Journal of the Korean
OR/ MS Society
Vol. 19, No. 2, August 1994

Network Enlarging Search Technique(NEST) for the Crew Scheduling Problem

Gwan-Ho PAEK*

Abstract

We consider an algorithm for the Crew Scheduling Problem (CSP) based on the Transportation Problem approach. The main flows of the algorithm are arranged in three steps. First we propose a heuristic algorithm of the greedy principle to obtain an initial feasible solution. Secondly we present a method of formulating CSP into a Modified Transportation Problem format. Lastly the procedures of network search to get the optimal solution are presented. This algorithm can be applied to the general CSP and also to most combinatorial problems like the Vehicle Routing Problems. The computational results show that the large size CSP' could be tackled.

1. Introduction

The general Linear Programming (LP) formulation of the Crew Scheduling Problem (CSP) does not guarantee the integer results. Even though we can find the formulation to guarantee the integer results, we cannot get the optimal solution easily owing to the astronomical size of the problem.

The Transportation Problem (TP), however, which is a special type of LP formulation like the Assignment Problem (AP), has many advantages in providing a streamlined short cut to the optimal solutions. We can solve the large size CSP's by TP-based algorithms because this formulation needs much smaller size than the general LP formulation. Optimal solutions can be obtained in a relatively short period of time compared with LP as well. Above all the TP formulations guarantee the integer solution which cannot be obtained automatically by the general LP. We can save a lot of computing time without the procedures for checking integrality of the solutions.

* Department of Business Administration, Sun-Moon University

Consequently if we once formulate CSP into TP, we can claim the above advantages for our algorithms. It is our great fortune that the basic structure of CSP is similar to that of TP. Except for the row and column of super source and super sink, their structures are exactly the same. This means we can transform CSP into TP with slight modifications, even though there arise some difficulties such as the large number of zero pivots in the models [9]. This Modified Transportation Problem (MTP) table is the basic unit in our algorithm.

The formulation of MTP is given in the following format. It should be noted that the integrality constraint is omitted because TP always guarantees integer results. Except for the time constraints in (5) and (6), the formulation is the typical transportation problem. If we formulate CSP constraints only by the expressions (1), (2), (3) and (4), we can apply the TP algorithm to this problem. The additional constraints (5) and (6) can be considered only for the feasibility tests of the results from MTP.

$$\textit{Minimise} \quad Z = \sum_{i=0}^{n} \sum_{j=0}^{n} C_{ij} X_{ij}$$

$$\textit{subject to} \quad \sum_{j=0}^{n} X_{0j} = n \tag{1}$$

$$\sum_{j=0}^{n} X_{ij} = 1, \quad \text{for } i = 1, \cdots, n \tag{2}$$

$$\sum_{j=0}^{n} X_{ij} = 1, \quad \textit{for } j = 1, \cdots, n \tag{3}$$

$$\sum_{i=0}^{n} X_{i0} = n \tag{4}$$

$$FT_i + T_{ij} X_{ij} \leq ST_j \quad \textit{for } X_{ij} \subseteq S^+ \tag{5}$$

$$FT_l - ST_f \leq WT \quad \textit{for } X_{ff} X_{ll} \subseteq S^+ \tag{6}$$

where X_{ij} is the linking status, 1 (0) for linking (disconnecting)

C_{ij} is the transition cost from flight i to flight j

n is the number of total flights

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 the flight i to the flight j

S^+ is the set of feasible paths with consecutive flights

FT_l is the finishing time of the last flight in a path of S^-

ST_j is the starting time of the first flight in a path of S^+

WT is the work-duty-time allowable for a path of flights

X_{ff} is the first flight in a path of S^+

X_{ll} is the last flight in the same path of X_{ff}.

The MTP table generates several solutions, feasible or infeasible, with which we can make a graph of solution network. In this graph we can trace the optimal solution by checking the neighbours of each node of a solution until the solution is not to be improved anymore. If all the possible nodes less than the upper bound in the graph could be checked, the present upper bound of feasible solution must be optimal. This Network Enlarging Search Technique (NEST) provides us the possibility of tackling the large size CSP's.

The NEST algorithm can provide flexible approaches to most network problems and can be applied to the combinatorial problems including the CSP variants and extensions as long as they can be formulated in the form of TP. We formulate the minimum required constraints of the problem into an MTP table. We consider then the other constraints only when they are necessary for a feasibility test. So the MTP formulations for different problems can share a lot of common areas to be dealt with together in very similar ways. This is the main reason why MTP has very flexible applicability and great versatility.

The skeleton of NEST algorithm is presented as follows. The details of this algorithm with examples will be discussed in the rest of the paper.

Step 1. Initialization
Set upper bound by the heuristic algorithms
Formulate CSP with the above solution into a MTP table
Find the best solution of MTP without additional constraints
Go to step 3

Step 2. MTP table status
Find unchecked neighbour node with least value less than upper bound
If there is no unchecked node, go to step 5
Change MTP table to neighbour node status

Step 3. Feasibility test
Check the feasibility of present solution with all constraints
If feasible, replace the upper bound with a new solution

Step 4. Network of MTP node
Increase MTP network with new neighbours less than upper bound
Remove present checked node from MTP network

Step 5. Termination
If there is unchecked node in the network, go to step 2
Present upper bound is optimal solution
Stop

2. A Heuristic Algorithm for Upper Bound

Let us discuss a heuristic to get an upper bound which is a key factor to reduce the problem size in the network search. In 1976 Orloff developed a heuristic which can be considered as another variant of 3-opt algorithm proposed for the Travelling Salesman Problem (TSP) by Lin 10 years ago [5,6,8]. Orloff argued that CSP is a special case of TSP, so that Lin's heuristic can be applied to it directly. In 1981 Smith and Wren suggested an interchange heuristics [11]. The basic concept of this heuristic algorithm can also be considered as an adaptation of the 2-opt algorithm for TSP [5]. In this algorithm only the interchange between two paths is taken into consideration. If the first half of a path and the second half of another path can be joined with less total cost, a new path is created. If all the combinations have the positive change cost, the optimum is obtained.

