THE BELL SYSTEM TECHNICAL JOURNAL
Vol. 56, No. 8, October 1977
Printed in U.S.A.

A Descent Algorithm for the Multihour Sizing of Traffic Networks

By W. B. ELSNER

(Manuscript received March 3, 1977)

Multihour engineering is a technique for designing trunk networks when the hours of peak traffic loads between various pairs of offices do not coincide. A new descent-type computational algorithm for the multihour engineering problem is derived. This algorithm obtains the unique solution to the minimization of the multihour cost function, which is strictly convex but only piecewise differentiable. The noninteger minimum-cost solution is subsequently rounded to the nearest allowable integer solution to give a realizable network. The new algorithm is applied to three numerical examples from the California network. The results are compared with the nonoptimal, nonunique solutions obtained with an earlier algorithm, and with the traditional single-hour solutions.

I. INTRODUCTION

This paper describes a numerical algorithm which obtains the unique, optimal noninteger solution to the multihour traffic network engineering problem. This solution is subsequently rounded to the nearest allowable integer solution to yield a unique, near-optimal realizable network.

As described in Ref. 1, multihour engineering is a procedure whereby a least-cost traffic network is engineered for more than one set of point-to-point loads, subject to the constraint that blocking on any last-choice trunk group not exceed a specified value. For networks which exhibit noncoincident traffic patterns,† the multihour engineering method has been shown to achieve significant capital-cost savings over the conventional single-hour engineering procedures.[1]

The results reported in Ref. 1 were based on an algorithm which optimizes the high-usage trunk group sizes‡ one at a time, in a fixed but

† Traffic loads between different pairs of offices are said to be noncoincident if their highest average values occur in different hours, or at the same hour but in different seasons.

‡ High-usage groups are direct groups which carry the majority of the load between those pairs of offices which have a large enough community of interest to warrant direct trunking.

arbitrary sequence, until no further cost reductions can be obtained. This algorithm is called a "coordinate-search" algorithm here. Such an algorithm has the undesirable property that it does not generally converge to a unique solution of the multihour problem. It can converge to any one of a family of suboptimal solutions, depending on the initialization of the algorithm, and on the specific order in which the calculations are performed.

The practical reasons for obtaining the optimal—and hence (as shown in Section III below) unique—solution to the multihour problem are as follows: First, the periodic re-engineering of the network in response to new load forecasts is facilitated. (In the Bell System, most networks must be re-engineered at least once each year.) A unique solution guarantees that changes in trunk-group sizes from one forecast period to the next reflect only changes in the loads. In contrast, the coordinate-search algorithm can produce changes in trunk-group sizes which are as much a function of nonuniqueness as they are of actual alterations in the loads. It is not possible to distinguish between these two effects, and thus use of the coordinate-search algorithm could lead to excessive rearrangements. Second, capital-cost savings with respect to the coordinate-search solutions can be realized in most cases.

The essential difficulty of the multihour engineering problem arises from the fact that the network cost function is not differentiable everywhere in its domain. The algorithm presented here, however, is assured of convergence to the minimum-cost noninteger solution by the convexity of the cost function and by the particular mechanism for executing the search process.

II. MULTIHOUR ENGINEERING—THE MODEL AND ITS COST FUNCTION

The model of the network considered in this paper is shown in Fig. 1. Traffic which is destined from the single originating office to one of n terminating offices is first offered to the appropriate one-way high-usage trunk group. If all the trunks in that group are busy, the traffic overflows and is offered to a final group which routes this parcel to a tandem switch, from where it is sent to its destination via a tandem-completing group. The final and tandem-completing groups are sized so that the blocking probability on each is 0.01 during its respective busy hour. The object of the engineering process is to determine the high-usage trunk group sizes which minimize the cost of the network subject to the blocking constraints on the alternate routes.

The cost of the network can be divided into four components, which are defined below:

(i) The direct-route cost: It is assumed that the cost of each high-usage trunk group is directly proportional to the number of trunks in the

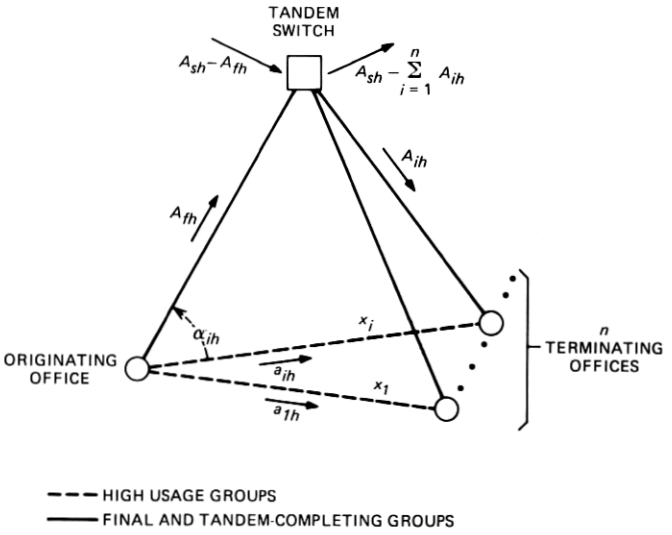

Fig. 1—Network model.

group.† If the cost per trunk of the ith high-usage group is c_{di}, and there are x_i trunks in this group, then the total cost for high-usage trunks in this network is given by

$$C_d = \sum_{i=1}^{n} c_{di} x_i. \tag{1}$$

In the theory which follows, as well as in the algorithm based on this theory, x_i is treated as a nonnegative real variable.

(*ii*) The final-route cost: Let $\mathcal{H} = \{1, 2, \ldots, H\}$ be the set of hours for which the network is to be engineered, and let $h \in \mathcal{H}$. Let a_{ih} be the load, in erlangs, offered to the ith high-usage group in hour h. Then the overflow from this group in hour h is given by

$$\alpha_{ih} = a_{ih} B(x_i, a_{ih}) \tag{2}$$

where $B(\cdot, a_{ih})$ is a strictly convex and continuously differentiable function of x_i which agrees with the Erlang loss function on the integers.‡ The overflow parcels from all the high-usage groups are combined and offered to the final trunk group. It is assumed that, in addition to the overflow traffic, the final group also has offered to it a first-routed load in hour h, designated by A_{fh}.

† Such an assumption is necessary, since the eventual realization of the network in terms of facilities is not known at the time that the groups are sized. Thus, average costs per trunk are used in approximating the eventual cost of each group.

