ON CONSTRAINED SHORTEST-ROUTE PROBLEMS *)

Wolfgang Gaul

Summary: Among the many constrained shortest-route problems are only few for which time-dependent restricitons are taken into consideration. Instead of determining a shortest route through a network with travel times depending on the departure time and with additional time-dependent constraints on movement and parking a dual problem and an algorithm for solving this dual program is considered giving sufficient information for the route problem. A comparison with known time-dependent shortest-route formulations is made.

Introduction

Shortest-route problems (with constraints) and algorithms for determining such routes have successfully been used for the formulation and solution of lots of network problems (see for example [6],[8],[9],[14],[16],[19]). Naturally, attempts have been made to take into consideration the known results and methods of determining shortest routes without additional restrictions (For unconstrained shortest-route (u.s.r.) problems see for example [1],[5],[6] and the references of [19].). Introductory, some of the most important (but not always very efficient) attempts are mentioned.

Let $G(N,A,l)$ be a finite, directed graph where N denotes the nodes, $A\subset N\times N\setminus\bigcup\{(i,i)\mid i\in N\}$ the (directed) arcs and $l\in R_+^{\# A}$ the length of the arcs. If G is connected, it is also called a network. A route $R(p,q) = \{p=n_o,n_1,\ldots,n_m = q\}$ from p to q, $p,q\in N$, is described by the sequence of nodes which are visited on the way from p to q in the given order. $A(R(p,q)) = \{(n_o,n_1),\ldots,(n_{m-1},n_m)\}$ is the arc set belonging to $R(p,q)$. $l(R(p,q)) = \sum_{(i,j)\in A(R(p,q))} l_{ij}$, $i,j\in N$, is the length of $R(p,q)$.

$R(p,p)$ is called a cycle and positive, negative or zero if its length $l(R(p,p))$ is positive, negative or equal to zero (Throughout this paper all values assigned to the elements of the graph G are assumed nonnegative to avoid negative cycles.). A route $R(p,q)$ is called simple or elementary if $A(R(p,q))$ or $R(p,q)$ consists of distinct elements only.

An intuitive idea is to sort all routes from p to q with respect to their lengths $(l(R_1(p,q))\leqslant l(R_2(p,q))\leqslant \ldots \leqslant l(R_k(p,q))\leqslant \ldots)$ and to search for the minimum value of k for which $R_k(p,q)$ will satisfy the additional

---

*) This research has been supported by Sonderforschungsbereich 72, University Bonn.

constraints. If $n(i,j,k)$, $i,j \in N$, defines the number of routes, among the k shortest routes from i to q, that begin by going from i to j, the algorithm has to compute

$$l(R_k(i,q)) = \min_{j \neq i} l(i,j,k) \quad , k \geq 2$$

where

$$l(i,j,k) := l_{ij} + l(R_{n(i,j,k-1)+1}(j,q)) \quad , k \geq 2$$

describes the length of the deviation $(i,j) \cup A(R_{n(i,j,k-1)+1}(j,q))$ (which corresponds to $R_h(i,q)$ for some $h \geq k$) from the k-1 shortest routes from i to q (The existence of zero-cycles must be avoided by ε-pertubating the l-data.). In most applications one is interested in elementary solutions (without cycles) (see for example [18] where u.s.r.-problems have to be determined in subgraphs of the given graph G). Naturally, the efficiency of all procedures for computing the k shortest (elementary) routes depends on the network structure and the l-data, but can be recommended if k is bounded by a small value such as in planning models for urban traffic.

There are many other possibilities of applying branch and bound procedures and the dynamic programming technique.

A class of problems which can be solved in this way is that of determining a shortest (elementary) route from p to q in a graph $G(N,A,l)$ where all nodes of a set $N_o \subset N$ must be visited exactly once, abbreviated (G,p,N_o,q) (see [15]). Here, too, the possibility of determining such problems is constrained by the magnitude of $\#N$, as it is apparent from the well-known traveling-salesman problem $(G,p,N \sim \{p\},p)$ (see [2],[13], [20]). A dynamic programming solution for the other special case $(G,p,\emptyset,q)$ is due to Bellman [1] himself.