Among this kind of heuristics, following simple heuristic algorithm "Concurrent Scheduler", proposed by Bodin in 1978, has proved quite successful for solving a variety of constrained scheduling problems [1]. The main theme of this algorithm is the greedy heuristics which have become popular in recent years owing to their simplicity [7]. It is also reported that this algorithm is widely used in practice [1].

Step 1. Number the flights in an increasing order of their starting time.
Assign flight 1 to path 1
k = 1
n = number of flights

Step 2. k = k + 1
If possible, assign flight k to the existing path with minimum linking cost
Otherwise, assign flight k to a new path

Step 3. If k < n, go to step 2
If k = n, stop.

Consider following Example Problem which has 7 flights. We assume that the work-duty-time is 480 minutes and that the crew cost is 50. For simplicity, we arrange the flights in the ascending order of their starting times.

〈Table 1〉 Flight Time of Example Problem

Flight No.	Starting Time	Finishing Time	Starting Place	Finishing Place
1	30	90	A	E
2	120	150	B	D
3	150	240	D	A
4	270	330	C	B
5	360	450	E	A
6	420	520	A	D
7	540	630	D	C

Example Problem covers 5 airports. The transition times between them are given in the following table. The places in the table indicate the airports to be visited by the flights.

〈Table 2〉 Transition Time of Example Problem

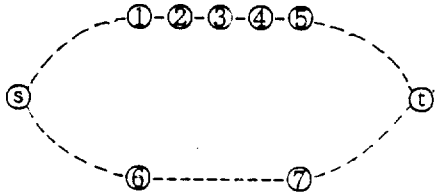
Place	A	B	C	D	E
A	0	25	30	25	30
B	30	0	20	35	30
C	35	30	0	35	40
D	20	45	30	0	30
E	25	30	25	30	0

The corresponding transition costs of Example Problem are shown in the following table. We assume that they are not symmetric and are not proportional to the transition times.

〈Table 3〉 Transition Cost of Example Problem

Place	A	B	C	D	E
A	0	60	45	28	52
B	39	0	18	55	23
C	47	50	0	33	76
D	25	41	66	0	28
E	31	75	22	24	0

By Bodin's Concurrent Scheduler, the first flight to be chosen is flight 1 then flight 2 and then flight 3 and so on. The flights 2, 3, 4 and 5 can be connected on the same path consecutively to form a rotation. But the flight 6 cannot be connected to this path because the duration of this path exceeds the work-duty-time. So flight 6 makes another separate path with the flight 7. We then get the final solution for Example Problem with the total cost 243 as follows. The time durations of all paths are less than the work−duty−time.



[Figure 1] Example Problem by Concurrent Scheduler

Bodin's algorithm confers the highest priority on the linking possibility. However the final solution is ultimately evaluated by the total cost of all paths. So, first of all, the transition costs are to be considered with the highest priority for better solutions. For some CSP's, however, the individual costs are not such good criteria to decide the feasible solutions. The total cost with respect to all possible combinations should be considered. Moreover it is tacitly assumed that there are no additional constraints in the above algorithm. The real life CSP's usually have additional time constraints. We propose therefore another heuristic algorithm based on the best path principle as follows. In this algorithm Best Combination, the total cost of each variable in the best paths is considered first.

Step 1. Initialization

Exclude the variables whose costs are greater than the crew cost

Rearrange variables in an ascending order of the individual cost (C_{ij})

Set total variable number = n

Set Upper Bound (UB) = crew cost × flights number

Set the best total cost of each variable (B_{ij})=0

Starting number (k) = 1

Step 2. Best path

a. Make an initial path with the k−th variable

b. Select the next feasible variable X_{ij} from the remaining variables

c. Add the flight in X_{ij} to the existing path or create a new path

d. If all flights are not covered, go to Step 2. b

e. Best total cost (C) = $\sum C_{ij}$ for X_{ij} in the best path

Step 3. Feedback

a. Set $B_{ij}=C$, if $B_{ij}=0$ or $B_{ij}>C$

b. Set UB=C, if $C<$UB

c. k = k + 1

d. If k > n, go to Step 4.

Otherwise go to Step 2.

Step 4. Reinitialization

Rearrange the variables in ascending order of B_{ij}

If the order is not changed, go to step 5

k= the order of variable to be changed first

Set $C = \infty$ and go to Step 2.

Step 5. Termination

Write present UB as the best feasible solution

Stop

Let us assume the transition variables of Example Problem are arranged in an increasing order as in 〈Table 4〉 Variables X_{35}, X_{47}, X_{24}, and X_{12} are not included because their costs are greater than the crew cost 50.