‡ Such an interpolating function can always be constructed since $B(n-1, a) - B(n, a) > B(n, a) - B(n+1, a)$, $n = 1, 2, \ldots$.[2]

In sizing high-usage trunk groups, it is customary to approximate the number of trunks required in the final group by dividing the total load offered to this group by its so-called "marginal capacity." [1,3†] If the cost per trunk of the final group is c_f and its marginal capacity γ_f, then the cost of sizing the final to carry only its hour-h load is approximated by

$$C_{fh} = \frac{c_f}{\gamma_f}\left(A_{fh} + \sum_{i=1}^{n} \alpha_{ih}\right). \tag{3}$$

Since the final group must be engineered for its busiest hour, its approximate cost is

$$C_f = \max_{h \in \mathcal{H}} C_{fh}. \tag{4}$$

The actual sizing of the final group (which takes place only after all the high-usage groups have been sized) is done more precisely, of course.

(*iii*) The switching cost: It is assumed that the cost of switching is proportional to the load, with a unit-cost per erlang of c_s. Let A_{sh} denote the load switched by the tandem in hour h, excluding the overflows from the n high-usage groups. (The first-routed load on the final in hour h, A_{fh}, is included in A_{sh}.) Ignoring the blocking on the final group, the cost of switching only the hour-h load is

$$C_{sh} = c_s\left(A_{sh} + \sum_{i=1}^{n} \alpha_{ih}\right). \tag{5}$$

The cost of the tandem switch, when engineered for its busy hour, is then

$$C_s = \max_{h \in \mathcal{H}} C_{sh}. \tag{6}$$

(*iv*) The tandem-completing costs: The total load offered to the ith tandem-completing group in hour h is $A_{ih} + \alpha_{ih}$, where α_{ih} is the overflow from the ith high-usage group (neglecting the blocking on the final group and at the tandem) and A_{ih} is the remaining load destined to the ith terminating office via the tandem. As in the case of the final group, the size of the tandem-completing group is not computed exactly, but rather approximated by dividing its offered load by its marginal capacity. Let γ_{ti} be the marginal capacity of the ith tandem-completing group, and c_{ti} its cost per trunk. Then the cost of sizing this group to carry only its hour-h load is

† While the marginal-capacity assumption is not appropriate for determining actual trunk requirements on the final group, it is sufficiently accurate for the comparative purpose to which it is put here.

$$C_{tih} = \frac{c_{ti}}{\gamma_{ti}} (A_{ih} + \alpha_{ih}), \tag{7}$$

and sizing this group for its busy hour results in a cost

$$C_{ti} = \max_{h \in \mathcal{H}} C_{tih}. \tag{8}$$

The cost of providing trunks for all tandem-completing groups is thus

$$C_t = \sum_{i=1}^{n} C_{ti}. \tag{9}$$

The total cost of the network is simply the sum of these four components:

$$\begin{aligned} C(x) &= C_d + C_f + C_s + C_t \\ &= \sum_{i=1}^{n} c_{di} x_i + \frac{c_f}{\gamma_f} \max_{h \in \mathcal{H}} \left(A_{fh} + \sum_{i=1}^{n} \alpha_{ih} \right) \\ &\quad + c_s \max_{h \in \mathcal{H}} \left(A_{sh} + \sum_{i=1}^{n} \alpha_{ih} \right) \\ &\quad + \sum_{i=1}^{n} \frac{c_{ti}}{\gamma_{ti}} \max_{h \in \mathcal{H}} (A_{ih} + \alpha_{ih}), \end{aligned} \tag{10}$$

where $x = \{x_1, \ldots, x_n\}$ is the n-vector of high-usage group sizes. This function is called the "multihour cost function." Note that the final, switch, and tandem-completing costs may attain their maxima for different values of h, since each of these alternate-route components may be busy in a different hour.

The object of multihour engineering, then, is to minimize the cost function defined by eq. (10) with respect to the high-usage trunk group sizes, i.e., to determine $x = x^*$ such that

$$C(x^*) = \min_{x \in \mathcal{X}} C(x) \tag{11}$$

where the set $\mathcal{X}$ is defined by

$$\mathcal{X} = \{x: x_i \geq 0, \quad i = 1, \ldots, n\}. \tag{12}$$

From the point of view of the (continuous) multihour cost function, any x is "feasible" if $x \in \mathcal{X}$. Of course, an actual network is realizable only in whole trunks (or, in the presence of modular engineering rules,† in terms of whole modules of trunks). Thus, the noninteger solution x^*

† The uncertainty in load forecasts and the inherent modularity of some facilities have recently led to the engineering and administration of some networks in modules of trunks.

is subsequently rounded to an integer (or modular) solution, as discussed in Section IV.

III. MINIMIZATION OF THE MULTIHOUR COST FUNCTION

3.1 A reformulation

Equation (10) expresses the cost of the network as the sum of a linear term and $n + 2$ maxima of sets of nonlinear terms. For the analysis which follows, it is convenient to rewrite this cost function in terms of a single maximization operator.

Let $\mathcal{M}$ be a vector-valued index set with elements $\mu = (\mu_1, \mu_2, \ldots, \mu_{n+2})$. Each component of μ, in turn, is a member of the set $\mathcal{H} = \{1, 2, \ldots, H\}$, i.e., $\mathcal{M} = \{\mu = (\mu_1, \ldots, \mu_{n+2}): \mu_i \in \mathcal{H}\}$. Let $\{C_\mu(x): \mu \in \mathcal{M}\}$ be a new family of cost functions, called "elementary cost functions," which are defined by

$$C_\mu(x) \triangleq C_d(x) + C_{f\mu_{n+1}}(x) + C_{s\mu_{n+2}}(x) + \sum_{i=1}^{n} C_{ti\mu_i}(x). \tag{13}$$

In this equation, $C_{ti\mu_i}(x)$ is the cost of sizing the ith tandem completing group for its hour-μ_i load, $C_{f\mu_{n+1}}(x)$ is the cost of sizing the final for its hour-μ_{n+1} load, and $C_{s\mu_{n+2}}(x)$ is the cost of sizing the switch for its hour-μ_{n+2} load, as defined by eqs. (7), (3), and (5), respectively; the functional dependence upon the trunk-group-size vector x is explicitly indicated. It follows from eq. (10) that the multihour cost function is obtained from eq. (13) by maximizing each term on the right-hand side with respect to the appropriate component of μ:

$$C(x) = C_d(x) + \max_{\mu_{n+1} \in \mathcal{H}} C_{f\mu_{n+1}}(x) + \max_{\mu_{n+2} \in \mathcal{H}} C_{s\mu_{n+2}}(x) + \sum_{i=1}^{n} \max_{\mu_i \in \mathcal{H}} C_{ti\mu_i}(x). \tag{14}$$