Determining the longest (elementary) route in a graph is another well-known constrained shortest-route problem which has relations to the traveling-salesman problem (see [12]). Here, the existence of positive cycles prohibits the application of u.s.r.-procedures. Additional restrictions to exclude routes with (positive) cycles from the set of solutions (such as the subtour-elimination constraints for the traveling-salesman problem) have to be taken into consideration. Only for project planning models when the underlying graph $G(N,A,l)$ has no cycles a critical (longest) route can easily be found determining a u.s.r.-problem in $G(N,A,-l)$.

A class of problems where a reduction to u.s.r.-problems was successful is that of determining shortest routes with arc-changing costs (Such problems arise for example when reloading between different transport facilities or, in urban traffic, turning to the left or prohibitions of turning must be taken into consideration.) (see [3],[16]). While in all former mentioned problems only arcs $a \varepsilon A$ have been assigned lengths l_a now additional values

$$p_{a_1 a_2} = \begin{cases} p_{ijk} & (a_1, a_2) \varepsilon A_o \\ \infty & \text{otherwise} \end{cases}$$

with $A_o = \{(a_1, a_2) \mid a_1 = (i,j), a_2 = (j,k)\} \subset A \times A$, have to be considered. $p_{ijk} = \infty$ indicates a prohibited arc change from (i,j) to (j,k). Such problems can be solved interpreting the arcs of $G(N,A,l)$ as nodes and the A_o-arc pairs as arcs of a new graph $\tilde{G}(\tilde{N}, \tilde{A}, \tilde{l})$ with $\tilde{l}_{a_1 a_2} = l_{a_1} + p_{a_1 a_2}$ and applying an u.s.r.-procedure to $\tilde{G}$ where $\tilde{G}$, advantageously, needs not to be constructed explicitly (The elementary solutions of $\tilde{G}$ correspond to the simple ones of G, and it can be shown that the subset of simple routes contains all optimal solutions.).

As the most network problems are formulated from a static viewpoint (as all ones mentioned up to now) attention is now transferred to problems with time-dependent constraints. To the first authors who have formulated network problems from a dynamic viewpoint belong Ford/Fulkerson with their work on maximal dynamic flows (see [7],[8]). They, too, pointed out that, with much more effort, consideration could be restricted to the static case introducing a so-called "time-expanded" network. This is also true for the following discussion on shortest-route problems with time-dependent constraints. There are some authors who have dealt with these topics. Cooke/Halsey [4] use dynamic programming, Dreyfus [6] suggests the Dijkstra [5]-method, Klafszky [17] applies duality principles and gives a procedure, which, too, is closely related to paper [5], and recently, Halpern/Priess [11] have considered additional constraints of parking in the nodes, Gaul [10] has treated the problem of computing optimum routes within prescribed time-periods in case of time-dependent capacities, costs and arc-lengths.

In this paper an algorithm is presented which handles a slightly more general situation than that solved by [11], if parking constraints are excluded from consideration corresponding simplifications are pointed out and a necessary modification of [17] is given. Also, little additional expenditure is included to lighten the backtracking process of an optimal route.

Formulation of the Problem

Let the nonnegative integers $l_{ij}(t)$, $t \varepsilon \mathcal{T} = \{0, 1, \ldots, T\}$, $i,j \varepsilon N$, denote the traversal times from i to j which depend on the departure time t from node i. [1] Now, the corresponding length for R(p,q) can be described by $l(R(p,q)) = \sum_{(n_i,n_{i+1})\varepsilon A(R(p,q))} l_{n_i n_{i+1}}(l(R(p,n_i)) + t_i)$ where $t_i \varepsilon \mathcal{T}$ is the waiting time in n_i.

Searching for a shortest route within $\mathcal{T}$ time-periods means to consider the triples (R(p,q), a(R(p,q)), d(R(p,q))) where

$a(R(p,q)) = (a(n_o), \ldots, a(n_m))$ describes the arrival times,

$d(R(p,q)) = (d(n_o), \ldots, d(n_{m-1}))$ with $d(n_i)\varepsilon \mathcal{T}$ the departure times with

$$\begin{aligned} &d(n_i) \geq a(n_i) \;, \quad n_i \varepsilon R(p,q) \sim \{n_m\} \\ &a(n_o) = 0 \\ l(R(p,q)) = {}&a(n_m) \leq \mathcal{T} \qquad\qquad (1) \\ &a(n_i) = d(n_{i-1}) + l_{n_{i-1}n_i}(d(n_{i-1})) \;, \; i = 1, \ldots, m \end{aligned}$$

