333

Working Paper

विद्याविनियोगाद्विकासः
IIM
AHMEDABAD

INDIAN INSTITUTE OF MANAGEMENT AHMEDABAD

OPTIMIZATION OF VEHICLE SCHEDULES FOR A ROAD TRANSPORT CORPORATION

By

S.R. Ankolekar

Nitin R. Patel

&

J.L. Saha

W P No. 333

October 1980

INDIAN INSTITUTE OF MANAGEMENT
AHMEDABAD

OPTIMIZATION OF VEHICLE SCHEDULES FOR A ROAD TRANSPORT CORPORATION

S.R.Ankolekar, Nitin R Patel and J.L.Saha
Indian Institute of Management, Ahmedabad

ABSTRACT

This paper discusses construction of a model to minimize fleet-size required to operate a time-table subject to a maintenance constraint. The constraint requires that vehicles be provided maintenance at least once every two days at a specific location. A heuristic algorithm was devised which also attempts to take advantage of flexibility available in trip timings. The algorithm was applied to a problem faced by one of India's largest state road transport undertakings with encouraging results. Computational experience with the algorithm is described.

OPTIMIZATION OF VEHICLE SCHEDULES FOR A ROAD TRANSPORT CORPORATION

S.R. Ankolekar, Nitin R Patel and J.L.Saha
Indian Institute of Management, Ahmedabad.

INTRODUCTION

In this paper we discuss experience with a model we are developing to provide practical assistance to the management of a large state-owned road transport corporation in planning its vehicle schedules. The corporation currently operates over 20,000 trips daily with a fleet of over 5,000 vehicles. The growth rate in the number of trips is around 10% per annum. Each year three different sets of time-tables are operated for the three seasons of summer, monsoon and winter. The magnitude of the problem and the economic importance of vehicle utilization to the corporation led the management to explore the possibility of computer-based allocation of vehicles to time-table trips.

Discussions with management revealed that there are a number of objectives that are desirable such as minimizing fleet-size, minimizing crew requirements, minimizing fuel costs etc., However, it was agreed that the overwhelming criterion should be minimizing the fleet-size. This reflects the fact that there was capital scarcity which in fact led to vehicles being used beyond their economic life. Also, in India the cost of capital relative to cost of labour being high and the fact that fewer schedules tend to require fewer crews suggested that this should be the dominating objective.

The minimum fleet-size required to operate a given time-table has been studied by several researchers. Bartlett[1] gives an algorithm to compute the minimum fleet-size. Saha[2] shows that FIFO (First-in first out) assignment of vehicles is optimal. Gertsbach and Gurevich[3] have described a procedure to generate all possible solutions using minimum fleet-size. However, their work could not be directly applied in our situation, due to following maintenance consideration.

The various trips are assigned to different depots. There are 120 such depots. Each depot operates its trips independently with its own vehicles. These vehicles can be given routine maintenance only at the depot to which they are assigned. A vehicle requires about 90 minutes for routine maintenance. As far as possible such vehicle should get a maintenance time of 90 minutes at its home depot every day. In any case every vehicle must be provided maintenance at least once every two days. To our knowledge the problem of minimizing fleet size subject to such maintenance constraints has not been addressed in the literature.

The time table had a certain flexibility so that timings of trips were not exactly fixed, but could be chosen to be within some tolerance limits of the given times. This flexibility could be exploited to reduce the fleet size. Martin-Lof[4] has suggested a branch-and-bound approach, but for the size of the problem we are concerned with, his approach is computationally infeasible. Our 'perturbation' algorithm for exploiting time-table flexibility is based on the heuristic idea suggested in reference (3).

We took up a typical depot which operates 266 trips. We found that the maintenance constraint was not satisfied by FIFO schedule. Using the results in reference (3) we were able to compute that there are over 10^{60} schedules using minimum fleet-size. Clearly, exhaustive enumeration of each is ruled out, so we developed a heuristic search procedure which systematically searches through these schedules to attempt to find a schedule meeting maintenance requirements.