〈Table 4〉 First Table of Example Problem by Best Combination Algorithm

Transition Variable	Individual Cost	First Combination	Second Combination
X_{15}	0	200	189
X_{23}	0	200	189
X_{36}	0	200	195
X_{67}	0	189	–
X_{14}	22	195	–
X_{45}	23	195	–
X_{13}	24	247	–
X_{26}	25	249	–
X_{25}	28	200	–
X_{37}	28	253	–
X_{57}	28	200	–
X_{46}	39	189	–
X_{34}	45	195	–

In the above table the first combination of feasible paths starting from X_{15} is connected with the next possible variable X_{23} which generates the set $\{X_{15}, X_{23}\}$. The next possible variable is X_{36}. So the first feasible set of flights is $\{X_{15}, X_{23}, X_{36}\}$ with the cost 200. Each of the remaining flights ④,⑦ makes the independent path. For the second feasible path, the first starting variable X_{67} is connected with X_{15} to produce the feasible paths set $\{X_{67}, X_{15}\}$. The next feasible variable to be connected to this set is X_{23} to produce the set $\{X_{67}, X_{15}, X_{23}\}$. Again the variable X_{46} is added to complete the final set of this combination $\{X_{67}, X_{15}, X_{23}, X_{46}\}$ with the cost 189. The cost of the first combination of the variables X_{15} and X_{23}, 200 is replaced with this cost 189 because they are improved.

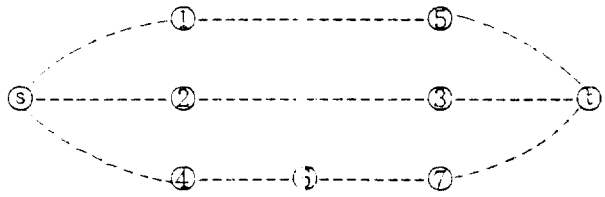
In the same manner we make the above table of the best combination algorithm. The best feasible set in the table is $\{X_{67}, X_{15}, X_{23}, X_{46}\}$ with the cost 189. We rearrange the above table in the ascending order of best combination costs. Then the above procedures are iterated until this order no longer changes.

The last table of Example Problem by the Best Combination algorithm is shown in the following table. The order of each variable cannot be changed any more by our algorithm. We assume the best solution is obtained.

〈Table 5〉 Last Table of Example Problem by Best Combination Algorithm

Transition Variable	Individual Cost	First Combination	Second Combination
X_{15}	0	189	–
X_{23}	0	189	–
X_{67}	0	189	–
X_{46}	39	189	–
X_{17}	22	195	–
X_{45}	23	195	–
X_{34}	45	195	–
X_{36}	0	200	–
X_{13}	24	241	–
X_{25}	28	262	–
X_{57}	28	267	–
X_{37}	28	267	–
X_{26}	25	270	–

The best feasible solution of Example Problem by the algorithm Best Combination is $\{X_{15}, X_{23}, X_{36}, X_{46}\}$ with cost 189. This solution is equivalent to the path in the following figure.

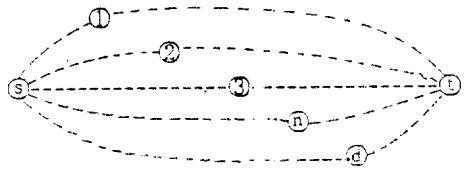


[Figure 2] Solution of Example Problem by Best Combination Algorithm

3. Formulation of Modified Transportation Problem(MTP)

3.1 Initial MTP Table

The first bottleneck of formulating CSP into the Transportation Problem (TP) is how to fix the number of demand and supply. In the general TP table of CSP, we cannot fix them because the number of paths are varied. We can circumvent this bottleneck by creating dummy paths, from super source ⓢ to super sink ⓣ, which has no physical flights on them.



ⓓ are dummy arcs

[Figure 3] Basic Conceptual Graph of MTP

The general format of the initial MTP table is shown in the following table. The variable X_{dd} represents the number of dummy arcs. It can be varied from 0 to n according to the number of allocated paths in the corresponding solution. We assume X_{dd} is 0 in the initial status. The other variables are binary to take the value 0 or 1.

For simplicity, we do not write the cost in the table for the time being. It is worth noting that the cost C_{dd} of the variable X_{dd} is assumed to be 0. The costs of each flights from super-source and to super-sink, C_{sj} and C_{it} in the first row and first column respectively, are assumed to be half the crew cost. The other costs are the actual transition costs as given in the initial data.

〈Table 6〉 Initial MTP Table

Flight	0	1	2	···	n−1	n	Total
0	0	1	1	···	1	1	n
1	1	0	0	···	0	0	1
2	1	0	0	···	0	0	1
···	···	···	···	···	···	···	···
n−1	1	0	0	···	0	0	1
n	1	0	0	···	0	0	1
Total	n	1	1	···	1	1	

1) n is the number of flight

3.2 Problem Reduction in MTP Table

The problem size of the MTP formulation of CSP is to be reduced in three ways. The main criteria for these reductions are the starting time of flight, the work-duty-time and the crew cost. The first way to reduce the problem size is to eliminate the transition variable X_{ij} in which the starting time of flight i is greater than or equal to the starting time of flight j. If we rearrange the flights in the ascending order of the starting times, we do not need to consider the transition from the flight with later starting time to the flight with the earlier starting time. This condition reduces the MTP table almost to half the original table.