The parking (waiting) time in node n_i is $d(n_i) - a(n_i)$. If no parking is allowed in node i within the time intervalls $[t^i_{1k}, t^i_{2k}\rangle$, k=1,...,r(i), where "$\rangle$" symbolizes the appropriate upper bound, one has the additional constraints

$$\begin{aligned} a(i) \;\varepsilon\; [t^i_{1k}, t^i_{2k}\rangle &\Rightarrow d(i) = a(i) \\ t^i_{2k} < a(i) < t^i_{1(k+1)} &\Rightarrow d(i) \leq t^i_{1(k+1)} \end{aligned} \qquad (2)$$

If no connection from node i to node j is possible at time t let be $l_{ij}(t) = M > T$.

If $\mathfrak{A}_\mathcal{T}$ denotes the set of feasible solutions (R(p,q), a(R(p,q)), d(R(p,q))) with respect to the constraints (1), (2), the problem is to find $(R_o(p,q), a^o(R_o(p,q)), d^o(R_o(p,q)))$ (if $\mathfrak{A}_\mathcal{T} \neq \emptyset$) so that $a^o(n_m)$ describes the minimum arrival time at node $n_m = q$.

The Solution Procedure

Instead of dealing with $\mathfrak{A}_\mathcal{T}$ consider the problem

$$\max\; b(q)$$

under the constraints

$$\begin{aligned} &b(p) = 0 \\ &b(j) \leq \min_{(i,j)\varepsilon A} \; \min_{t \varepsilon \mathcal{T}_i} \{l_{ij}(t) + t\} \end{aligned} \qquad (3)$$

---

[1] For computational convenience a restriction is made to integer values (and a discrete measure of time). Also, observation time is constrained, and there are good reasons that exceeding a given time $\mathcal{T}$ is of no interest.

where $T_i = \{t \in \mathcal{T} \mid \exists\, R(p,i)$ with $l(R(p,i)) = t$ or $l(R(p,i)) = \tau < t$ and $[\tau,t) \cap \bigcup [t^i_{1k}, t^i_{2k}\rangle = \emptyset\}$.

If $\mathcal{L}$ denotes the set of feasible solutions of (3), one has $\mathcal{L} \neq \emptyset$, for $O \in \mathcal{L}$ because of $l_{ij}(t) \geq 0$.

<u>Lemma 1:</u> $\min a(q) \geq \max b(q)$

<u>Proof:</u> Either $\mathcal{O}_T = \emptyset$ yields $\min a(q) = +\infty$, or for $(R(p,q), a(R(p,q)), d(R(p,q))) \in \mathcal{O}_T$, $b \in \mathcal{L}$

$$a(q) = a(n_m) = d(n_{m-1}) + l_{n_{m-1}q}(d(n_{m-1})) \geq \min_{t \in T_{n_{m-1}}} \{ l_{n_{m-1}q}(t) + t\}$$

$$\geq \min_{(i,q) \in A} \min_{t \in T_i} \{l_{iq}(t) + t\} \geq b(q).$$

The following procedure for determining an optimum solution $b^o \in \mathcal{L}$ will indicate $\mathcal{O}_T = \emptyset$ (if $b^o(q) > T$) or construct an optimum route $R_o(p,q)$. If $N_o \subset N$ is the set of nodes with known earliest arrival times, $\beta(N_o) \in \{0,1\}^{*N}$ with

$$[\beta(N_o)]_i = \begin{cases} 0 & i \in N_o \\ 1 & \text{otherwise} \end{cases}$$

and α_i or γ_i denotes the set of feasible arrival or departure times for node i one gets

<u>Algorithm:</u>

<u>Step 1:</u> $N_o = \{p\}$, $(b(i) = 0,\ \alpha_i = \emptyset,\ \gamma_i = \emptyset,\ \forall i \in N)$

$\delta_i = \emptyset,\ i \in N \setminus \{p\}$, $\delta_p = \{[0, t^p_{11}]\} \cap \mathcal{T}$

$i(\tau) = \emptyset,\ i \in N,\ \tau \in \mathcal{T}$ $\qquad s = p$

<u>Step 2:</u>

2.1 : $\alpha^s_i = \{\tau \mid \tau = l_{si}(t) + t,\ t \in \delta_s\}$ $\quad$ (2.1–2.3: $i \in N$)