1. THEORETICAL RESULTS AND ALGORITHM

1.1 DEFINITIONS AND NOTATIONS

A trip is characterised by a departure and an arrival event. An event is completely defined by the place and time of its occurance. Time establishes a precedence relationship among events occuring at a terminal resulting in a chronologically ordered sequence of events called the A.D. sequence for the terminal.

Let T represent the set of trips to be operated.

T_i represent the i^{th} trip

E_{ijk} represent an event where,
- i denotes the trip to which it belongs
- j denotes the type (j = 1 for departure, j = 2 for arrival)
- k denotes the rank of the event in the A – D sequence.

P_{ij} place of occurrence of E_{ijk}

t_{ij} time of occurrence of E_{ijk}

Q_a the A – D sequence at terminal 'a'.

We have,

$$T = \{T_i\}$$

$$T_i = \{E_{i1k_i}, E_{i2k_2}\}$$

---

* We choose a time instant during the day when no trip is operative as zero time. We can assume that this is always possible, since otherwise we can introduce a dummy terminal for each trip which is operative at time zero and break the trip into two trips such that there is an arrival and departure from the dummy terminal at time zero.

Q_a is the ordered set of all events E_{ijk} having $P_{ij} = a$, arranged in increasing order of k.

We define a surplus function over Q_a for every event giving surplus vehicles after occurrence of the event. This serves the same purpose as the deficit function proposed in reference (3) but is more suited to our purposes.

We define it as a recursive function for terminal 'a' as

$$S(E_{i_1 j_1 1}) = S_o(a) + \delta_1 \qquad (1)$$

$$S(E_{ijk}) = S(E_{i'j'k-1}) + \delta_k \qquad (2)$$

where

$\delta_k = -1$ for $j = 1$

$\delta_k = +1$ for $j = 2$

$S_o(a)$ = Number of vehicles available before the occurrence of the first event.

We can make following observations about the surplus function.

i) $S(E_{ijk})$ is the surplus number of vehicles left over immediately after the event E_{ijk}

ii) A schedule would be infeasible if $S(E_{ijk}) < 0$.

iii) $S(E^*_{ijk}) = 0$ indicates that E^*_{ijk} is a critical event, arrivals preceding this event cannot be linked to departures following this event without increases in fleet size. The subsequence of events between two critical events is known as a 'hollow zone', one which contains the time instant at which no trips are operative (time zero) will be called the 'over-night' hollow zone.

iv) $\min_k \{ S(E_{ijk}) \}$ gives redundent number of vehicles at a terminal. Clearly, for minimum fleet size schedule

$\min_k \{ S(E_{ijk}) \} = 0$ for every terminal.

v) Fleet size is given by $F = \Sigma\ S(E_{ijL})$ where E_{ijL} is the last event in the A-D sequence.

Henceforth we will use the surplus function only in the context of a minimum fleet size schedule. It may be computed as follows :

Step 1 : Compute surplus functions using definitions (1) and (2) using arbitrary S_o (a), say zero.

Step 2 : Revise surplus function for minimum fleet size as

$$S^*(E_{ijk}) = S(E_{ijk}) - \min_k \{ S(E_{ijk}) \}$$

Clearly, $F_{min} = \Sigma\ S^*(E_{ijL})$ will be the minimum fleet size.

1.2 PERTURBATION OF TIME TABLE

It is possible to reduce F_{min} by appropriately perturbing the time table within the tolerance limits. Perturbation of an event, if it results in change of its rank implies changes in surplus function of intermediate events as follows :

i) Postponement of a departure event (E_{i1k}) or preponement of an arrival event (E_{i2k}) <u>increases</u> surplus scores of intermediate events by 1 together with change in its own score depending upon its new predecessor.

ii) Conversely, preponement of E_{i1k} or postponement of E_{12k} <u>decreases</u> surplus scores of intermediate events by 1.

Thus perturbation of a trip tends to produce opposing effects at two terminals. We have reduction in fleet size if revised $\min_k \{ S(E_{ijk}) \} > 0$ and increase if $\min_k \{ S(E_{ijk}) \} < 0$ after perturbation. There will be net reduction of $\sum_{\text{all terminals}} \min \{ S(E_{ijk}) \}$.