The second way to reduce the MTP table is to eliminate again the variables X_{ij} in which the matchings between flight i and flight j are physically impossible owing to the transition time between them or the work-duty-time. The other variables can be removed from the feasible solutions. They cannot be included in the feasible solutions.

The third way to reduce the MTP formulation size is to remove any transition variables with greater transition costs C_{ij} than the crew cost from the feasible solutions, as long as they are not the only variable in that row or column. Any feasible solutions with these variables can be improved just by breaking them into two paths. In our CSP we take the crew cost into consideration because it guarantees the global optimums. All the costs between super source/sink and flights, which will be called "depot cost" hereafter, have a special relationship between them.

We assume the depot costs are half the crew cost. It is reasonable to assume that the crew cost is shared equally by the inflows from super source and the outflows to the super sink because the inflows and outflows must be the same in the feasible solutions.

The example of the initial MTP table for Example Problem is shown in the following table. The variables X_{ij} in which i is not less than j are reduced by the first reduction rule of starting time, the variables X_{16}, X_{17}, and X_{27} are removed by the second reduction rule of work-duty-time and the variables X_{12}, X_{24}, $X_{\ }$ and X_{47} are excluded from the table by the third reduction rule of crew cost. The total cost of this MTP table is 189 as calculated by the algorithm Best Combination.

〈Table 7〉 Initial MTP Table of Example Problem

F	0	1	2	3	4	5	6	7	Total	U_i
0	4^{0}	1^{25}	1^{25}	0^{25}	1^{2}	0^{25}	0^{25}	0^{25}	7	–
1	0^{25}	*	*	0^{24}	0^{2}	1^{0}	*	*	1	–
2	0^{25}	*	*	1^{0}	*	0^{28}	0^{25}	*	1	–
3	1^{25}	*	*	*	0^{4}	*	0^{3}	0^{28}	1	–
4	0^{25}	*	*	*	*	0^{23}	1^{39}	*	1	–
5	1^{25}	*	*	*	*	*	*	0^{28}	1	–
6	0^{25}	*	*	*	*	*	*	1^{0}	1	–
7	1^{25}	*	*	*	*	*	*	*	1	–
Total	7	1	1	1	1	1	1	1		
V_j	–	–	–	–	–	–	–	–		

1) superscript is the transition cost

2) * is the removed variable

3.3 Regeneration of MTP Table

The MTP table of the CSP has a slightly different structure from the ordinary TP table. The black hole, denoted by $X_{(0)}$ in the MTP table, is one of such special characteristics. This black hole $X_{(0)}$ absorbs any basic variables in the first row or first column of the MTP table, making

one of them degenerate. Both the summation of first row and first column must be exactly the same as the flight number. The value of black hole, which represents the number of dummy arcs, is the difference between the flight number and the path number.

Because of this degeneracy, which occurs very frequently in the MTP table, we cannot confirm the chain reaction to make the closed loop with basic variables starting from each non-basic variable.

For a general m×n Transportation Problem, with n>m, with whole number supply and requirement levels, we change a_i to $a_i+\varepsilon$, b_1 to $b+m\varepsilon$, and leave the remaining levels unaltered. For computer applications it is sufficient to put $\varepsilon=1/2n$. This regeneration, which can be done at any step of the TP algorithm, is performed usually in the beginning of the beginning of the problem formulation [3].

Let us show the example of the regenerated MTP table of Example Problem in the following table. The original MTP table had only 11 basic variables, so 4 variables were degenerated. The variables X_{03}, X_{05}, X_{06} and X_{07} in the first row are degenerated to increase X_{01} from 0 to 4. The crew number k is decreased to from 7 to 3. In the same manner we can confirm the variables X_{10}, X_{20}, $X_{\beta 0}$ and X_{60} in the first column are degenerated.

Owing to the degeneration, we could not make a closed loop in this table to improve the present solution. To regenerate this table, we add the very small number ε in the first column and total. The number of basic variables are increased from 11 to 15. The number with positive or negative sign denotes the change value, and the other unchanged values are basic variables.

〈Table 8〉 Regenerated MTP Table of Example Problem

F	0	1	2	3	4	5	6	7	Total	U_i
0	$4+\varepsilon^{0}$	1^{25}	1^{25}	$+50^{25}$	1^{25}	$+50^{25}$	$+11^{25}$	$+50^{25}$	$7+\varepsilon$	0
1	ε^{25}	*	*		-38^{62}	1^{0}	*	*	$1+\varepsilon$	25
2	ε^{25}	*	*	1^{0}		$+28^{28}$	$+14^{2}$	*	$1+\varepsilon$	25
3	$1+\varepsilon^{25}$	*	*	*	0^{0}	*	-39^{0}	$+28^{28}$	$1+\varepsilon$	25
4	ε^{25}	*	*	*		0^{28}	1^{0}	*	$1+\varepsilon$	25
5	$1+\varepsilon^{25}$	*	*	*		*	*	$+28^{28}$	$1+\varepsilon$	25
6	ε^{25}	*	*	*		*	*	1^{0}	$1+\varepsilon$	25
7	$1+\varepsilon^{25}$	*	*	*		*	*	*	$1+\varepsilon$	25
Total	$7+8\varepsilon$	1	1	1		1	1	1		
V_j	0	25	25	−25	25	−25	14	−25		

1) superscript is the transition cost
2)* is the removed variable

By the general improving method for the ordinary TP's, we get the following best MTP table of Example Problem from the regenerated MTP table. This is the best feasible solution which cannot be improved any more because all the change values in the table are positive. But this best solution is not feasible because the time duration of one of its path is 510 minutes which exceeds the wok-duty-time. In case the best solution is feasible, it must be the optimal solution at the same time.