2.2 : $i(\tau) = i(\tau) \cup \{s\},\ \tau \in \alpha^s_i$

2.3.: $\alpha_i = \alpha_i \cup \alpha^s_i \setminus \gamma_i$

2.4 : $\varepsilon = \min_{i \in N} \min\{\tau \in \alpha_i\} - \max_{j \in N} b(j) \longrightarrow$ yields $i_o \in N$

2.5 : $b(i) = b(i) + [\beta(N_o)]_i\, \varepsilon$, $\quad i \in N$

2.6 : $\max_{j \in N} b(j) > T$? $\longrightarrow$ stop

2.7 : $N_o = N_o \cup \{i_o\}$

2.8 : $i_o = q$? $\longrightarrow$ stop

2.9 : $\delta_{i_o} = \{d\varepsilon \mathcal{T} \mid d\varepsilon \bigcup[\tau,\theta],\ \tau\varepsilon\alpha_{i_o},\ \bigcup[\tau,\theta) \cap \bigcup[t_{1k}^{i_o}, t_{2k}^{i_o}\rangle = \emptyset\} - \gamma_{i_o}$

2.10: $i_o(t) = \bigcup_{\varkappa \le t} i_o(\varkappa),\ \varkappa, t\varepsilon\ [\tau,\theta] \cap \delta_{i_o}$

2.11: $\gamma_{i_o} = \gamma_{i_o} \cup \delta_{i_o}$

2.12: $\alpha_{i_o} = \emptyset$

2.13: $s = i_o$

2.14: repeat step 2

In order to see how the algorithm works one has to check the single operations. Step 1 gives the starting conditions. In step 2 in 2.1, 2.3 feasible arrival times are computed where consideration is restricted to such times which have not served for determining departure times in 2.9, 2.11.
If n is the iteration index (for repeating step 2 of the algorithm) it is shown

<u>Lemma 2:</u> Computing 2.4 yields $\varepsilon \geq 0$.

<u>Proof:</u> Because of $b^1 \equiv 0$ and $1_{ij}(t) \geq 0$ one has

$$\varepsilon^1 = \min_{i\varepsilon N} \min \{\tau\varepsilon\alpha_i^1\} = \min \{\tau = 1_{si_o(1)}(t) + t \mid t\varepsilon\delta_s^1\} \geq 0$$

From $\varepsilon^{n-1} = \min \{\tau\varepsilon\alpha_{i_o(n-1)}^{n-1}\} - \max_{j\varepsilon N} b^{n-1}(j)$ it follows

$\min \{\tau\varepsilon\alpha_i^{n-1}\} \geq \min \{\tau\varepsilon\alpha_{i_o(n-1)}^{n-1}\}$ and from

$\min \{\tau\varepsilon\delta_s^n\} = \min \{\tau\varepsilon\alpha_{i_o(n-1)}^{n-1}\}$ and 2.1

$\min \{\tau\varepsilon\alpha_i^n\} \geq \min \{\tau\varepsilon\alpha_{i_o(n-1)}^{n-1}\}$ (remember $\alpha_{i_o(n-1)}^n = \alpha_s^n = \emptyset$ because of 2.12). Thus

$$\varepsilon^n = \min_{i\varepsilon N} \min \{\tau\varepsilon\alpha_i^n\} - \max_{j\varepsilon N} b^n(j)$$
$$\geq \min\{\tau\varepsilon\alpha_{i_o(n-1)}^{n-1}\} - \max_{j\varepsilon N} b^{n-1}(j) - \varepsilon^{n-1} = 0$$

as $\max_{j\varepsilon N} b^n(j)$ is always taken on $\bigcap_{\rho=1}^{n-1} \overline{N_o^\rho}$ because of

$$b^n = \sum_{\rho=1}^{n-1} \beta(N_o^\rho)\varepsilon^\rho$$ from 2.5.

<u>Lemma 3:</u> If $b\varepsilon \mathcal{B}$ then $b + \varepsilon\cdot\beta(N_o)\ \varepsilon\mathcal{B}$.