The following algorithm attempts to increase $\min_k \{ S(E_{ijk}) \}$ at a particular terminal while maintaining $S(E_{ijk}) \geq 0$ at all other terminals.

1.2.1 PERTURBATION ALGORITHM

Step 1 Initialise I = 1, saving = 0

Step 2 For terminal I identify critical event such that $S(E^*_{ijk}) = 0$.

Step 3 Perturb the critical event and the neighbouring events as follows appropriately updating surplus scores during perturbation.

Step 3(a) If j = 1, move E_{i1k} forward towards its upper tolerance limit till its complementary event $E_{i2k'}$ crosses a critical event.

Step 3(b) If j = 2, move E_{i2k} backward towards its lower tolerance limit till itscomplementary event $E_{i1k'}$ becomes critical.

Step 4 Repeat Step 2 till all E^*_{ijk} at terminal I are exhausted.

Step 5 Compute saving = saving + $\min_k \{ S(E_{ijk}) \}$

$S'(E_{ijk}) = S(E_{ijk}) - \min_k \{ S(E_{ijk}) \}$

Step 6 I = I + 1. Go to step 2 if I $\leq$ number of terminals.

Step 7 Repeat Step 1 if saving > 0

Stop.

1.3 ALLOCATION OF TRIPS TO VEHICLES

We partition set T of trips into subsets forming supertrips, each constituting daily workload that can be assigned to a vehicle.

Let $S_j = \{ T_{j_1}, T_{j_2}, \ldots \ldots T_{j_{n_j}} \}$ where n_j is the number of trips in S_j.

$$P_{j_k 2} = P_{j_{k+1} 1} \text{ and } t_{j_k 2} < t_{j_{k+1} 1}$$

We have, $T = \bigcup_{\text{all } j} S_j$ and $S_j \cap S_k = \emptyset$ for $j \neq k$.

We classify each supertrip according to its maintenability.

i) supertrip S'_j is non-maintenable if it contains no maintenance gap of 90 minutes interval between arrival and departure at depot.

ii) Supertrip S^*_j is maintenable if it contains exactly one maintenance gap,

iii) Supertrip $S_j{}^{**}$ is 'rich' if it contains more than one maintenance gap.

Let $S' = \{ S'_j \}$, $S^* = \{ S_{j^*} \}$, $S^{**} = \{ S_{j^{**}} \}$ $S = \{ S_j \}$

$= S' \cup S^* \cup S^{**}$

A solution will be maintenance feasible if and only if a supertrip $S_i \in S^* \bigcup S^{**}$ can be assigned to a vehicle after a non-maintenable $S_i' \in S'$ is assigned to it. This implies that S must satisfy following conditions.

i) Number of non-maintenable supertrips $S_j' \in S'$ ending at any terminal must be less than or equal to number of maintenable trips $S_j \in S^* \bigcup S^{**}$ starting from that terminal.

ii) Conversely number of $S_j' \in S'$ starting from any terminal must be less than number of $S_j \in S^* \bigcup S^{**}$ ending at that terminal.

We perform the partitioning in two phases. In phase one, we construct a minimum fleet size schedule S while maximising number of maintenance gaps (M_{max}) embedded in S, using a greedy algorithm. At this stage, M_{max} maintenance gaps are likely to be unevenly distributed among S_j since we ignore the composition of S in terms of S', S* and S**. We redistribute maintenance gaps in phase two such that we heuristically maximise $|S^*|$ while satisfying vehicle maintenance feasibility conditions mentioned above, using a swapping algorithm.

1.3.1 GREEDY ALGORITHM

The greedy algorithm links arrivals to departures which are within same hollow zone so as to use the minimum fleet size. At

terminals other than depot any such set of links would suffice for the initial solution, in fact, we use well known FIFO rule for simplicity. At the depot we employ a greedy procedure which makes linkages so as to maximise the number of links containing gaps. This can be formulated as an assignment problem with rows and columns representing chronologically ordered arrivals and departures within a hollow zone.