It is worth noting that the number of non-basic variables in the MTP table is constant all through our algorithm NEST. Let us name it neighbour number which is the number of neighbours of each node. In our Example Problem the neighbour number is 13. This number will be used to check the redundancy of nodes with the same values.

〈Table 9〉 MTP Table of Best Solution of Example Problem

F	0	1	2	3	4	5	6	7	Total	U_i
0	$5+\varepsilon^{0}$	1^{25}	1^{25}	$+50^{25}$	$+5^{25}$	$+27^{25}$	$+50^{25}$	$+50^{25}$	$7+\varepsilon$	0
1	$+23^{25}$	*	*	$+47^{24}$	1^{2}	ε^{0}	*	*	$1+\varepsilon$	2
2	ε^{25}	*	*	1^{0}	*	$+5^{38}$	$+25^{25}$	*	$1+\varepsilon$	25
3	ε^{25}	*	*	*	$+(^{25}$	*	1^{7}	$+28^{28}$	$1+\varepsilon$	25
4	$2\varepsilon^{25}$	*	*	*	*	$1-\varepsilon^{23}$	$+39^{30}$	*	$1+\varepsilon$	25
5	$1+\varepsilon^{25}$	*	*	*	*	*	*	$+28^{28}$	$1+\varepsilon$	25
6	ε^{25}	*	*	*	*	*	*	1^{3}	$1+\varepsilon$	25
7	$1+\varepsilon^{25}$	*	*	*	*	*	*	*	$1+\varepsilon$	25
Total	$7+8\varepsilon$	1	1	1	1	1	1	1		
V_j	0	25	25	−25	2(	−2	−25	−25		

1) superscript is the transition cost

2) * is the removed variable

At this point it is worth discussing the method to make a closed loop in the MTP table. Generally each non−basic variable can form only one closed loop. The basic feasible solution in the TP is optimal, if and only if the change value $\Delta_{ij} \geq 0$ for every non−basic variables X_{ij}[4].

where $\Delta_{ij} = C_{ij} - U_i - V_j$. If X_{ij} is a non−basic variable. Δ_{ij} is the rate of change value of objec-

tive function changes as X_{ij} is increased. U_i is the multiple of original row i and V_j is the multiple of original row (m+j) that has been subtracted (directly or indirectly) from original row 0 by Simplex Method during all iterations leading to current Simplex Table.

We can reduce the procedure to find the loop owing to the peculiar structure of the MTP table. As shown in the formulation in Section 1, the MTP is very similar to the AP. Except for the first row and column for depot, the value of the other rows and columns must be one. This characteristic implies we can form a loop in a very efficient way to save time. For example let us consider the loop starting from C_{34} in the above MTP table of the best solution of Example Problem. Suppose we choose C_{30} as the next basic variable to be added to C_{34}. Then by the general method all the next 6 candidates, all the basic variables except for C_{30} in the first column, should be considered. It is easily recognised that this method soon produces an enormous tree combination of possible loop candidates. We can avoid this problem, if the loop encounters the first column or row, just by going back to the other direction from the starting point. In our example we choose X_{14} and X_{15}, X_{45}, X_{40} consecutively until the loop encounters again another first column or row.

4. Network Search for Optimal Solution

We consider in this section the network search algorithm by the MTP network graph. This network can be made with the costs of all MTP tables, i. e. the value of objective function. Unfortunately we cannot guarantee the optimality of the solution obtained by the heuristic algorithms discussed in Section 2. We cannot improve our initial solution by the stream-lined algorithm as in the ordinary TP approach because the additional constraints of the work-duty-time are not considered in the MTP table.

The general approach to this kind of problems is the tree search in which all the plausible solutions are examined. This approach is, however, not suitable for the CSP's of large size in a real life because the nodes generated during the tree search are too numerous.

Instead we can trace the optimal solution by checking the enlarged neighbours of each node from the best solution. We search the better feasible solutions in the MTP graph on a systematic order to avoid the biased local solution. The combinations of all possible paths are checked until the optimal or satisfactory solution is obtained.

Consider the following partial sub-graph of Example Problem in which node number presents

the total cost of the MTP table. The lines between each node represent the direct transformation of each other. We exclude the nodes between 350 and 189 which is not necessary in this case. Starting from the node of the best solution 145, we can make the solution set for the better solution with the neighbours of each node. This solution set S is:

