graph is acyclic as is the case here. If the length of this longest path is feasible for the work-duty-time restriction, then all paths in G(P) are feasible and the generation of constraints from the LP solution is complete.

**Step 2** : If the longest paths in G(P) is not feasible then use the k-shortest path algorithm to generate in order of length (from the shortest to the longest) k paths from S to T. Note that if k is chosen to be very large (e.g. set to $\infty$), all paths in G(P) from S to T will be generated in length-order. In our case we used an algorithm proposed by Yen [12] to generate the k-shortest path.

**Step 3** : For each path generated, say path $P^-$ and which violates the work-duty-time restrictions, we can then check that the constraint

$$\sum_{j\in P^-,\ i\in V_j^+} + X_{ij} \leq |P^-| - 1 \tag{9}$$

where $V_j^+ = \{i,j\} \in B\}$,

is violated by the LP, and if so then include this constraint into problem P. A similar procedure can be used to generate a possible constraint from every other path $P^-$ produced at step 2. Note that Constraint (9) does not necessarily violate the LP Solution, even if $P^-$ is infeasible in terms of the work-duty-time, and therefore, it is quite possible that the addition of Constraint (9) will not disturb the existing LP solution in any way. thus, Constraint (9) must be checked against the LP solution $[\bar{X}_{ij}]$ and only added to problem P, if infeasible.

**Step 4** : When all such constraints are added to P, the LP, is solved again to obtain a new solution $[\bar{X}_{ij}]$ and the process repeated Step 1 until no constraints can be found for addition at Step 3.

For the example in Section 2, the initial LP solution is :

$$X_{14}=X_{46}=X_{23}=X_{35}=X_{57}=1$$

implying a path (say) S−2−3−5−7−T in graph G(P). This path is identified at Step 3 as one of the paths in G(P) that is infeasible (length of 500 compared with the maximum allowed length of 300), then Constraint (9) becomes :

$$X_{23}+X_{35}+X_{57}\leq 2 \tag{10}$$

(Note that if the LP solution $[\bar{X}_{ij}]$ has values $\bar{X}_{ij}=1\ \forall(i,j)\in P^-$, then Constraint (9) would always be violated by the LP solution.)

## 4.2 Strengthening Time Constraints

In the above procedure, Constraints (9) have been generated from S to T paths($P^-$, say) which were infeasible. However, it may also be possible, that a sub-path of $P^-$ could itself be infeasible in duration. Thus, if $P^-=(S,a,b,c,\cdots,i,j,k,T)$ then the sub-path (b,c,⋯,i) could also be infeasible. In this case Constraint (9) is clearly not the strongest constraint that could be written from $P^-$. Indeed each Constraint (9) could be decomposed into a set of constraints

$$\sum_{j\in W^-,\, i\in V_j^+} X_{ij}\leq |W^-|-1 \tag{11}$$

one for each sub-path $W^-$ of $P^-$ which is also infeasible. Clearly Constraints (11) dominate Constraint (9). For the example of previous section, Constraint (10), obtained from (9) directly, could also be decomposed into :

$$X_{35}+X_{57}\leq 1$$

(S−3−5−7−T is an infeasible sub-path of S−2−3−5−7−T)

and :

$$X_{23}+X_{35}\leq 1$$

(S−2−3−5−T is also an infeasible sub-path of S−2−3−5−7−T)

## 4.3 Generation of the Insurance Constraints

It was mentioned earlier, that we would like to avoid the situation, whereby a feasible LP solution is finally produced, having added a number of constraints of type (9) ; only to find that

when an integer solution is produced (after some tree search procedure is applied to the LP solution—See Section 5) we discover that the integer solution has become once-more infeasible. This is a very serious situation since the tree search is a computationally very time consuming process.

With this in mind, it was decided that after no more constraints of type (9) could be added, a new constraint-generation procedure would be applied, independently of the LP solution in order to maximize the chances that the integer solution found by the initial tree search is an optimal "feasible solution to the CSP" and no more tree searches are necessary. In view of the purpose of these additional constraints we will refer to them as "insurance" constraints.

It is easy to see that the most useful constraints of type (3), that one could use as "insurance" constraints, would be derived from those S to T paths in graph G for which, although of small cardinality, they are infeasible with respect to the work-duty-time. Some of these paths have already been generated from the LP solution, either as Constraints (9) or one of its decompositions. Other constraints, however, of the above type might well have not been generated by the investigation of the LP solution and would be candidates for the insurances constraint group. Thus, the following procedure was applied to generate the insurance constraints.