This term-by-term maximization, however, is equivalent to maximizing $C_\mu(x)$ over all possible choices of μ:

$$C(x) = \max_{\mu \in \mathcal{M}} C_\mu(x). \tag{15}$$

In other words, the multihour cost function can be viewed as the pointwise maximum of the elementary cost functions defined by eq. (13). The multihour engineering problem now has the following form: Determine the vector $x^* \in \mathcal{X}$ with the property that

$$C(x^*) = \min_{x \in \mathcal{X}} \max_{\mu \in \mathrm{M}} C_\mu(x). \tag{16}$$

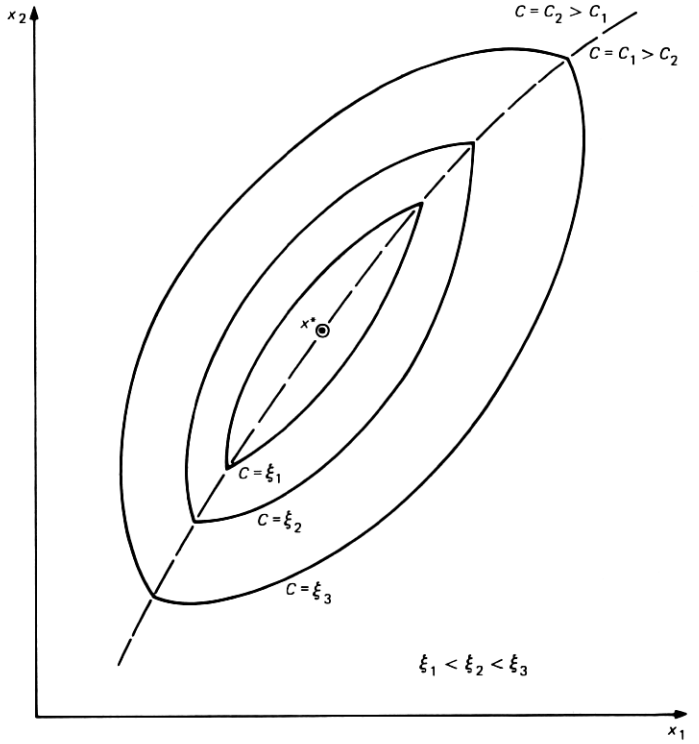

Fig. 2—Level curves of the multihour cost function.

3.2 Some properties of the multihour cost function

The multihour cost function has two fundamental properties on which the algorithm for finding its minimum is based: (*i*) it is a strictly convex function of trunk group sizes; (*ii*) it is only piecewise differentiable in these variables. Each member of the family of functions $C_\mu(x)$, $\mu \in \mathcal{M}$, is the sum of differentiable, strictly convex functions plus a linear term, and is therefore itself strictly convex and differentiable.[4] Since the multihour cost function is the pointwise maximum of a family of strictly convex functions, it is also strictly convex,[4] but not necessarily differentiable everywhere. In particular, if two or more elementary cost functions are maximal at some point (and hence their graphs intersect), the multihour cost function is generally not differentiable at that point.

Figure 2 illustrates the possible nondifferentiability of the multihour cost function for an example with two high-usage groups, and in which only two distinct elementary cost functions [denoted simply by $C_1(x)$ and $C_2(x)$] are maximal anywhere. The dashed curve separates the two regions in the $x_1 - x_2$ plane in which $C_1(x) > C_2(x)$ and $C_2(x) > C_1(x)$,

respectively. The solid lines are the level curves of $C(x)$, i.e., the loci of points for which $C(x) = \xi$, where ξ is a constant. The location of the minimum, x^*, is indicated by the circled point. Clearly, the multihour cost function is not differentiable anywhere along the dashed curve, where the graphs of the two elementary cost functions intersect.

This simple example also illustrates why a coordinate-search algorithm may fail to converge to the minimal solution. Figure 3 shows the same level curves as Fig. 2, together with three typical paths which a coordinate-search algorithm might follow: Path I (0-a-b) and Path II (0-c-d-e) start at the same initial point, but their orders of search are reversed. Path III (0′-f) starts with a different initial solution. Note that the three paths terminate at three different locations (b, e, and f), none of which is the minimal solution. In this example, the algorithm stops whenever it reaches a point x for which $C_1(x) = C_2(x)$ and at which no further decrease in the function $C(x)$ can be achieved by changing only one coordinate at a time.

3.3 A feasible search direction

The principle of the algorithm presented in this paper is to perform a sequence of searches through $\mathcal{X}$, in "feasible directions of descent." A feasible direction of descent is the direction of any vector $y(x)$ with the property that if $x \in \mathcal{X}$, there exists some $\lambda > 0$ such that $x + \lambda y \in \mathcal{X}$ and $C(x + \lambda y) < C(x)$. Whenever such a direction exists, a step size for the search is chosen to maximize the decrease of the multihour cost function in that direction, while maintaining the feasibility of the solution.

In order to determine a direction of descent, we use the concept of the one-sided directional derivative of $C(x)$ with respect to a vector $y \in \mathcal{Y}(x)$, where $\mathcal{Y}(x) = \{y \in R^n$: for $x \in \mathcal{X}$ and for some $\lambda > 0$, $x + \lambda y \in \mathcal{X}\}$. This derivative is denoted by $C'(x; y)$ and is defined as follows:

$$C'(x; y) \triangleq \lim_{\lambda \downarrow 0} \frac{C(x + \lambda y) - C(x)}{\lambda}. \tag{17}$$

$C'(x; y)$ is thus the rate of change of the function $C(x)$ in the direction of y, multiplied by $\|y\|$, where $\|\cdot\|$ is the Euclidean norm. If $C(x)$ is convex, $C'(x; y)$ exists and is a convex function of y for every x at which C is finite. If $C(x)$ is actually differentiable at x, then

$$C'(x; y) = \langle y, \nabla C(x) \rangle \tag{18}$$

where ∇ is the gradient operator and $\langle \cdot, \cdot \rangle$ denotes the scalar (or inner) product of two vectors.[4]