Let $C_{ij} = \infty$ if arrival i cannot be linked with departure j

$C_{ij} = 1$ if i can be linked with j without maintenance gap

$C_{ij} = 0$ if i can be linked with j with maintenance gap.

Since the events are chronologically ordered, the assignment tableau has ∅ following properties.

$C_{ij} = 0$ implies $C_{i'j} = 0$ for $i' \leq i$ and
$C_{ij'} = 0$ for $j' \geq j$

$C_{ij} = \infty$ implies $C_{i'j} = \infty$ for $i' \geq i$ and
$C_{ij'} = \infty$ for $j' \leq j$ and

$C_{ij} = 1$ implies $C_{i'j} \geq 1$ for $i' \geq i$ and
$C_{ij'} \geq 1$ for $j' \leq j$

The assignment problem is

$$\text{Min} \quad \sum_{i,j} C_{ij} X_{ij}$$

$$\text{S.t.} \quad \sum_{i} X_{ij} = 1$$

$$\sum_{j} X_{ij} = 1 \qquad X_{ij} = 0,1$$

The assignment problem can be very simply solved by following greedy proceedure.

Step 1 Set j = last column

Step 2 Set $x_{ij} = 1$ for $i = \max_k \{ k \mid C_{kj} = 0 \}$ if possible.
If not, set $X_{ij} = 1$ for $i = \max_k \{ k \mid C_{kj} = 1 \}$
set $x_{kj} = 0$ for $k \neq i$

Step 3 Strike off row i and column j
If $C_{i+1,i} = \infty$ in the reduced problem,
set $C_{kj} = \infty$ for $k \leq i$ and $j > i$. In this case the problem is reduced to two independent subproblems.

Step 4 Repeat Step 1 till all the assignments are made.

The above procedure can be generalised to permit selection of any arbitrary column j in step 1. (Appendix 1 proves the optimality of the procedure in general case). We follow the inverse chronological order merely as a matter of convenience.

1.3.2 THE SWAPPING ALGORITHM

Every supertrip starts at the overnight hollow zone of some terminal and ends at the overnight hollow zone of another terminal. In between the trips, it may pass through one or more intermediate hollow zone, where it may be a 'neighbour in a hollow zone if their linkings can be interchanged. In other words, both the arrival events of neighbouring super trips in a hollow zone must precede both their departure events. Thus, if we break the supertrips, it is possible to swap the portions among the neighbours. We use this idea both

to remove infeasibility and heuristically maximize $|S^*|$ by breaking the $S_{j'}$ ϵ S' and swapping its portions with $S_i \epsilon S^* \cup S^{**}$ using following algorithm.

Step 1 Initialise supertrip index to i = 1

Step 2 (a) If $S_i \epsilon S^*$ go to step 6

Step 2 (b) If $S_i \epsilon S^{**}$, scan S_i between its first and last maintenance for a neighbour $S_j \epsilon S'$. If search is successful go to step 3. If not, scan S_i again in searth of a neighbour $S_j \epsilon S^*$. go to step 3, if successful and step 5 if not successful.

Step 2 (c) If $S_i \epsilon S'$, scan S_i for a neighbour $S_j \epsilon S^{**}$. Go to step 3, if successful. If not, scan S_i again for a neighbour $S_j \epsilon S^*$. Go to step 5 if unsuccessful.

Step 3 Swap portions of S_i and S_j and reclassify them in terms of S', S*, S**.

Step 4 If every supertrip in S' can be linked overnight with a supertrip in $S^* \cup S^{**}$ and if $|S'|$ < current best solution, store the solution as new current best solution.

Step 5 i = i + 1. If i $\leq$ number of supertrips go to step 2.

Step 6 Go to Step 1 if there was any improvement in S' in the last iteration.

Step 7 Print the current best solution and stop.

