Journal of the Research Institute of Basic Sciences,
Korea Maritime University, Vol.1,1991.

Vehicle-routeing with Time Windows and Time-varying Congestion

*Jae-Yeong Shin**

Abstract

This paper discusses vehicle-routeing and scheduling problems subject to time window restriction and time-varying traffic congestion. Time-varying congestion increases the complications of the problems and, in fact, makes it challenging even to find feasible solutions. Numerous heuristics have been developed for problems with time windows only, but few for those with both time windows and time-varying congestion. We identify a simple yet robust *monotonicity* property of arrival times, which allows us to simplify the computation. This property enhances the performance of existing heuristics. The feasibility check routines built upon this property considerably reduce computational burden.

key words: heuristic, time-varying congestion, time window, vehicle-routeing problem

INTRODUCTION

Allocating and routeing vehicles are essential elements of distribution systems for collecting and delivering goods and services. The effective management of these vehicles and crews requires the solution of a variety of routeing and scheduling problems. The vehicle-routeing problem is to find the routes of

* Ddpartment of Port and Transportation Engineering Pusan 606, Korea

customers to be visited that minimize total routeing costs, satisfying some constraints.

Since Dantzig and Ramser[1] first introduced a routeing problem and proposed a linear-programming-based heuristic, much has been done to better understand the problem and to develop relevant solution schemes. Recently, much work in routeing problems has been carried out to develop realistic models and algorithms that can handle complications in practice. Schrage[2] discussed issues in more complex and realistic routeing problem such as frequency requirements, time windows, travel times under time-varying traffic congestion, multiple vehicle types and capacities, split deliveries, messy cost functions and stochastic demands.

Among these, the time window has frequently been studied in light of the generalization of basic routeing problems. In time-windowed routeing and scheduling problems, some customers must be served between the earliest and the latest delivery times. Solomon and Desrosiers[3] summarized the state of the art in solution algorithms for this class of problems, but did not discuss any issues on time-varying congestion(time-dependent internode travel times). However, in the presence of time windows, congestion effects become relevant, since the temporal issue is of greater concern than the spatial one in routeing vehicles. Therefore, both time windows and time-varying congestion need to be considered simultaneously.

Traffic congestion is particularly relevant in urban routeing systems. During rush hours, the travel time increases dramatically in most urban areas. This implies that the travel time over a road depends upon the time at which a vehicle travels along the road. Assad[4] addressed the issue of traffic congestion in travel time determination. But little work has been done on solving problems under time-varying congestion. Alfa[5] dealt with a travelling-salesman problem with time-varying travel time(but without time windows), and suggested an insertion heuristic to find a travel-time-minimizing solution. To our knowledge, there exists no algorithm that simultaneously deals with both time windows and time-varying congestion.

This paper attempts to fill this gap. We do this by modifying and extending already existing heuristics(that were originally developed for problems with time windows only), so that time-varying congestion can be addressed as well. This has been made possible through the identification of a simple yet handy property of arrival time functions that allows us to derive efficient route feasibility check routines.

PROBLEM DESCRIPTION AND TIME FEASIBILITY CONDITIONS

First, we use the following notations:

N	=	the number of customer nodes;
e_i	=	the earliest service time at node i;
l_i	=	the latest service time at node i;
$\tau_{ij}(x)$	=	the travel time from node i to node j via arc(i,j), given that the trip starts from node i at time x;
$\tau_{ij}(y)$	=	the arrival time at node j through arc(i,j) given that service beings at time y at nodt i, that is, $A_{ij}(y) = y + s_i + \tau_{ij}(y + s_i)$, where s_i is the constant service duration time for node i.

Furthermore, we use the following variables:

t_i	=	the time at which service begins for node i;
d_i	=	the *effective* latest service time at node i that allows us to maintain the feasibility of a current route.

The problem concerning us is a class of routeing problems with time windows and time-varying congestion. Given N customers to serve, multiple vehicles are routed for delivery of goods or services to all customers. The service start time at node j is constrained within a time window $[e_j, l_j]$. Thus, if a vehicle travels directly from node i to node $j, t_j = \max\{e_j, A_{ij}(t_i)\}$. The vehicles leave the depot, denoted by node 0, at time $e_0(t_0=e_0)$, and should return to the depot not later than l_0. Our objective is to find the feasible routes with minimum total travel time. Moreover, the internode travel time $\tau_{ij}(\cdot)$ is a function of the departure time representing time-varying congestion, as is the arrival time $A_{ij}(\cdot)$. In the case of multiple links, $\tau_{ij}(x)$ is the travel time along the shortest(fastest) arc at time x, and the travel time function could have kinks at time points where the shortest arc gets switched. In this sense $\tau_{ij}(\cdot)$ might well be called the internode *shortest*-travel-time function.