Substituting eq. (15) into the definition of the directional derivative

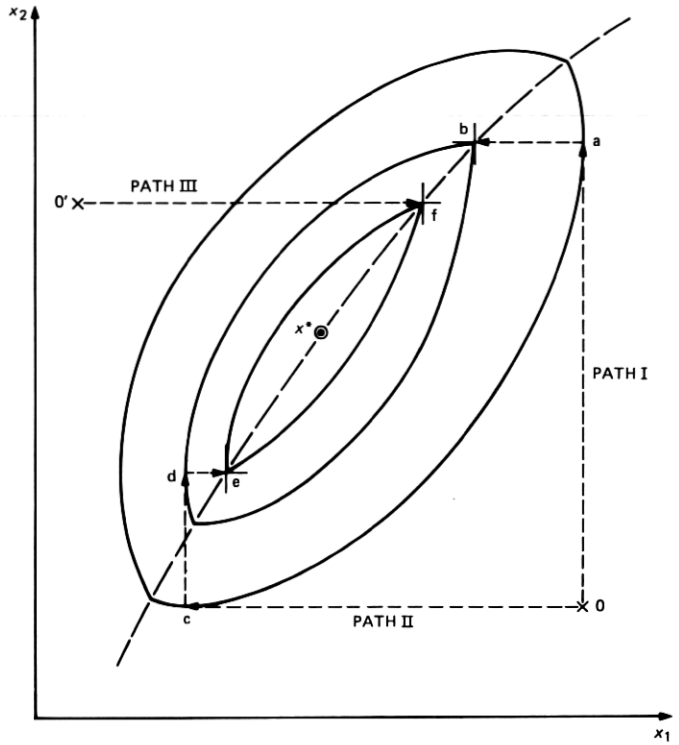

Fig. 3—Typical paths for a coordinate-search algorithm.

we have

$$C'(x;y) = \lim_{\lambda \downarrow 0} \frac{\max_{\mu \in \mathcal{M}} C_\mu(x + \lambda y) - C(x)}{\lambda}. \tag{19}$$

Let $\mathcal{I}(x) \subseteq \mathcal{M}$ be the set of indices of those elementary cost functions which are maximal at x:

$$\mathcal{I}(x) \triangleq \{\mu: C_\mu(x) = C(x)\}. \tag{20}$$

For each $\mu \in \mathcal{M}$, $C_\mu(x)$ is continuous for all $x \in \mathcal{X}$. Consequently, for each $x \in \mathcal{X}$ and for each $y \in \mathcal{Y}(x)$, there exists a $\lambda' > 0$ such that for all λ with $0 < \lambda < \lambda'$,

$$C(x + \lambda y) = \max_{\mu \in \mathcal{I}(x)} C_\mu(x + \lambda y). \tag{21}$$

In words, there exists a neighborhood of x in which no elementary cost function can be maximal which is not also maximal at x. Therefore, the maximization over $\mu \in \mathcal{M}$ in eq. (19) can be replaced by a maximization

over $\mu \in \mathcal{J} \equiv \mathcal{J}(x)$:

$$C'(x; y) = \lim_{\lambda \downarrow 0} \frac{\max_{\mu \in \mathcal{J}} C_\mu(x + \lambda y) - C(x)}{\lambda}. \tag{22}$$

Since $C_\mu(x) = C(x)$ for each $\mu \in \mathcal{J}$,

$$\begin{aligned} C'(x; y) &= \lim_{\lambda \downarrow 0} \max_{\mu \in \mathcal{J}} \left[\frac{C_\mu(x + \lambda y) - C_\mu(x)}{\lambda} \right] \\ &= \max_{\mu \in \mathcal{J}} \lim_{\lambda \downarrow 0} \left[\frac{C_\mu(x + \lambda y) - C_\mu(x)}{\lambda} \right] \\ &= \max_{\mu \in \mathcal{J}} C'_\mu(x; y), \end{aligned} \tag{23}$$

where $C'_\mu(x; y)$ is the directional derivative of $C_\mu(x)$ with respect to y. (The order of the lim and max operators can be interchanged since $\mathcal{J}$ is finite and C_μ is continuous for each $\mu \in \mathcal{J}$.) Since $C_\mu(x)$ is differentiable for each $\mu \in \mathcal{M}$,†

$$C'(x; y) = \max_{\mu \in \mathcal{J}} \langle y, \nabla C_\mu(x) \rangle. \tag{24}$$

Thus, the rate of change of $C(x)$ in the direction of y is negative (i.e., the direction of y is a feasible direction of descent) if and only if

$$\max_{\mu \in \mathcal{J}} \langle y, \nabla C_\mu(x) \rangle < 0, \quad y \in \mathcal{Y}(x) \tag{25}$$

or, equivalently,

$$\langle y, \nabla C_\mu(x) \rangle < 0, \quad \text{for all } \mu \in \mathcal{J}(x), y \in \mathcal{Y}(x). \tag{26}$$

A point at which no feasible direction of descent for $C(x)$ exists must be the location of the minimum of $C(x)$. In fact, a convex function $C(x)$ defined over a convex domain attains its global minimum at $x = x^*$ if and only if $C(x)$ is finite and

$$C'(x^*; y) \geq 0 \quad \text{for all } y \in \mathcal{Y}(x^*). \tag{27}$$

Furthermore, x^* is unique if $C(x)$ is strictly convex.[4] Since $0 \in \mathcal{Y}(x)$, eq. (27) is equivalent to

$$\min_{y \in \mathcal{Y}(x^*)} C'(x^*; y) = 0, \tag{28}$$

† If x is on the boundary of $\mathcal{X}$, $\nabla C_\mu(x)$ is defined as the limit of all sequences $\nabla C_\mu[x^{(1)}]$, $\nabla C_\mu[x^{(2)}], \ldots$, such that $x^{(i)} \in \mathcal{X}$ and $x^{(i)} \to x$.

or, with eq. (24) substituted,

$$\min_{y \in \mathcal{Y}(x^*)} \max_{\mu \in \mathcal{I}(x^*)} \langle y, \nabla C_\mu(x^*) \rangle = 0. \tag{29}$$

3.4 The descent algorithm

Inequality (25) gives the condition for y to be in a feasible direction of descent for $C(x)$. Such a y is generally not unique, and it is necessary to select a particular direction of descent at each iteration of the algorithm. A logical choice is the direction of steepest descent for $C(x)$, i.e., the vector y^* such that

$$C'(x; y^*) = \min_{y \in \mathcal{S} \cap \mathcal{Y}} C'(x; y) \tag{30}$$

where $\mathcal{S}$ is the unit sphere in R^n:

$$\mathcal{S} = \{y: \|y\| \leq 1\}. \tag{31}$$

A solution for y^* can be obtained in explicit form, as shown in the Appendix, provided $\mathcal{I}(x)$ contains either one or two elements, and $\mathcal{Y}(x) = R^n$ (i.e., x is not on the boundary of $\mathcal{X}$).