2. COMPUTATIONAL EXPERIENCE

The complete procedure consisting of perturbation, greedy and swapping algorithms has been coded in FORTRAN-IV PLUS and implemented on a PDP 11/70 minicomputer. Using appropriate data structures, we have been able to handle networks with 500 trips within core memory of 28 K bytes. We have used two-way linked lists to store and manipulate A-D sequences and supertrips. This seems to be a natural data structure for our procedures in view of its efficiency in both forward and backward scanning. Our computational experience is summarised in Table 1.

As can be seen from the table, small flexibility can result in significant reductions in vehicle requirement. If one is willing to prepons or postpone some of the trips by 12 minutes we can achieve an 8% reduction in fleet size. With this magnitude of reduction at each depot it is possible to reduce the fleet requirement for the corporation by about 400 buses. Heuristics succeeded in giving maintenance feasible solutions in all the cases we tried, both with real-life and random trip data.

TABLE I

COMPUTATIONAL PERFORMANCE OF THE ALGORITHM

Data	Number of trips	Min. Fleet size without Perturbation	Tolerance limits (Min)	Min. Fleet size after perturbation	Theoretical upper limit on No. of maintenances providable per day	Actual No. of maintenances provided per day	CPU time Min.
ACTUAL	286	49	± 15	45	32	31	1.09
DEPOT	286	49	± 12	45	32	30	0.66
	286	49	± 10	46	35	32	0.57
	286	49	± 5	47	47	41	0.33
RANDOM	302	75	± 15	69	65	51	0.53
	450	95	± 15	81	79	69	1.50
	454	104	± 15	84	57	55	1.7

3. AN ILLUSTRATIVE EXAMPLE

Consider an example consisting of 18 trips Table 2 covering four terminals, A–D sequence and corresponding surplus scores as given in Table 3. We have used D_i and A_i to denote departure and arrival events of trip i for better visualization.

Minimum Fleet Size $F_{min} = \sum_{\text{all Terminal}} S(\text{last event})$

$$= S(A_{12}) + S(A_{18}) + S(D_5) + S(D_{12})$$

$$= 2 + 2 + 0 + 0$$

$$= 4.$$

A-D sequences Q_1, Q_3 and Q_4 do not offer much scope for perturbation due to their tight hollow zones. Suppose we are able to prepone trip number 5 such that rank of D_5 remains unchanged at Q_3 while rank of A_5 reduces by one. Accordingly surplus scores will change as follows :

$$S'(D_6) = 1 \quad S'(A_5) = 2$$

We have $\min_j \{S'(j)\} = 1$ for terminal 2. Thus the fleet size can be reduced by 1. Consequently, all the surplus scores at Q_2 will be reduced by 1.

$$\begin{aligned} F'_{min} &= S(A_{12}) + S(A_{18}) + S(D_5) + S(D_{12}) \\ &= 2 + 1 + 0 + 0 \\ &= 3. \end{aligned}$$

Let terminal 1 be the depot.

Application of greedy algorithm will result in following linkings

Terminal 1 (D_3, A_2), maintenable arrivals (A_6, A_{12})

Terminal 2 (D_6, A_5), (D_{11}, A_{10}), (D_4, A_3), (D_{17}, A_{16}), (D_9, A_8), (D_{15}, A_{14}), $(D_2, A_1$

Terminal 3 (D_5, A_4), (D_{10}, A_9), (D_{16}, A_{15})

Terminal 4 (D_{12}, A_{11}), (D_{18}, A_{17}), (D_8, A_7), (D_{14}, A_{13})

We can identify following supertrips

$$S_1^* = \{ D_1, (A_1, D_2), (A_2, D_3), (A_3, D_4), (A_4, D_5), (A_5, D_6), A_6^* \}$$

$$S_2 = \{ D_{13}, (A_{13}, D_{14}), (A_{14}, D_{15}), A_{15}, D_{16}), (A_{16}, D_{17}), (A_{17}, D_{18}), A_{18} \}$$

$$S_3^* = \{ D_7, (A_7, D_8), (A_8, D_9), (A_9, D_{10}), (A_{10}, D_{11}), (A_{11}, D_{12}), A_{12}^* \}$$

Origins and destinations of the supertrips is as follows

Supertrip	Origin	Destination
S_1^*	1	1
S_2	1	2
S_3^*	2	1

Thus S_1^* is a circular trip which can be assigned to one vehicle. We can always assign S_3^* to vehicle after completing S_2. Therefore the solution is maintenance feasible. It is also optimal since $|S^{**}| = 0$ Neverthless, we can illustrate swapping algorithm to give an alternate optimum solution.