Now, the question is: How much is the added computational complexity of time-varying congestion? The answer is 'marginal', in the sense that it is no more difficult to solve time-windowed routeing problems with time-varying congestion than to solve those without, provided that the following monotonicity (of arrival times) and the associated feasibility check routines are utilized.

Non-passing property. For any two nodes i and j, and any two service start times x and y such that $x > y$, $A_{ij}(x) > A_{ij}(y)$, that is, earlier departure from node i guarantees earlier arrival at node j.

This property is readily acceptable in light of the fact that a vehicle that departs later cannot arrive earlier at the adjacent destination unless it bypasses other vehicles that departed earlier. The non-passing assumption is acceptable in most realistic situations.

With this non-passing property, we know that $A_{ij}(\cdot)$ is a strictly increasing function and possesses the inverse function $A_{ij}^{-1}(\cdot)$. $A_{ij}^{-1}(x)$ is interpreted as the departure time at node i so that node j can be reached at time x. This inverse function is what simplifies route feasibility checks. It is noted that the case of *constant* internode travel times trivially satisfies this non-passing property as well, so that all discussion below is equally applicable to conventonal routeing problems with constant internode travel times.

Let $(0, i_1, i_2, \cdot\cdot\cdot, i_m, 0)$ denote a partial route comprised of m nodes in sequence, $i_1, i_2, \cdot\cdot\cdot, i_m$, and the depot(denoted as 0). For notational simplicity, we relabel this route as $(0,1,2,\cdot\cdot\cdot,m,0)$. We say that a route $(0,1,2,\cdot\cdot\cdot,m,0)$ is feasible if each node in the route is visited within each corresponding time window.

The *effective* latest service start time at node i on a feasible route $(0,1,2,\cdot\cdot\cdot,m,0)$, denoted as d_i, is given from the following backward recursive relations:

$$d_i=\min\{l_i,\ A^{-1}_{i,i+1}(d_{i+1})\},\qquad 0\le i\le m-1,$$
$$d_m=\min\{l_m,\ A^{-1}_{m,0}(l_0)\}.$$

Thus, using the inverse function $A^{-1}_{ij}(\cdot)$, d_i s of any given partial route can be obtained easily from the above equations. In the absence of the non-passing property, d_i's cnannot be easily obtained. Also, the actual service start times t_i's are trivially computed from the forward recursion $t_i=\text{mas}\{e_i,\ A_{i-1,i}(t_{i-1})\}$ starting from $i=1$.

Now, using the non-passing property and these d_i's and t_i's, we can state the following three route feasibility conditions: for inserting an unrouted node a given route, for combining two distinct routes into one, and for exchanging some nodes.

Feasibility condition 1. When node u, an unrouted node, is inserted between two adjacent nodes, say, s and $s+1$ on a feasible partial route $(0,1,2,\cdot\cdot,m,0)$, the resulting new partial route remains feasible if $t_u\le d_u$, where $t_u=\max\{e_u,\ A_{su}(t_s)\}$ and $d_u=\min\{l_u,\ A^{-1}_{u,s+1}(d_{s+1})\}$.

This condition implies that we don't have to update the values of t_i's and d_i's of the original route each time a new insertion location is tried. However, without using this condition, we would have to repeatedly update t_i's for all nodes coming after each insertion point and compare with l_i's for the time-window feasibility check. As shown later, feasibility condition 1 substantially saves computaion time.

Next, we address the case of combining two distinct routes into one. This merger is typically done by linking the last node(other than the depot) of one route into the first node(other than the depot) of the other route, in order to ensure that all nodes of respective routes are also visited even after merging.

Feasibility conditions 2. When two distinct routes are merged into one by linking the last node i of one route into the first node j of the other route, then the newly combined route is feasible if $A_{ij}(t_i) \leq d_j$.

Here again, since we utilze only two values, t_i and d_j, that are already available, this feasibility check is fairly simple. This type of feasibility check is valuable to the savings heuristics based on Clarke and Wright's spproach, as will be discussed later.