<u>Proof:</u> For $\varepsilon^n = 0$, nothing is to be shown. For $\varepsilon^n > 0$, only $i\varepsilon\overline{N_o^n}$ are

of interest. One has

$$\min_{i\in\overline{N}_o^n}\ \min_{(\alpha,i)\in A}\ \min_{t\in T_\alpha}\{l_{\alpha i}(t)+t\}\geq\min_{i\in N}\min\{\tau\in\alpha_i^n\}\qquad(4)$$

for otherwise there exists $k_o\in\overline{N}_o^n$ (which takes the minimum of (4)) and $R(p,k_o)$ with $l(R(p,k_o))=t_o<\min_{i\in N}\min\{\tau\in\alpha_i^n\}$. But then there exists a node j_o with $A(R(p,k_o))=\{A(R(p,j_o)),(j_o,k_o)\}$. $j_o\in N_o^n$ forces $t_o\geq\min_{i\in N}\min\{\tau\in\alpha_i^n\}$, a contradiction, but $j_o\in\overline{N}_o^n$ forces $t_o=l(R(p,j_o))$ because of $l_{ij}(t)\geq 0$ and the minimality of k_o and further backtracking on $R(p,j_o)$ must give a node $\rho_o\in N_o^n$ (because of $p\in N_o^n$), a contradiction. Thus, (4) is valid and

$$\begin{aligned}\varepsilon^n&=\min_{i\in N}\min\{\tau\in\alpha_i^n\}-\max_{j\in N}b^n(j)\\&\leq\min_{i\in\overline{N}_o^n}\ \min_{(\alpha,i)\in A}\ \min_{t\in T_\alpha}\{l_{\alpha i}(t)+t\}-\max_{j\in N}b^n(j)\\&\leq\min_{(\alpha,i)\in A}\ \min_{t\in T_\alpha}\{l_{\alpha i}(t)+t\}-b^n(i),\ \forall i\in\overline{N}_o^n\end{aligned}$$

When performing the described algorithm one can get the following possibilities (for the n-th iteration) :

$$\varepsilon^n>0,\ \#N_o^{n+1}\ \#N_o^n+1\qquad(5)$$

$$\varepsilon^n>0,\ \#N_o^{n+1}=\#N_o^n\qquad(6)$$

$$\varepsilon^n=0,\ \#N_o^{n+1}=\#N_o^n+1\qquad(7)$$

$$\varepsilon^n=0,\ \#N_o^{n+1}=\#N_o^n\qquad(8)$$

In view to the criteria 2.6 and 2.8 the best which can happen is (5), the worst (8).

<u>Lemma 4:</u> If (8) occurs at iteration n there is $k(n)\leq\#N_o^n-1$ so that for iteration $n+k(n)$ (5), (6) or (7) must hold.

<u>Proof:</u> If $\varepsilon^n=0$ and $i_o(n)\in N_o^n$ (which means $N_o^{n+1}=N_o^n$) there exists an integer ν_n with $1\leq\nu_n\leq\#N_o^n-1$ which is maximal with $i_o(n-\mu)\in N_o^n$ and $\max_{j\in N}b^n(j)\in\gamma_{i_o(n-\mu)}$, $\mu=1,\dots,\nu_n$.

$$i_o(n-\mu_1)\neq i_o(n-\mu_2),\ 0\leq\mu_1<\mu_2\leq\nu_n\qquad(9)$$

because of 2.3 and there must exist an integer $k_n\geq 1$ that if $\varepsilon^{n+\varkappa}=0$ and $i_o(n+\varkappa)\in N_o^{n+\varkappa}=N_o^n$, $\varkappa=0,1,\dots,k_n-1$

$$N_o^{n+k_n-1} \setminus \bigcup_{\rho=-\nu_n}^{k_n-1} \{i_o(n+\rho)\} = \emptyset$$

(as $i_o(n+\rho_1) \neq i_o(n+\rho_2)$, $-\nu_n \leq \rho_1 < \rho_2 \leq k_n-1$, which follows from (9) by induction on n) and

$$\min_{i\in N_o^{n+k_n-1}} \min\{\tau\epsilon\alpha_i^{n+k_n}\} > \max_{j\in N} b^{n+k_n}(j) = \max_{j\in N} b^n(j)$$

must hold. Thus, $k(n) \leq k_n \leq \# N_o^{n+k_n-1} - 1 = \# N_o^n - 1$.