S_2 has S_1^* as neighbour at terminal 2, with existing links (D_{15}, A_{14}) and (D_2, A_1), If we swap to make new links (D_{15}, A_1), (D_2, A_{14}) the new supertrips would be :

$$S_1' = \{ D_1, (A_1, D_{15}), (A_{15}, D_{16}), (A_{16}, D_{17}), (A_{17}, D_{18}), A_{18} \}$$

$$S_2'^* = \{ D_{13}, (A_{13}, D_{14}), (A_{14}, D_2), (A_2, D_3), (A_3, D_4), (A_4, D_5), (A_5, D_6), A_6^* \}$$

$$S_3'^* = S_3^*$$

This schedule is again optimum and feasible.

TABLE 2

TIME TABLE DATA FOR ILLUSTRATIVE EXAMPLE

Trip Number	Place of dep	Time of dep Minutes	Place of arr.	Time of arr. Minutes
1	1	240	2	375
2	2	495	1	630
3	1	645	2	780
4	2	900	3	960
5	3	990	2	1050
6	2	1040	1	1185
7	2	405	4	495
8	4	510	2	600
9	2	795	3	855
10	3	870	2	930
11	2	1020	4	1110
12	4	1140	1	1185
13	1	300	4	345
14	4	360	2	450
15	2	630	3	690
16	3	705	2	765
17	2	855	4	945
18	4	975	2	1065

TABLE 3

A–D SEQUENCE AND SURPLUS FUNCTION FOR ILLUSTRATIVE EXAMPLE

Terminal 1		Terminal 2		Terminal 3		Terminal 4	
Q_1	S(.)	Q_2	S(.)	Q_3	S(.)	Q_4	S(.)
D_1	1	A_1	3	A_{15}	1	A_{13}	1
D_{13}	0	D_7	2	D_{16}	0	D_{14}	0
A_2	1	A_{14}	3	A_9	1	A_7	1
D_3	0	D_2	2	D_{10}	0	D_8	0
A_6	1	A_8	3	A_4	1	A_{17}	1
A_{12}	2	D_{15}	2	D_5	0	D_{18}	0
		A_{16}	3			A_{11}	1
		A_3	4			D_{12}	0
		D_9	3				
		D_{17}	2				
		D_4	1				
		A_{10}	2				
		D_{11}	1				
		D_6	0				
		A_5	1				
		A_{18}	2				

APPENDIX 1

PROOF OF MAXIMUM MAINTENANCE THEOREM

We first prove a lemma which is required in the proof of this theorem.

LEMMA 1

If $S(j)$ is the surplus function value of departure event j

then (i) $C_{ij'} = \infty$ for $i > j + S(j)$ and $j' \leq j$.

(ii) $\sum_{k,\ell} X_{k\ell} = S(j)$ for $k \leq j + S(j)$ and $\ell > j$.

Proof: (i) By definition, $S(j)$ = number of excess arrivals preceeding j.

Therefore number of arrivals preceeding $j = j + S(j)$

$\therefore$ $i > j + S(j)$ represent arrivals succeeding the departure j and hence are not linkable to $j' \leq j$.

$\therefore$ $C_{ij'} = \infty$ for $i > j + S(j)$ and $j' \leq j$.

(ii) Result (i) implies that all departures $j' \leq j$ must be linked with $i \leq j + S(j)$, leaving exactly $S(j)$ arrivals proceeding j unlinked. Since every arrival must be linked, exactly $S(j)$ of the arrivals $k \leq j + S(j)$ are linked with departures $\ell > j$.

$\therefore$ $\sum_{k,\ell} X_{k\ell} = S(j)$ for $k \leq j + S(j)$ and $\ell > j$.