Another generic approach to routeing heuristics is to exchange the visitation sequence among nodes within a given route, in particular, between a string of adjacent nodes and some single node within a given route. However, exchanging a set of nodes may disrupt the route feasibility, and thus an efficient feasibility check routine is desired.

The non-passing property and the resulting existence of $A^{-1}_{ij}(\cdot)$ satisfy this need. Consider a route $(0,1,2,\cdots,m,0)$ that is currently feasible. Now a part of this route, say, the path$(y,y+1\cdots,z)$ such that $1 \leq y \leq z \leq m$, is to be moved to other location on the route. First we can tighten up the time windows along this path as follows:

$$e'_y = e'_y$$
$$e'_i = \max\{e_i, A_{i-1,i}(e'_{i-1})\}, \qquad y+1 \leq i \leq z,$$
$$l'_z = l_z$$
$$l'_i = \min\{l_i, A^{-1}_{ij+1}(l'_{i+1})\}, \qquad y \leq i \leq z-1.$$

Feasibility condition 3. Using $[e'_i, l'_i]$in place $[e_i, l_i]$, we can state the following time feasibility condition:

(i) If a vehicle that departs node z at time e'_z cannot arrive at node u by l_u, that is, if $A_{zu}(e'_z) > l_u$, then the path $(y,y+1,\cdots,z)$ cannot precede node u.

(ⅱ) Symmetrically, if a vehicle that departs node u at time e_u cannot arrive at node y by l'_y, that is, if $A_{uy}(e_u) > l'_y$, then node u cannot precede the path $(y, y+1, \cdots, z)$.

It is noted that, once e'_i and l'_y for each node on the path$(y, y+1, \cdots, z)$ are in hand, feasibility condition 3 allows us to find a new feasible location of this path without repeatedly updating service start times of all affected nodes, which results in a substantial reduction in computation time.

When the path consists of a single node, we can easily get the precedence relationship of all node pairs from feasibility condition 3. However, this precedence relationship is justified only under the non-passing property. Using the relationship, we can adopt the exchange heuristic suggested by Psaraftis[6], even for routeing subject to time-varying congestion.

HEURISTICS: MODIFICATIONS NEEDED

With the proposed route feasibility check routines, we are now ready to discuss heuristic algorithms that make use of these routines to solve problems with time-varying congestion. Our strategy here is to develop desired heuristics, not from scratch, but by extending and modifying existing ones that were originally developed for conventional vehicle routeing problems. We present three generic heuristics for our purposes, namely, *insertion*, *savings* and *arc exchange* heuristics. Discussion is focused on extensions and modifications required to accommodate time-varying congestion, rather than on the repetition of details for each heuristic.

Insertion heuristic

The insertion heuristic takes a partial route and attempts to insert an unassigned candidate node between two selected adjacent nodes on the route. When there is no node with a feasible insertion point on the current route, the procedure switches to a new route unless it has already routed all nodes. The process involves two substeps at each iteration: *selection mode* and *insertion mode*.

Insertion mode. Insertion mode determines the insertion point of an unrouted node on a given partial route. One of the most widely used insertion criteria is the cheapest insertion[7] that minimizes the extra time required to visit the inserted node. We suggest the following modified version for problems with time-varying congestion. Suppose that an unrouted node, say node u, is to be inserted into a given feasible partial route. To determine the insertion point, define the insertion measure IC_{iuj} for a pair of adjacent nodes i and j as:

$$IC_{iuj}=t_j^n - t_j,$$

with

$$t_u=\max\{e_u,\ A_{iu}(t_i)\},$$

$$t_j^n=\max\{e_j,\ A_{uj}(t_u)\},$$

where t_i and t_j are the service start times already available before insertion, and t_u and t_j^n are the likely service start times for node u and node j, respectively, when node u is inserted between i and j. This measure indicates the delay in service time at node j due to the insertion of node u. Then, the insertion position is that with minimum IC_{iuj}.

Since the insertion of node u could alter service start times of all subsequent nodes, this minimum need be taken only among feasible locations where the time window feasibility is not broken by the insertion of node u. It is typically time consuming to identify feasible locations for each candidate insertion node u. Fortunately, however, feasibility condition 1 makes this task simple.