While it is possible, at least in principle, to solve for the steepest-descent vector in the general case (see the Appendix), the computation is cumbersome for three or more elements in $\mathcal{I}(x)$, or if boundary constraints are active. In this case it is more practical to choose a feasible search direction based on computational simplicity. For example, if $\mathcal{S}$ is chosen to be the set

$$\mathcal{S} = \{y: |y_i| \leq 1, \quad i = 1, \ldots, n\}, \tag{32}$$

the min-max problem expressed by eq. (30) can be converted into a linear program:

$$\begin{gathered} \min \sigma \\ \text{subject to} \\ \langle y, \nabla C_\mu(x) \rangle \leq \sigma \quad \text{for all } \mu \in \mathcal{I}(x) \\ |y_i| \leq 1, \quad i = 1, \ldots, n \\ y_i \geq 0 \quad \text{whenever } x_i = 0. \end{gathered} \tag{33}$$

This linear-programming problem is solved by standard methods. Although the vector y^* which solves that linear program may no longer be in the steepest-descent direction, the algorithm can still be shown to converge.†

† It can be shown that the algorithm will converge with y^* chosen according to eq. (30) as long as $\mathcal{S}$ is any convex, compact subset of R^n containing the origin in its interior.[5]

In principle, the descent algorithm consists of alternately computing a search direction according to eq. (30) and performing a one-parameter search to locate the minimum of $C(x)$ along that direction. (Each such combination is called an iteration.) While this procedure results in a sequence of feasible solutions with strictly decreasing costs, there is still no guarantee that this sequence will converge to the minimal solution. It is possible, as the result of a phenomenon known as "jamming" or "zig-zagging," for the sequence of solutions to converge to a point which does not satisfy eq. (29).[6]

The device used here to prevent jamming (and thus assure convergence to the minimal solution) is similar to that used by Zoutendijk.[6] This device consists of expanding the set $\mathcal{J}(x)$ to include all those elementary cost functions which are "nearly" maximal at x. Let $\epsilon > 0$ and define the new index set

$$\mathcal{J}_\epsilon(x) \triangleq \{\mu: C(x) - C_\mu(x) \leq \epsilon\}. \tag{34}$$

The direction-finding subproblem thus takes into account the directional derivatives of all those elementary cost functions which are within ϵ of being maximal at x. Similarly, the feasibility conditions are modified to prevent the algorithm from attempting to reduce any further those trunk-group sizes which are already "nearly" equal to zero. To this end, let $\delta > 0$ and define the set

$$\mathcal{Y}_\delta(x) \triangleq \{y \in \mathcal{Y}(x): \quad y_i \geq 0 \quad \text{whenever} \quad x_i \leq \delta\}. \tag{35}$$

For notational consistency, the original sets $\mathcal{J}(x)$ and $\mathcal{Y}(x)$ are henceforth denoted by $\mathcal{J}_0(x)$ and $\mathcal{Y}_0(x)$, respectively.

The original problem of determining a search direction y^* as expressed by eq. (30) is now replaced with the problem of finding $y = \hat{y}$ which solves the min-max problem

$$D(x) \triangleq \min_{y \in \mathcal{S} \cap \mathcal{Y}_\delta} \max_{\mu \in \mathcal{J}_\epsilon} \langle y, \nabla C_\mu(x) \rangle. \tag{36}$$

In this definition, the new symbol $D(x)$ replaces the symbol $C'(x; \hat{y})$, since the quantity which it denotes is no longer a directional derivative in the sense of eq. (17). Note, however, that if $\mathcal{J}_\epsilon(x) = \mathcal{J}_0(x)$ and $\mathcal{Y}_\delta(x) = \mathcal{Y}_0(x)$, then $\hat{y} = y^*$ and $D(x) = C'(x; y^*)$.

Since the inclusion of any additional necessary conditions may overly constrain the direction-finding subproblem, the values of ϵ and δ are reduced adaptively throughout the progress of the algorithm, so that $\epsilon \to 0$ and $\delta \to 0$. The use of this procedure also serves a computational purpose, in that ϵ and δ can be viewed as the tolerances within which elementary cost functions are deemed maximal and within which trunk-group sizes are considered to be zero, respectively.

The descent algorithm is now specified as follows:

Step 1 Let k be the iteration counter, and set $k = 0$.
Select an arbitrary initial solution $x^{(0)} \in \mathcal{X}$.
Select ϵ and δ.

Step 2 Compute $C[x^{(k)}]$.

Step 3 Determine the feasible search vector $y^{(k)} \equiv \hat{y}$ and compute $D[x^{(k)}]$.

Step 4 If $D[x^{(k)}] \leq -\epsilon$, go to Step 6.
If $-\epsilon < D[x^{(k)}] < 0$, go to Step 5.
If $D[x^{(k)}] = 0$, but $\mathcal{I}_\epsilon[x^{(k)}] \neq \mathcal{I}_0[x^{(k)}]$ or $\mathcal{Y}_\delta[x^{(k)}] \neq \mathcal{Y}_0[x^{(k)}]$, go to Step 5.
If $D[x^{(k)}] = 0$ and $\mathcal{I}_\epsilon[x^{(k)}] = \mathcal{I}_0[x^{(k)}]$ and $\mathcal{Y}_\delta[x^{(k)}] = \mathcal{Y}_0[x^{(k)}]$, stop. The solution $x^{(k)} = x^*$ has been found.

Step 5 Set $\epsilon = \epsilon/2$ and $\delta = \delta/2$; go to Step 3.

Step 6 Determine a scalar $\lambda^{(k)}$ such that

$$C[x^{(k)} + \lambda^{(k)} y^{(k)}] = \min_{\lambda \in \Lambda^{(k)}} C[x^{(k)} + \lambda y^{(k)}]$$

where

$$\Lambda^{(k)} = \{\lambda : x^{(k)} + \lambda y^{(k)} \in \mathcal{X}\}.$$

Set $x^{(k+1)} = x^{(k)} + \lambda^{(k)} y^{(k)}$, set $k = k + 1$, and go to Step 2.

The adaptive reduction of ϵ and δ is contained in Steps 4 and 5. Whenever $|D[x^{(k)}]|$ becomes sufficiently small (perhaps even zero), ϵ and δ are divided in half. If this reduction results in a decrease of the number of near-maximal elementary cost-functions or near-active boundary constraints, the direction-finding subproblem may be less constrained, and a new search direction may be found. If, on the other hand, the sets $\mathcal{I}_\epsilon(x)$ and $\mathcal{Y}_\delta(x)$ remain unaltered after ϵ and δ are divided in half, $D[x^{(k)}]$ remains unchanged as well, and the algorithm proceeds directly to Step 4.