Theorem : The greedy procedure outlined in the algorithm leads to optimum solution.

Proof : Let the procedure indicate $x_{ij} = 1$.
Let A be an optimum assignment with $x_{ij} = 0$. Since every row and column must be assigned $x_{i'j} = x_{ij'} = 1$ for some $i' \neq i$ and $j' \neq j$. We will consider four cases, namely

(i) $i_1 < i,\ j_1 < j$ (ii) $i_1 < i,\ j_2 > j$

(iii) $i_2 > i,\ j_2 > j$ (iv) $i_2 > i,\ j_1 < j$

along with whether $C_{ij} = 0$ or $C_{ij} = 1$.

Case A : Let $C_{ij} = 0$. This implies $C_{i_1 j} = 0$, $C_{i_2 j} = 1$, $C_{i j_2} = 0$,

$$C_{i_1 j_2} = 0,\ C_{i_2 j_2} \leq 1,\ C_{i_1 j_1} \leq 1 \quad C_{i_1 j_1} \leq C_{i j_1} \leq C_{i_2 j_1}$$

$$C_{i_2 j_2} = 1 \text{ or } \infty \quad C_{i_2 j_2} \leq C_{i_2 j} \leq C_{i_2 j_1}$$

In every case we will change the assignmentto include $x_{ij} = 1$ using valid interchanges.

Case A (i) $x_{i_1 j} = x_{i j_1} = 1 \qquad x_{ij} = x_{i_1 j_1} = 0 \in A$

It is feasible to interchange the assignments.

Let $x_{ij} = x_{i_1 j_1} = 1\ x_{i_1 j} = x_{i j_1} = 0 \in A'$.

We have $Z_{A'} = Z_A - C_{i_1 j} - C_{i j_1} + C_{ij} + X_{i_1 j_1}$

$$\leq Z_A \quad \because\ C_{ij} = C_{i_1 j} = 0 \text{ and } C_{i_1 j_1} \leq C_{i j_1}$$

Case A (ii): $X_{i_1 j} = x_{i j_2} = 1 \quad x_{ij} = x_{i_1 j_2} = 0 \in A$

Let $x_{ij} = x_{i_1 j_2} = 1 \quad x_{i_1 j} = x_{i j_2} = 0 \in A'$ which is feasible

$$Z_{A'} = Z_A - C_{i_1 j} - C_{i j_2} + C_{ij} + C_{i_1 j_2} = Z_A$$

$$\because \quad C_{i_1 j} = C_{i j_2} = C_{ij} = C_{i_1 j_2} = 0.$$

Case (A) iii: $X_{i_2 j} = x_{i j_2} = 1\ X_{ij} = x_{i_2 j_2} = 0 \in A$

Let $x_{ij} = x_{i_2j_2} = 1$ $x_{i_2j} = x_{ij_2} = 0$ $\epsilon A'$ which is feasible

$Z_{A'} = Z_A - C_{i_2j} - C_{ij_2} + C_{ij} + C_{i_2j_2} \leq Z_A$

$\therefore$ $C_{ij} = C_{ij_2} = 0$ and $C_{i_2j_2} \leq C_{i_2j}$

Case A (iv) : $C_{i_2j_1} = 1$, $x_{ij_1} = x_{i_2j} = 1$ $x_{ij} = x_{i_2j_1} = 0 \epsilon A$

Let $x_{ij} = x_{i_2j_1} = 1$ $x_{ij} = x_{i_2j} = 0$ $\epsilon A'$ which is feasible

$Z_{A'} = Z_A - C_{ij_1} - C_{i_2j} + C_{i_2j_1} \leq Z_A$

$\therefore$ $C_{i_2j} = C_{i_2j_1} = 1$ and $C_{ij} \leq C_{ij_1}$

Case A(iv) : $C_{i_2j_1} = \infty$, $x_{ij_1} = x_{i_2j} = 1$ $x_{ij} = x_{i_2j_1} = 0 \epsilon A$

In this case $x_{i_2j_1} = 1$ is not feasible.