$$S = \{ Z^i \mid \bigcup_{k=0}^{n} (Z^b) \} \text{ for } Z^i < Z^u$$

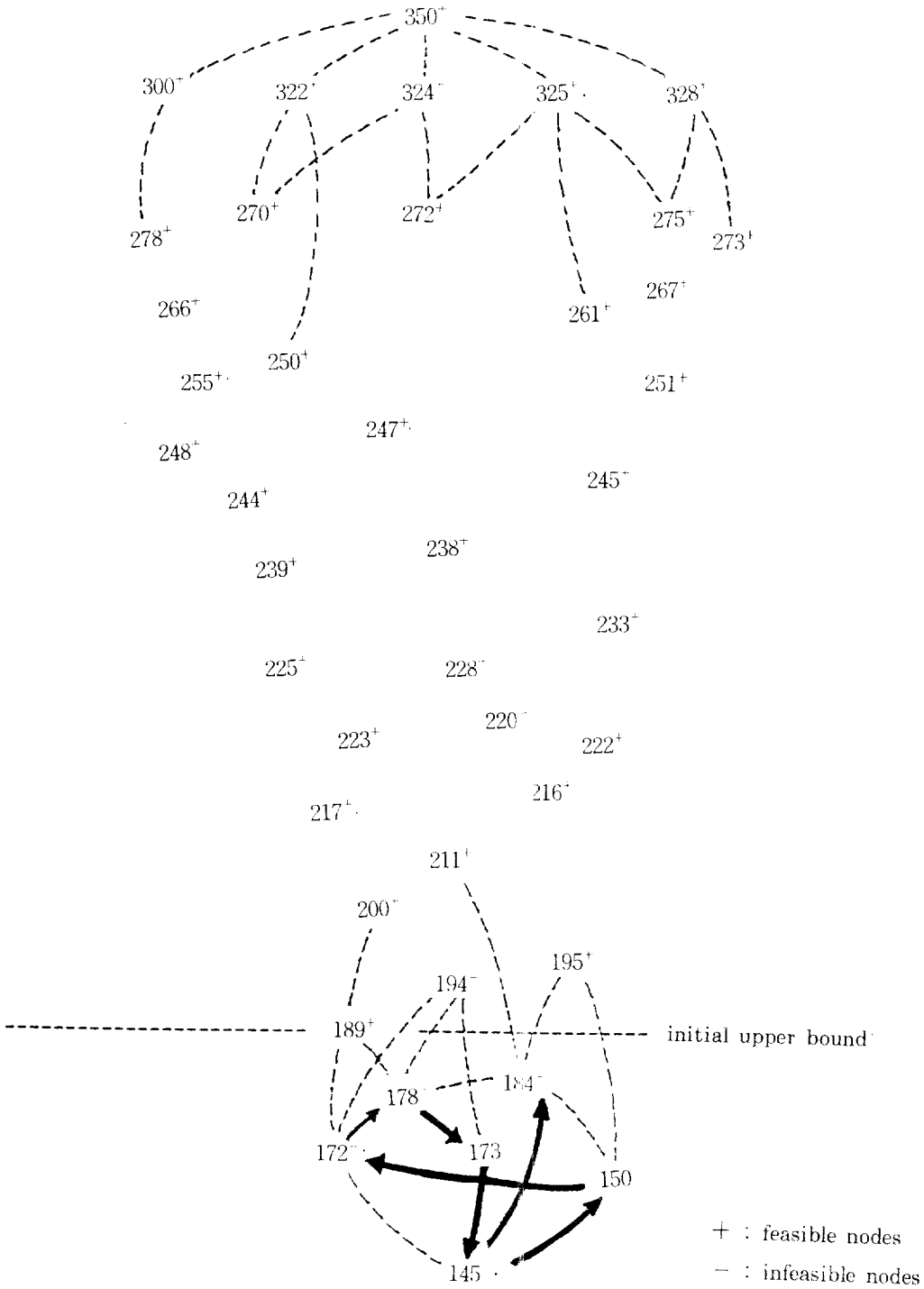
where Z^u is the node of the upper bound
Z^b is the node of the best solution
G^k is n-th outflow function in the graph
n is the number of flight

In our Example Problem, the first outflows of node 145 are the nodes 150, 172, 173, 184 and 195. The node 195 is excluded from consideration because it is higher than the upper bound. In the same manner the node 178 is generated on the second outflow function from the node 145 as well. The solution set includes all the nodes between 189 and 145. All possible paths to improve the upper bound 189 must be included in this set. So we have the solution space with 6 nodes in total, 145, 150, 172, 173, 178 and 184. The searching path starts from the node 145 and ends at the node 184. The whole searching path, presented with the heavy line in the following figure, is:

145 -- 150 -- 172 -- 173 -- 173 -- 145 -- 184

If all the nodes in the solutions set are found to be infeasible, the present upper bound must be optimal because it cannot be omproved any more. In case a new feasible solution is found, we rearrange the solution set and start algorithm again.

The solution is feasible only when it satisfies all the constraints of the problem. In our case all the time duration of the paths must not be greater than the work-duty-time. In the other variants of the CSP, of course, other constraints would be added to be satisfied.



[figure 4] Partial Sub-graph of MTP of Example Problem

5. Computational Results

The algorithm NEST is tested on IBM PC/AT with the FORTRAN compiler for the randomly generated problems. The results are summarised in Table 10, 11, and 12. Figures 5 and 6 show the general trend of the computing time as the iteration number increases. These computational results show that the large size CSP's in the real world can be tackled by our algorithm NEST.

In Table 10, the heuristic algorithm for the initial solution provides very good upper bounds for the network search. Statistics of this algorithm shows that most initial solutions are within 8% gap from the optimums. All these heuristic results are obtained within 2 seconds for the CSP's with up to 30 flights.

With the algorithm NEST we solve large size CSP's which cannot be tackled by other algorithms on the PC level. The maximum size of flights to be solved by other algorithms is about 50 flights. In Table 11 we summarised the computational results of the algorithm NEST for the CSP's with up to 100 flights. These results are obtained on the PC up to its maximum capacity, so we expect that large size CSP's can be solved, hopefully in a short period of time, on mainframes or even on super computers.