Because of (6), a restriction to integer values [2] is necessary to ensure the finitness of the algorithm (of course, taking into account additional difficulties (when determining inf/sup in 2.4) and restrictions on the family of functions $l_{ij}(t)$ (to ensure convergency of the algorithm) a continuous version of the problem is possible (see [11])). In [11] the lengths of the arcs are of the special form

$$l_{ij}(t) = \begin{cases} \infty & t\epsilon V \\ l_{ij} & \text{otherwise} \end{cases}$$

where V describes times of forbidden movement in arc (i,j) which does not take into consideration (as it is done here) that a later departure from i may yield an earlier arrival at j via arc (i,j).
If no parking constraints (in the nodes) are considered a simplification, which yields elementary optimum solutions by waiting in the nodes as long as the most convenient departure time is at hand, is possible. This simplified situation is considered in [17], but to maintain feasibility (for the dual problem), the correct formula for ε would be (in notations of [17])

$$\varepsilon = \min_{x\epsilon S, y\epsilon T} \min_{\theta} \{\gamma(x,y,\mu(x) + \theta) + \theta + \mu(x) - \mu(y)\}$$

In 2.2 and 2.10 the nodes are assigned predecessor nodes to lighten the backtracking process of an optimum route. An example how the algorithm works is available in [10], a computer program is under preparation.

---

[2] This is, for computational convenience, assumed in the formulation of the problem.

References:

[1] Bellman, R.: On a Routing Problem. Quart. Appl. Math. 16 (1958) 87-90.

[2] Bellmore, M. and G.L. Nemhauser: The Traveling Salesman Problem: A Survey. J. ORSA 16 (1968) 538-558.

[3] Caldwell, T.: On Finding Minimum Routes in a Network with Turn Penalties. Comm. ACM 4 (1961) 107-108.

[4] Cooke, K.L. and E. Halsey: The Shortest Route through a Network with Time-Dependent Internodal Transit Times. J. Math. Anal. and Appl. 14 (1966) 493-498.

[5] Dijkstra, E.W.: A Note on Two Problems in Connexion with Graphs. Num. Math. 1 (1959) 269-271.

[6] Dreyfus, S.E.: An Appraisal of Some Shortest-Path Algorithms. J. ORSA 17 (1969) 395-412.

[7] Ford, L.R. and D.R. Fulkerson: Constructing Maximal Dynamic Flows from Static Flows. J. ORSA 6 (1958) 419-433.

[8] Ford, L.R. and D.R. Fulkerson: Flows in Networks. Princeton University Press, Princeton, N.Y. (1962).

[9] Gaul, W.: Über Flußprobleme in Netzwerken. to appear in ZAMM (1975).

[10] Gaul, W.: Optimale Wege bei zeitabhängigen Nebenbedingungen. working paper (1975).

[11] Halpern, J. and I. Priess: Shortest Path with Time Constraints on Movement and Parking. Networks 4 (1974) 241-253.

[12] Hardgrave, W.W. and G.L. Nemhauser: On the Relation between the Traveling Salesman and the Longest-Path Problems. J. ORSA 10 (1962) 647-657.

[13] Helbig Hansen, K. and J. Krarup: Improvement of the Held-Karp Algorithm for the Symmetric Traveling Salesman Problem. Math. Prog. 7 (1974) 87-96.

[14] Hu, T.C.: Integer Programming and Network Flows. Addison-Wesley, Reading, Mass. (1969).

[15] Ibaraki,T.: Algorithms for Obtaining Shortest Paths Visiting Specified Nodes. Siam Review 15 (1973) 309-317.

[16] Kirby, R.F. and R.B. Potts: The Minimum Route Problem for Networks with Turn Penalties and Prohibitions. Transp. Research 3 (1969) 397-408.

[17] Klafszky, E.: Determination of Shortest Path in a Network with Time-Dependent Edge-Lengths. Math. Operationsforsch. u. Statist. 3 (1972) 255-257.

[18] Lawler, E.L.: A Procedure for Computing the k Best Solutions to Discrete Optimization Problems and its Application to the Shortest Path Problem. Manag. Sc. 18 (1972) 401-405.

[19] Pierce, A.R.: Bibliography on Algorithms for Shortest Path, Shortest Spanning Tree, and Related Circuit Routing Problems (1956-1974). Networks 5 (1975) 129-149.

[20] Thompson, G.L.: Algorithmic and Computational Methods for Solving Symmetric and Asymmetric Travelling Salesman Problems. presented at'Workshop on Integer Programming', Bonn, Sept. (1975).

Wolfgang Gaul

Inst. Angew. Mathematik
u. Informatik
der Universität Bonn

53 BONN
Wegelerstr. 6