By lemma 1 $\sum_{k,\ell} x_{k\ell} = S(j_1)$ for $k < j_1 + S(j_1)$ and $\ell > j_1$

Since $S(j_1) \geq 1$ for any departure inside the hollow zone, there must be at least one $x_{pq} = 1$ ϵA where $p < i_2$ $q > j_1$.

So we have $C_{pq} \leq 1$, $x_{pq} = x_{i_2j} = x_{ij_1} = 1 \epsilon A$

If $C_{i_2q} = \infty$ the above argument can be repeated with q taking the place of j_1. Continuing like this we must stop with a value of q such that $C_{i_2q} \leq 1$.

Now let $x_{pj} = x_{i_2q} = 1$ $x_{i_2j} = x_{pq} = 0$ $\epsilon A'$

We have $Z_{A'} = Z_A - C_{i_2j} - C_{pq} + C_{pj} + C_{i_2q} \leq Z_A$

$\therefore$ $C_{i_2j} = 1$ $C_{pj} < 1$ $C_{i_2q} < C_{pq}$

If $C_{pj_1} = \infty$ we continue above argument till $C_{pj_1} \leq 1$

Now let $x_{ij_1} = x_{pj} = 1 \quad x_{ij} = x_{pq_1} = 0 \in A'$

and $x_{ij} = x_{pj_1} = 1 \quad x_{ij_1} = x_{pj} = 0 \in A''$

$Z_{A''} = Z_{A'} - C_{ij_1} - C_{pj} + C_{ij} + C_{pj_1}$

$Z_{A''} \leq Z_{A'} \quad \therefore \quad C_{ij} = 0 \quad C_{pj} \leq C_{pj_1} \quad C_{ij_1} \leq 1$

$\leq Z_A$

Case B $C_{ij} = 1$

This implies $C_{i_2j} = \infty$ for $i_2 > i$

$C_{ij_1} = 1 \quad C_{i_1j} = 1 \quad C_{i_1j_1} = 1 \quad C_{i_1j_2} \leq 1$

$C_{ij_2} \leq 1$

Case B (i) $X_{ij_1} = x_{i_1j} = 1 \quad x_{ij} = x_{i_1j_1} = 0 \in A$

Let $X_{ij} = x_{i_1j_1} = 1 \quad x_{ij_1} = x_{i_1j} = 0 \in A'$

$Z_{A'} = Z_A - C_{i_1j} - C_{ij_1} + C_{i_1j_1} = Z_A$

$\therefore \quad C_{ij_1} = C_{i_1j} = C_{ij} = C_{i_1j_1} = 1$

Case B (ii) $X_{ij_2} = X_{i_1j} = 1 \quad x_{ij} = x_{i_1j_2} = 0 \in A$

Let $x_{ij} = x_{i_1j_2} = 1 \quad x_{i_1j} = x_{ij_2} = 0 \in A$

$Z_{A'} = Z_A - C_{i_1j} - C_{ij_2} + C_{ij} + C_{i_1j_2}$

$\leq Z_A \quad \therefore \quad C_{ij} = C_{i_1j} = 1$ and $C_{i_1j_2} \leq C_{ij_2}$

Thus in every case $Z_{A'} \leq Z_A$

Therefore, the assignment A' containing $x_{ij} = 1$ as indicated by the procedure is always superior or at least equal to any other assignment with $x_{ij} = 0$.

Q.E.D.

REFERENCES

1. T E. Bartlett (1957) "An Algorithm for the minimum number of Transport units to maintain a fixed schedule". Naval Research Logistics Quarterly, 4, p. 139-140.

2. J.L. Saha (1970) "Algorithm for Bus Scheduling Problem" Operational Research Quarterly, 21, pp. 453 - 474.

3. Gertsbach and Yu Gurevich (1977) "Constructing an Optimal Fleet for a Transport Schedule" Trans. Sci., 11, pp. 20-36.

4. Anders Martin Lef (1969) "A Branch - and - Bound Algorithm for Determining the minimal fleet size of a Transportation System" Research Report No. 6, Operations Research Center, Massachusetts Institute of Technology.