It can be shown that this algorithm generates a sequence of solutions $\{x^{(k)}; k = 0, 1, 2, \ldots\}$ which is either finite, with its last term x^* satisfying

$$C'(x^*; y^*) = 0, \tag{37}$$

or infinite, with any accumulation point x^* satisfying eq. (37).[5†] It was shown earlier, however, that eq. (37) is a necessary and sufficient condition for x^* to be the unique, minimal noninteger solution to the multihour engineering problem.

† A practical stopping rule is suggested in Section 4.3.

IV. NUMERICAL RESULTS

4.1 The model

Three offices from the California network (Gardena, Compton, and Melrose) were engineered with the descent algorithm developed in Section III. For each office, the loads in two distinct hours (a morning hour and an evening hour) were considered. For the sake of simplicity, the loads A_{fh}, A_{sh}, and A_{ih}, $i = 1, \ldots, n$ (hereafter called "fixed loads," since they do not depend on the variables x_i) were assumed to be zero in both hours. All trunks were assumed to cost \$1000, and the switching cost was \$62/CCS.[†] The final and tandem-completing groups were assumed to have a common incremental capacity of 30 CCS/trunk. There were 37 high-usage groups in the Compton office, 43 in Gardena, and 35 in Melrose.

These three offices, together with the loads and costs, are the same as those used by M. Eisenberg;[1] they are used here again in order to allow direct comparisons with his results. While the simplifying assumptions in these cases certainly influence the numerical results, they preserve the essential properties of the multihour cost function and thus can be expected to demonstrate convergence characteristics similar to those which would occur in general.

4.2 Implementation of the descent algorithm

The assumption of zero fixed loads on the switch and on the tandem-completing groups allows the multihour cost function to be simplified considerably. Under this assumption, the busy hours on the tandem-completing groups are known *a priori*: the busy hour on the ith tandem-completing group is the same hour in which the load offered to the ith high-usage group is largest. Thus, only the final group and the switch have busy hours which may be functions of the high-usage group sizes. Furthermore, in the absence of fixed loads, the loads offered to the final group and to the switch are identical. Consequently, at most two of the elementary cost functions associated with each of these networks can ever be maximal.[‡] For the sake of notational simplicity, these two functions are denoted by $C_1(x)$ and $C_2(x)$, respectively.

The feasible search vector for each iteration was chosen by specifying the set $\mathcal{S}$ to be the unit sphere. For experimental purposes, two distinct initial feasible solutions were used for each of the three offices:

[†] 1 erlang = 36 CCS (hundred call-seconds per hour).

[‡] There are H^{n+2} elementary cost functions associated with a network with n high-usage groups which is engineered for H hours. Recent experience with more extensive data, including fixed loads, indicates that the busy hours of the tandem switch and of the tandem-completing groups are usually not affected by the sizes of the high-usage groups of the office which is being engineered.[7] Thus, one may need to consider only H elementary cost functions in a practical situation.

$$x_i^{(0)} = \max_{h \in \mathcal{H}} a_{ih}, \quad i = 1, \ldots, n \tag{38}$$

and

$$x_i^{(0)} = 0, \quad i = 1, \ldots, n. \tag{39}$$

At each iteration, the optimal step size $\lambda^{(k)}$ was determined by a simple parameter search: First, the location of the minimum of $C(x)$ along the search direction was bracketed between two points whose largest component-difference was 0.01 trunks. The minimal point was then computed more precisely, either by a quadratic approximation or by a linear interpolation, depending on whether $\mathcal{J}_\epsilon(x)$ contained one or two elements at that point. The Erlang-B function for noninteger trunk-group sizes was evaluated by an approximation due to Rapp,[8] and its partial derivatives were approximated by central differences with a step size of 0.01. The initial values for ϵ and δ were set equal to $10^{-3}C[x^{(0)}]$ and 0.1, respectively, and in all cases the algorithm was arbitrarily terminated after 25 iterations.

4.3 Convergence of the descent algorithm

The behavior of the algorithm was similar for all three of the offices and for both starting points. For each office, the final solutions obtained with each of the two starting points differed by less than 0.003 trunks on any high-usage group. Figures 4 to 7 summarize the convergence characteristics of the algorithm for the Gardena office, with the starting point given by eq. (38).

The cost of the network at each iteration, $C[x^{(k)}]$ (or simply $C^{(k)}$), as a function of the iteration number, k, is shown in Fig. 4. The cost decreased monotonically with k, and the rate of change became very small after the first few iterations (e.g., $C^{(4)} = 1.0003\ C^{(25)}$).

Figure 5 shows the magnitude of $D[x^{(k)}]$ (or simply $|D^{(k)}|$, and the value of ϵ, as functions of k. As this figure indicates, $|D^{(k)}|$, ϵ, and δ all approach zero as $k \to \infty$ (recall that $\epsilon < |D^{(k)}|$ for all $k \geq 0$, and that $\delta \sim \epsilon$). Thus, it is evident that the algorithm was converging to the minimal solution when it was terminated.

Unlike $C^{(k)}$, $|D^{(k)}|$ did not decrease monotonically. As the algorithm reduced ϵ, there was an occasional iteration ($k = 0, 2$, and 6) at which the solution point $x^{(k)}$ lay outside the region for which $|C_1^{(k)} - C_2^{(k)}| \leq \epsilon$. As a result, $\mathcal{J}_\epsilon[x^{(k)}]$ contained only one element at these iterations, and $\hat{y}$ was given by $-\nabla\{\max [C_1^{(k)}, C_2^{(k)}]\}$. (The dotted lines in Fig. 5 show the magnitudes of the gradients ∇C_1 and ∇C_2 as functions of k.) For the remaining iterations $\mathcal{J}_\epsilon[x^{(k)}]$ contained both indices, so that $\hat{y}$ and $D^{(k)}$ were computed via eqs. (61) to (63) in the Appendix. Note that since $\epsilon \to 0$ as $k \to \infty$ and $\mathcal{J}_\epsilon[x^{(k)}]$ contained both elements for all $k \geq 7$, the