As shown in Figure 5, however, computing times are biased on the last improving step. For example, 82.8% of the entire computing times of the 10000 iterations are used only for the last 1% iterations. [Figure 6] provides the simple reason for this biased time distribution. Most of the additional times are used for searching the necessary nodes not for computing the MTP tables. The searching time and its ratio to the total computing time also increases as the iteration number increases.

In summary the computational results show that we can tackle large size CSP's. The problem size increases polynomially by our formulation. At least on the aspect of problem size, we can solve the large size problems. Additional researches, however, to reduce the biased searching times of nodes are needed. Similar results have been reported that for large CSP's with over 800 flights the run time of the main procedure is only 0.1% of the total computing time, while the other 99.9% are used for pairing generation and reductions [10].

〈Table 10〉 Performance of NEST Algorithm for CSP

	Size		Intitial		Final	
No.	F	D(%)	Z^0	G(%)	Z^*	N
1	6	32	200	0.0	200	0
2	7	43	189	0.0	189	6
3	8	33	150	0.0	150	0
4	9	28	200	0.0	200	0
5	10	31	250	0.0	250	2
6	11	18	5693	0.0	5693	2
7	12	19	4484	0.0	4484	0
8	13	15	6600	0.0	6600	4
9	14	13	5650	7.9	5238	30
10	15	17	4500	0.0	4500	8
11	16	12	5374	5.3	5291	178
12	17	16	7700	0.0	7700	46
13	18	13	7920	0.0	7920	27
14	19	11	8632	0.0	8632	5
15	20	12	10172	6.3	9568	211
16	21	10	11900	0.0	11900	3
17	22	11	11845	6.5	10184	125
18	23	10	12787	5.2	12159	42
19	24	10	13962	0.0	13962	0
20	25	8	13556	0.0	13556	2
21	26	9	17444	0.0	17444	5
22	27	9	16125	1.2	15950	42
23	28	10	17069	7.7	15849	2237
24	29	9	12643	0.1	12576	18
25	30	7	16104	6.2	15164	151

1) F : Number of flight task
2) D(%) : Density of initial MTP table (%)
3) Z^0 : Initial solution by geuristic algorithm
4) G(%) : Gap between Z^0 *and* Z^*
5) Z^* : Final solution by NEST
6) N : Node number of network search

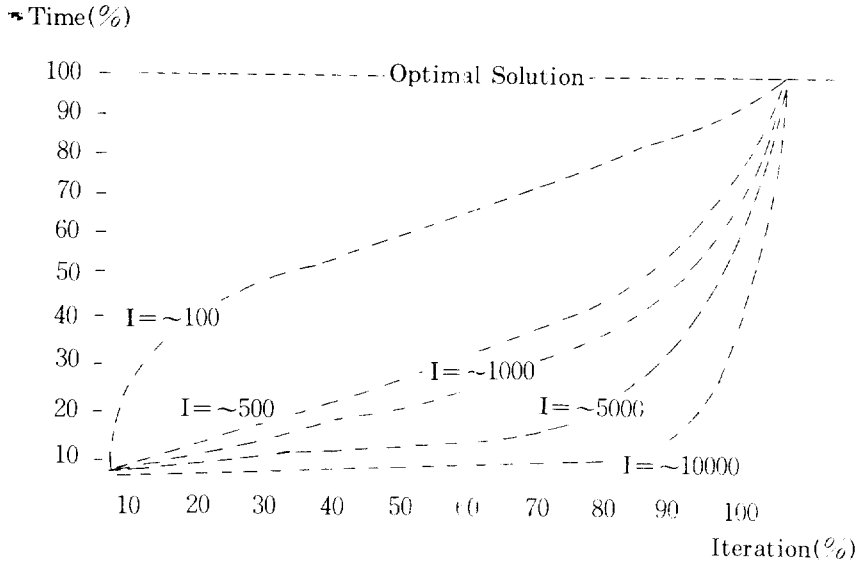
〈Table 11〉 General Performance of NEST Algorithm

Flight		Density	Node	Time
6	Min.	15.7	0	1.84
↓	Ave.	34.5	11	2.93
10	Max.	59.7	127	187.90
11	Min.	8.5	0	4.89
↓	Ave.	14.2	45	27.29
20	Max.	46.6	327	1674.49
21	Min.	4.8	0	16.76
↓	Ave.	8.9	55	149.68
30	Max.	33.1	2237	3748.34
31	Min.	3.3	0	56.47
↓	Ave.	6.5	74	229.64
40	Max.	30.2	2627	9345.92
41	Min.	2.4	7	51.67
↓	Ave.	4.8	115	379.29
50	Max.	5.3	5474	28625.91
61	Min.	1.4	0	46.74
↓	Ave.	3.1	153	427.16
70	Max.	5.8	6027	43845.91
71	Min.	1.1	10	76.74
↓	Ave.	2.6	196	654.63
80	Max.	4.3	6928	87745.97
81	Min.	0.8	12	66.71
↓	Ave.	2.1	215	824.68
90	Max.	3.8	8359	132143.82
91	Min.	0.7	46	115.37
↓	Ave.	1.8	467	1059.76
100	Max.	3.5	14312	345777.43