Selection mode. Selection mode determines a candidate unrouted node to be inserted. The four alternative criteria used-the furthest, the nearest, the cheapest[7] and Mole and Jameson's[8] criteria-are available in the literature, based upon two measures as follows:

$$SC^1_u = \mu t_{ou}(e_o) - \min\{IC_{iuj}\}, \qquad \mu \geq 0,$$

and

$$SC^2_u = \min_{k \in R}\{\tau_{ku},(t_k)\},$$

where R is the set of the nodes on the current partial route, and node i and j in R are adjacent. Mole and Jameson's criterion selects an unrouted node u with maximum SC^1_u. Setting $\mu=0$, this becomes the cheapest insertion criterion. The measure SC^2_u is the minimun travel time from the current route to node u[7]. The furthest rule selects an unrouted node with maximum SC^2_u, and the nearest rule finds one with minimum SC^2_u.

Combining the aforementioned insertion criterion and each of the four alternative selection criteria given above, we get a set of full-fledged insertion heuristics that can handle problems with time-varying congestion. The non-passing property and feasibility condition 1 contribute toward making this a straightforward extension.

Savings heuristic

Another class of construction type heuristics includes the savings heuristic extended and modified from Clarke and Wright's heuristic[9]. This heuristic begins with N trivial distinct routes on which each vehicle visits only one node. The heuristic repeatedly combines two partial routes into one by linking two respective and nodes while maintaining the feasibility.

For routeing under time-varying congestion, the saving $S_{ij}(\cdot)$ from combining two routes by linking the last node of one route, say node i, and the first node of the other, say node j, is given as a function of the start service time x at node i:

$$S_{ij}(x) = \tau_{io}(x) + \tau_{oj}(t_o) - \tau_{ij}(x), \qquad a_i \leq x \leq b_i,$$

where $a_i = \max\{e_i, A_{oi}(e_o)\}$, $b_i = \max\{l_i, A^{-1}_{ij}(l_j), A^{-1}_{ij}(A^{-1}_{jo}(l_o))\}$. In actual implementation, these savings are computed a priori for all node pairs at the same time, and are sorted and utilized (to combine associated routes) in decreasing order of values.

This savings equation is to be compared with Clarke and Wright's original (with respective definitions), which is not time-varying:

$$S_{ij}=\tau_{io}+\tau_{oj}-\tau_{ij}.$$

In our saving estimation, since it is dependent on time x at node i, we need to set x to some fixed value to compute savings. The bound $a_i \le x \le b_i$, given above is one way to make the bound tighter than the original time window, even thought it is not the tightest possible. For example, the saving can be computed at $x=(a_i+b_i)/2$.

In addition to computing and sorting the savings, we must check the time feasibility at every iteration. This is where the feasibility check routine of feasibility condition 2 is employed.

Tour improvement heuristic:

Tour improvement heuristics are often used to generate a new solution through improvements in the current one already obtained through tour construction heuristic. For example, in the *k*-opt branch exchange method presented by Lin and Kernighan[10], *k* arcs are removed from a current route and replaced by *k* arcs not in current route to form a feasible but less costly route.

Or[11] proposed a variation that attempts to exchange a string of one, two or three adjacent nodes with another node within a currently feasible route. Stewart[12] reported performance of Or's heuristic.

For our problems with both time windows and time-varying congestion, the iteration steps of Or's heuristic can be used as they are, with some form of route feasibility check routine added. Here, feasibility condition 3 can serve the purpose. However, since feasibility condition 3 is only a necessary condition, additional feasibility tests may be needed.

So far, we have discussed insertion, savings and arc exchange heuristics that make use of feasibility condition 1, feasibility condition 2 and feasibility condition 3.However, it is noted that any other routeing algorithms can benefit equally from using the suggested conditions.

COMPUTATIONAL RESULTS

Computational experiments were carried out to illustrate the computational performance of the feasibility conditions discussed in the preceding sections.

Sets of problems with 50, 75, 100 or 200 nodes were generated as test sampl-es. As a time-dependent congestion function $\tau_{ij}(\cdot)$ for each node (i,j), we assumed a horizontal line with a single peak as shown in Figure 1. This shape was chosen since it is the simple but illustrative form of traffic congestion. The peak times and congestion durations were randomly generated, and the