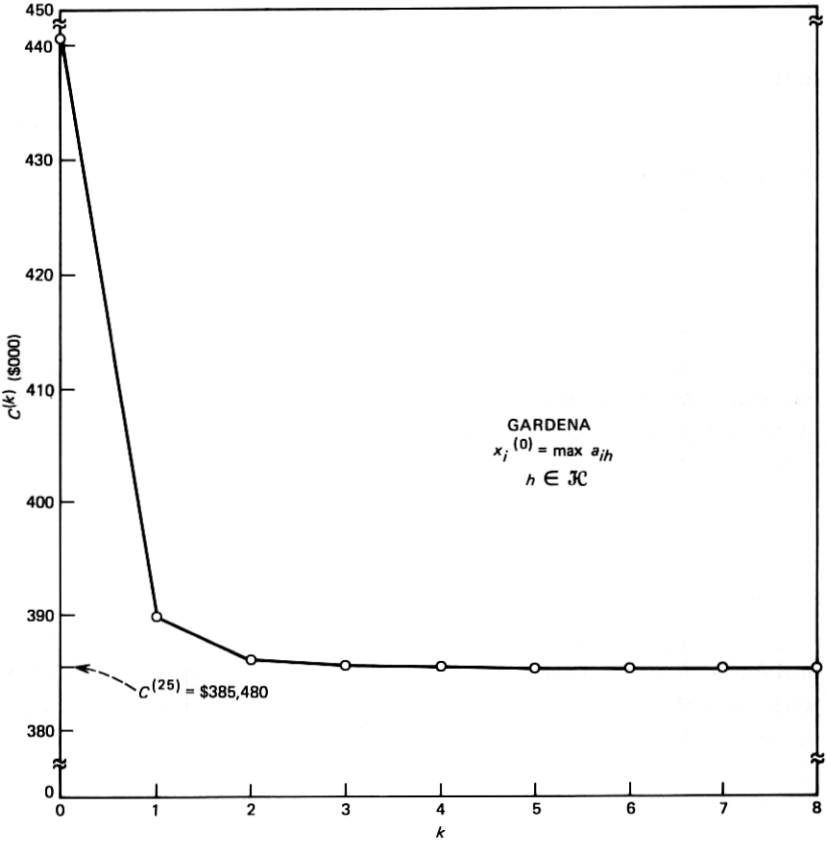

Fig. 4—Convergence of descent algorithm: network cost.

solution point x^* must be located on the intersection of the graphs of the two elementary cost functions.

The relative change in the solution from one iteration to the next, as measured by the Euclidean "distance" $\|x^{(k-1)} - x^{(k)}\|$, is shown in Fig. 6. Note that this quantity also tended to zero as k increased, although not monotonically. In particular, this plot shows that for each iteration (except the first one) at which $\mathcal{J}_\epsilon[x^{(k)}]$ contained only one element, the corresponding step size was small. Thus, the algorithm generated a sequence of solutions which tended to follow the intersection of the graphs of $C_1(x)$ and $C_2(x)$ toward the minimal solution x^*. Whenever the solution point moved too far from this intersection, a small step was taken to get back into the region defined by $|C_1(x) - C_2(x)| \leq \epsilon$, and the search along the intersection was resumed.

Figure 6 also suggests how the norm of the change in trunk-group sizes can be used as a measure for a practical stopping rule. Let Θ be a prese-

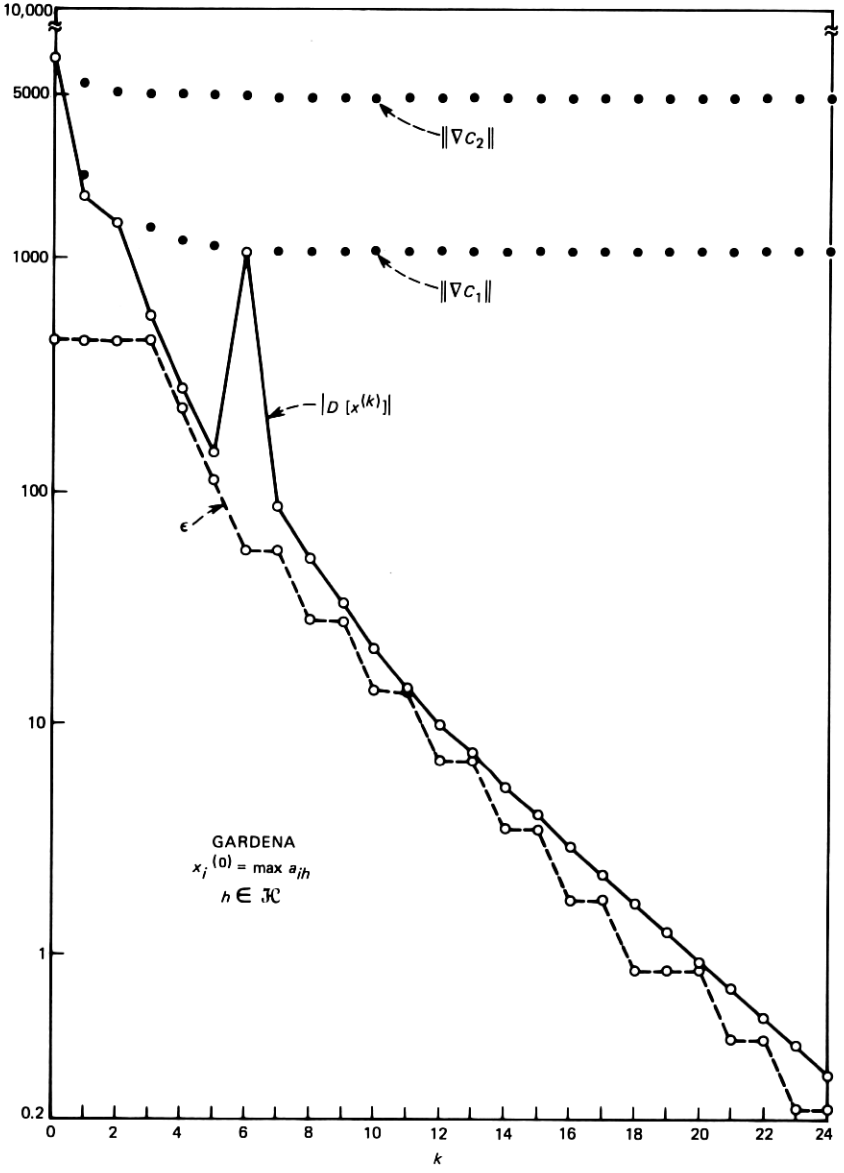

Fig. 5—Convergence of descent algorithm: $|D[x^{(k)}]|$ and ϵ.

lected threshold. Then the algorithm is deemed to have converged, and is terminated, if $k \geq 2$ and $\|x^{(k)} - x^{(k-1)}\| < \Theta$ for two consecutive iterations. The last solution $x^{(k)}$ is then an approximation to the exact solution x^*.