Step 1 : Apply the k-shortest path algorithm directly (using the algorithm [12] or [5] to graph G, with a values of k large enough (say k=1000) so as to generate the first k paths from S to T in order of their costs.

Step 2 : For each path $P^-$ generated in Step 1 above, and which is infeasible with respect to the work-duty-time generate the corresponding constraints.

Step 3 : Check whether the above constraints have not been previously generated during the LP solution construction and if not add them to problem P.

Step 3 : Repeat Step 2 and 3 for each path $P^-$ generated in Step 1, until a predefined maximum number of insurance constraints have been generated, or no more constraints can be produced.

The final problem P is now ready for the application of a tree search algorithm to find an integer optimal solution.

# 5. A Tree Search Algorithm for Obtaining an Integer Solution

The LP solution to the relaxed problem of (1)−(3) may not be integer. In this case we apply a tree search algorithm to ensure an integer solution. This tree search algorithm is a general generic procedure without any sophistication and will not be described in detail here. It consists of branching (to 0 or 1) on those variables with fractional values in the LP solution. The value of the LP itself is used as a bound.

# 6. Conclusion

In this paper we present an IP formulation of the CSP based on the Iterative Time-constraint Addition (ITA) algorithm. In the ITA algorithm the time constraints are added to the problem only when they are necessary to get the feasible solutions. At each iteration if the solution is not feasible in terms of work-duty-time, the unsatisfied time constraints are added. If the solution is fractional, to avoid unnecessary iterations for the integer results, the insurance constraints are produced apriori then partitioned into smaller units of a stronger form. We have defined a measure of problem "feasibilty" which is applicable to fractional solutions, and we use this measure to generate insurance constraints that inhibit the emergence of infeasible integer solutions, thus reducing the number of times we have to use a tree search to just once. The computational results show that the size of this formulation both in terms of the number of variables and the number of constraints, is reduced. With this formulation we can reduce (on average) the necessary number of variables by about 80% and the number of constraints by about 40% for the CSP's of up to 30 flights. Consequently with this algorithm large size problems could be tackled more easily, although the ITA takes about 75% longer to get the optimal solutions in terms of computing time. Despite its weakness in terms of computing time, we believe that our algorithm ITA with further elaboration could provide the basic foundation to tackle large scale combinatorial problems.

# References

[1] Arabeyre, J.P., J. Fearnley., F.C. Steiger and W. Theather, "The Airline Crew Scheduling Problem : A Survey," *Transportation Science*, Vol.3(1969), pp.140−163.

[2] Bodin, L., "Routing and Scheduling of Vehicles and Crews : the State of the Art," Samuel J. Raff(ed), *Computers and Operations Research*, Vol.10, No.2(1983), pp.113−129.

[3] Cheddad, H., *Algorithms for Crew Scheduling Problem*, Ph.D. Thesis, Imperial College, London, 1987.

[4] Christofides, N., *Graph Theory : An Algorithmic Approach*, Academic Press, 1975.

[5] Christofides, N. and J. Weston, "The Generation of k-shortest Paths in Graph," *IC OR Section Report*, Imperial College, 1992.

[6] Etschmaier, M.M., "Aircraft Scheduling : the State of the Art," *AGIFORS 24 Symposium*, Strassbourg, France, 1984.

[7] Giannessi, F. and B. Nicoletti, "The Crew Scheduling problem : A Travelling Salesman Approach," in *Combinatorial Optimization* edited by Christofides N. et al, Wiley & Sons, 1979, pp.389−419.

[8] Marsten, R.E., M. Muller and C. Killion, "Crew Planning at Flying Tiger : A Successful Application of Integer Programming," *Technical Report No.553, MIS Dept., University of Arizona*, (1978) Tucson, AZ.

[9] Mitra, G. and A.P.G. Welsh, "A Computer Based Crew Scheduling System Using a Mathematical Programming Approach," in *Computer Scheduling of Public Transport : Urban Passenger Vehicle and Crew Scheduling*, Wren A.(ed), North Holland, 1981, pp.281−296.

[10] Murty, K.G., "An Algorithm for Ranking All the Assignments in order of Increasing Cost," *Operations Research*, Vol.16(1968), pp.682−690.

[11] Paek, G-H., *Contributions to the Solution of the Crew Scheduling Problem*, Ph.D. Thesis, Imperial College, London, 1992.

[12] Yen, J.T., "Finding the k-shortest loopless paths in a network," *Management Science*, Vol. 17(1971), pp.712−716.