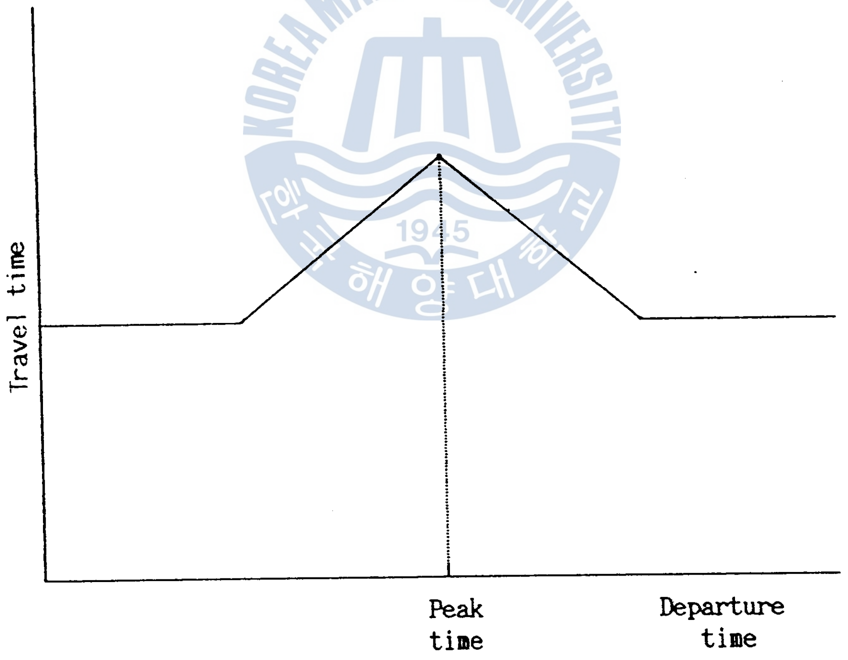

FIG. 1. *A function of the direct time between two nodes, i.e.*$\tau_{ij}(\cdot)$.

slopes of congestion functions were also randomly generated between 0 and 1 in order to satisfy the non-passing property. During non=congested hours, travel times are proportional to the internode distances. Internode distance data were taken from Krolak *et al*[13]. For 200-node problems, and from Eilon *et al*[14]. for others. All the solution procedures were implemented in Turbo Pascal on the Compaq Deskpro 386(16MHz).

First, we evaluated the effect of introducing feasibility condition 1 into an insertion heuristic that employs the cheapest insertion criterion and Mole and Jameson's selection criterion. We ran the heuristic once with this time feasibility condition and then without. Table 1 summarizes the resulting CPU times, where the substantial reduction in run-times can be seen. Even the worst cases produced with our feasibility check routine are better than the average run-times produced without it. Importantly, this phenomenon becomes more pronounced as the problem size increases. It should also be noted that the run-time reduction is more pronounced with less-tight time windows or with fewer time-windowed nodes.

TABLE 1. *Performance of feasibility condition 1(CPU time in seconds)*

No. of nodes	No. of nodes with time windows (%)	Using condition 1		Not using condition 1: average	Run-time reduction (%)
		Average	Worst		
50	50	4.8	5.3	7.1	32.4
	100	4.0	4.2	4.0	0.0
75	50	15.4	17.0	28.0	45.0
	100	11.9	12.6	11.9	0.0
100	50	32.2	34.7	70.5	54.3
	100	25.4	27.4	27.3	7.0
200	50	239.1	259.1	729.7	67.2
	100	180.9	183.9	226.2	20.0

TABLE 2. *Performance of feasibility condition 3(CPU time in seconds)*

No. of nodes	No. of nodes with time windows (%)	Using condition 3: Average	Using condition 3: Worst	Not using condition 3: average	Run-time reduction (%)
50	50	1.3	2.5	2.6	50.0
	100	0.6	0.7	1.9	73.7
75	50	6.0	8.7	9.1	34.1
	100	1.2	1.9	5.8	79.3
100	50	10.1	14.9	15.0	32.7
	100	3.0	6.1	14.7	79.6
200	50	68.1	76.7	110.8	38.5
	100	18.0	27.7	93.4	80.7

Likewise, we experimented on the effect of introducing feasibility condition 3 into a tour improvement heuristic, Or's method. Again, the suggested feasibility condition contributes much to relieving computational effort. Table 2 shows that, as with the case of feasibility condition 1, the run-time reduction here becomes more pronounced with larger problems. The only difference is that, in this case, the run-time reduction is greater with tighter time windows or an increasing number of time-windowed nodes.