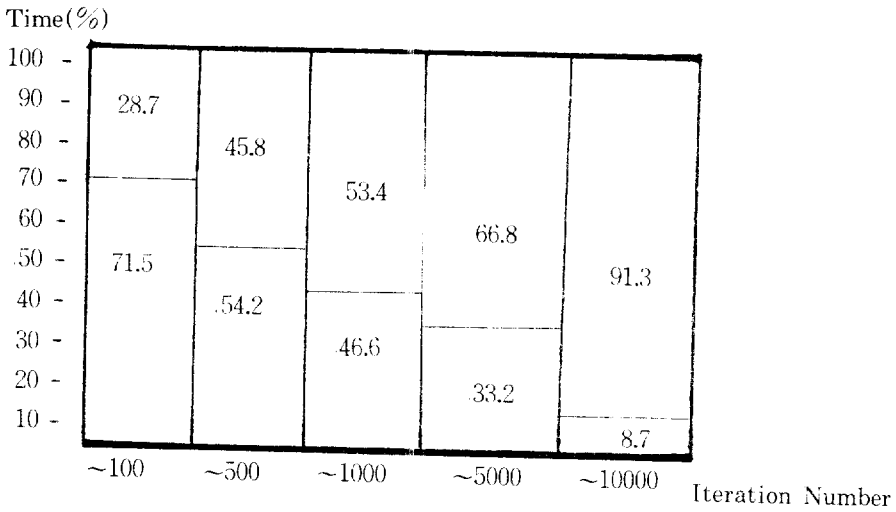
1) Density is of initial MTP table (%)

2) Node is the node number in MTP network

3) Time is seconds on IBM PC/AT



[Figure 5] Distribution of Total Computing Time by Iteration in NEST



1) Upper part is time for searching unchecked nodes (%)

2) Lower part is time for MTP calculation (%)

[Figure 6] Structure of Computing Time by Iteration in NEST

6. Conclusion

In this paper we have considered an algorithm, the Network Enlarging Search Technique (NEST), for the Crew scheduling problem (CSP) based on the transportation formulation. We have transformed the CSP into a Modified Transportation Problem (MTP) with the limited network flow constraints. The other constraints, usually the work-duty-time constraints, are considered only when they are necessary for the feasibility test of the solution.

The problem size can be reduced to the extent which is manageable by present-day computers by our algorithm. Moreover NEST algorithm can be extended to the other combinatorial problems such as General Crew Scheduling Problen (GCSP) and Vehicle Routing Problem (VRP) with slight modifications.

We propose also a heuristic algorithm based on the concept of the best combination with the aim of obtaining good upper bounds for the network search of the optimal solutions. This algorithm provides the upper bounds which are within 8% from the optimal solution. The initial upper bound is improved to the optimum by network search algorithm.

To trace the optimum, we make a network with the values of the MTP table. In this network, we can trace the optimal solution by enlarging and checking the neighbours of each node from the best solution without additional constraints.

The computational results also show that we can solve CSP's of large size in the real world. Through the extensive computational tests on the CSP's up to 100 flights, we confirm this algorithm could be applied to solve the large size CSP's in a real sense. This algorithm is, however, not so efficient in terms of computing time. We found that most of the computing times of the large problems are used for searching the necessary nodes not for the solving MTP tables. Further researches to improve the searching procedures are expected.

In summary, it might be possible to say that the large size CSP's could be tackled by our algorithm NEST. In our algorithm the problem formulation size increases polynomially.

References

[1] Bodin L. , B. Golden, A. Assad and M. Ball, "The State-of-the-Art in the Routing and Scheduling of Vehicles and Crews", *Computers and Operations Research*, Vol. 10, pp. 63-212, 1982.

[2] Bodin L. , D. Rosenfield and A. Kydes, "UCOST, A Micro Approach to a Transit Planning Problem", *Journal of Urban Analysis*, Vol. 5, No. 1, pp. 47-69, 1978.

[3] French S. , R. Hartley, L. Thomas and D. J. White, *Operational Research Technique*, Arnold, 1986.

[4] Hillier F. S. and G. J. Lieberman, *Introduction to Operations Research*, Holden-Day Inc. , Oakland, California, U. S. A. , 1986.

[5] Lin S. , "Computer Solution of the TSP", *Bell System Technical Journal*, Vol. 44, pp. 22-45, 1965.

[6] Lin S. and B. W. Kernighan, "An Effective Heuristic Algorithm for the Travelling Salesman Problem", *Operations Research*, Vol. 21, pp. 498-502, 1973.

[7] Maffioli F. , "The Complexity of Combinatorial Algorithms and the Challenge of Heuristics", *in Combinatorial Optimization* edited by Christofides N. Mingozzi A. Toth P. and Sandi C. , John Wiley & Sons, pp. 107-129, 1979.

[8] Orloff S. C. , "Route Constrained Fleet Scheduling", *Transportation Science*, Vol. 10, No. 2, pp. 149-168, 1976.

[9] Paixao J. , I. M. Branco, E. Captivo and M. V. Pato, "Bus and Crew Scheduling on a Micro Computer", in *OR Models on Microcomputer* edited by Coelho J. D. et al, North Holland, 1986.

[10] Rubin J. , "A technique for the solution of massive set covering problems with an application to airline crew scheduling", *Transportation Science*, Vol. 7, No. 1, pp. 34-48, 1973.

[11] Smith B. and A. Wren, "VAMPIRES and TASC: Two Successfuly Applied Bus Scheduling Programs", in *Computer Scheduling of Public Transport: Urban Passenger Vehicle and Crew Scheduling* edited by Wren A. , North Holland Publishing Company, pp. 97-124, 1981.