It would be also interesting to see how our feasibility check routines could help routeing problems with time windows only. For this class of problem. several algorithms already exist,[3,15] but few have employed the feasibility routines of our type. For this experiment, we ran the insertion heuristic of Table 1-one set with feasibility condition 1 and the other with Solomon's feasibility condition.[15] The results are summarized in Table 3. Again, our feasibility condition perform quite well in comparison with Solomon's. In fact, the contribution of feasibility condition 1 is more appreciable here than in the time-varying congestion case. As expected, the run-time reduction here is also greater for larger problems.

TABLE 3. *Comparison with feasibility condition 1 and Solomon's conditions (CPU time in seconds)*

No. of nodes	No. of nodes with time windows (%)	Using condition 1		Using Solomon's average	Run-time reduction (%)
		Average	Worst		
50	50	2.0	2.2	3.8	47.4
	100	1.5	1.6	2.1	28.6
75	50	6.1	6.9	15.0	59.3
	100	4.4	4.8	6.6	33.3
100	50	13.0	14.4	37.9	65.7
	100	9.6	10.1	15.1	36.4
200	50	98.6	106.8	386.8	74.5
	100	68.4	68.9	125.2	45.4

From these results, though they are limited, we have found that the feasibi- lity check routines suggested in this paper help a great deal in both tour con- struction and tour improvement heuristics for time-windowed routeing problems, whether they are subject to time-varying congestion or not. This is particul- arly so with larger problems.

CONCLUDING REMARKS

The vehicle-routeing problem with traffic congestion is an important and pr- actical problem deserving serious research attention. This paper has explicit- ly considered traffic congestion (the time-varying congestion) in the vehicle- routeing problem with time windows.

The added complexity of time-varying congestion makes it time-consuming to check the time feasibility of associated routes in most routeing heuristics. To alleviate this situation, we have identified a simple, but convenient, monotonicity property of the arrival time function which allows us to derive efficient time feasibility check routines for tour construction heuristics and tour improvement heuristics. We have also modified three heuristic- the savings, the insertion and Or methods-in order to solve the problem of our concern.

It is also noted that vehicle-routeing problems with time windows only trivially satisfy the non-passing property and the suggested feasibility conditions are well applied; and further, that these conditions enhance the performance of existing heuristics.

REFERENCES

1. G.B.Dantzig and J.H.Ramser(1959) The truck dispatching problem. *Mgmt. Sci.* 6, 80-91.
2. L.Schrage(1981) Formulation and structure of more complex/realistic routing and scheduling problems. *Networks* 11, 229-232.
3. M.M.Solomon and J.F.Desrosiers(1988) Time window constrained routing and scheduling problems, *Transport. Sci.* 22, 1-13.
4. A.A.Assad(1988) Modeling and implementation issues in vehicle In *Vehicle Routing: Methods and Studies*.(B.L.Golden and A.A.Assad,Eds),pp.7-45. North -Holland, Amsterdam.
5. A.S.Alfa(1987) A heuristic algorithm for the traveling salesman problem with time-varying travel costs. *Engng Optim.* 12, 325-338.
6. H.N.Psaraftis(1983) K-interchange procedures for local search in a precedence-constrained routing problem. *Eur. J. Opl Res.* 13, 391-402.
7. D.J.Rosenkrantz, R.E.Stearns and P.M.Lewis(1977) An analysis for several heuristics for the traveling salesman problem. *SIAM J. of Computing* 6, 563-581.
8. R.H.Mole and S.R.Jameson(1976)A sequential route-building algorithm employing a generalized savings criterion. *Opl Res. Q.* 27, 503-511.
9. G.Clarke and J.W.Wright(1964) Scheduling of vehicles from a central depot to a number of delivery points. *Opns Res.* 12, 568-581.

10. S.Lin and B.W.Kernighan(1973) An effective heuristic algorithm for the traveling salesman problem. *Opns Res.* 21, 498-516.

11. I.Or(1976) Traveling salesman-type combinatorial problems and their to the logistics of blood banking PhD. Thesis, Northwestern University.

12. W.R.Stewart, Jr.(1987) Accelerated branch exchange heuristics for symmetric traveling salesman problems. *Networks* 17, 423-437.

13. P.D.Krolak, W.J.Felts and J.Nelson(1972) A man-machine approach toward solving the generalized truck-dispatching problem. *Math. Prog.* 6, 149-170.

14. S.Eilon, C.D.T. Watson-Gandy and N.Christofides(1971) *Distribution Management: Mathematical Modeling and Practical Analysis.* Griffin, London.

15. M.M.Solomon(1987) Algorithms for the vehicle routing and scheduling problems with time window constraints.