Figure 7 shows how the solution point $x^{(k)}$ converged. For this purpose,

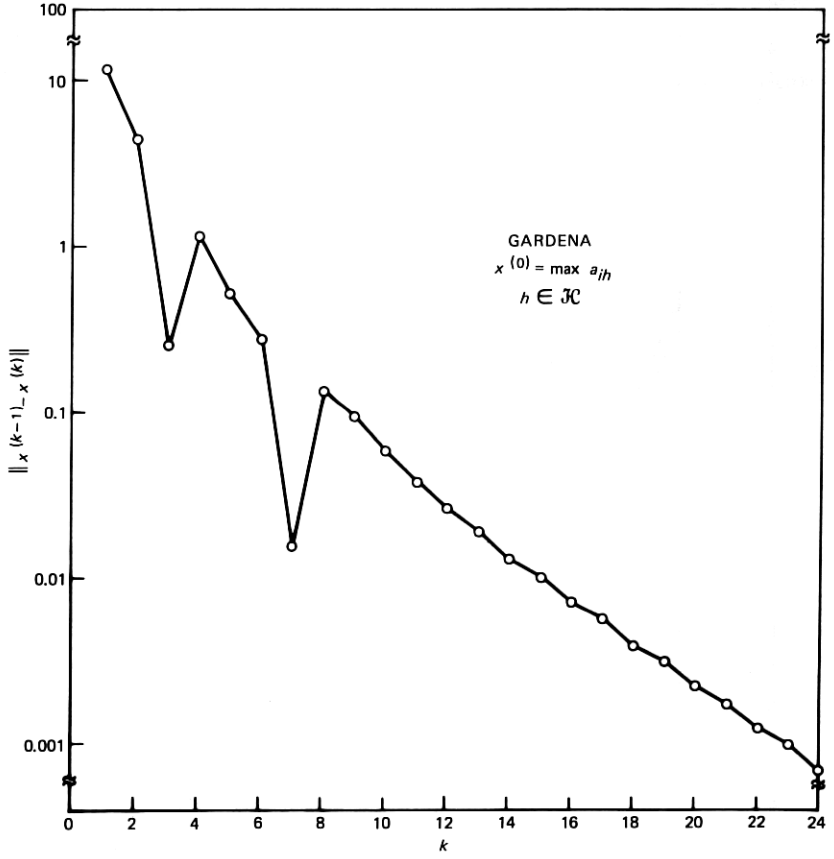

Fig. 6—Convergence of descent algorithm: change in trunk group sizes.

the Euclidean distance between $x^{(k)}$ and $x^{(25)}$, $\|x^{(k)} - x^{(25)}\|$, is plotted as a function of k. Note that $x^{(k)}$ reached a small neighborhood of $x^{(25)}$ in relatively few iterations (e.g., within one trunk after only four iterations, and within 0.1 trunks after ten iterations).

4.4 Further results

Table I shows the offered loads for the Gardena office (in CCS), and the trunk-group sizes (optimal, and rounded to the nearest integer) obtained by the descent algorithm. For the purpose of comparison, the following other sets of trunk-group sizes are included:

(*i*) For each high-usage group, the smallest and the largest number of trunks (in integers) selected from a set of solutions generated by the coordinate-search algorithm with 20 combinations of starting points and trunk-group orderings.

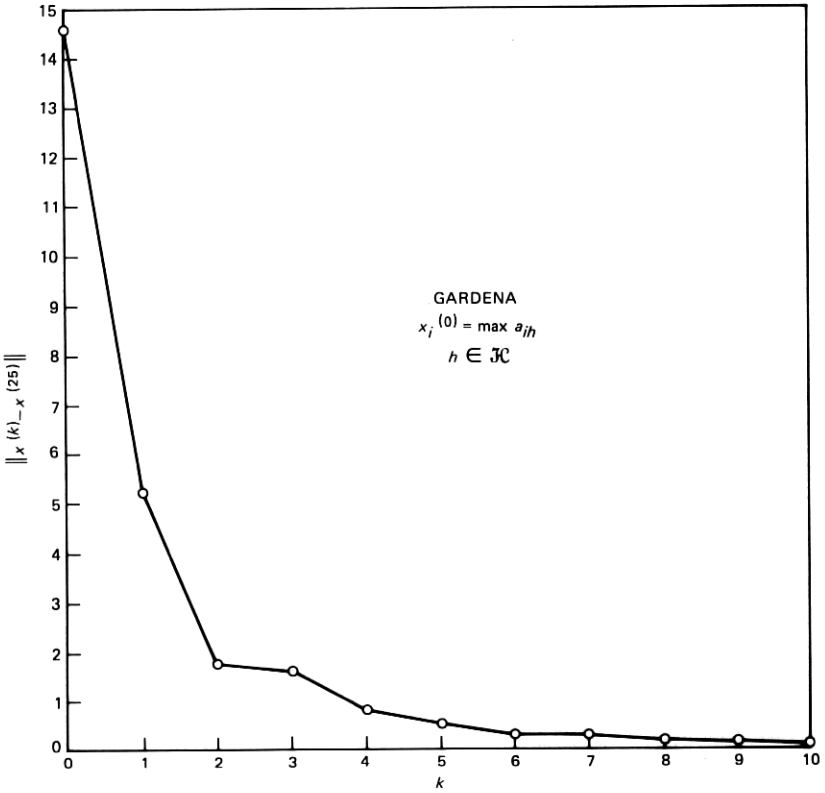

Fig. 7—Convergence of descent algorithm: trunk group sizes.

(*ii*) The number of trunks in each high-usage group (again in integers) as determined by the so-called "cluster-busy-hour" method.[1†]

This table illustrates the trunk-group-size variability when a coordinate-search algorithm is used, relative to the optimal—and hence unique—solution. Note that some groups varied by as many as eight trunks, while only six groups showed no variability. In view of the practical problems associated with such an uncertainty, as discussed in the Introduction, the need for obtaining a unique solution is clear.

Table II lists the costs[‡] of the optimal solutions for Compton, Gardena,

† Cluster-busy-hour engineering is a traditional method for sizing traffic networks, in which trunk-group sizes are chosen to minimize an elementary cost function whose alternate-route costs are all evaluated in a single, time-consistent hour. The hour selected is that hour in which the sum of the first-offered loads to all the high-usage groups which overflow to a common final group, plus the first-offered load to that final group, is largest.

‡ All costs are computed via eq. (3), and thus include the marginal-capacity approximation. The costs of comparable networks reported in Ref. 1 reflect the actual required sizes of the final groups as determined after the high-usage groups have been sized.

and Melrose. For the purpose of comparison, the table also lists the costs of the following other networks:

(*i*) A network with the optimal trunk-group sizes rounded to the nearest integers.

(*ii*) The two networks with the lowest and highest costs, respectively, selected from the set of 20 solutions generated by the coordinate-search algorithm.

(*iii*) The cluster-busy-hour network.

These results show that while some combinations of starting points and trunk-group orderings for the coordinate-search algorithm yielded solutions whose costs were only a fraction of a percent higher than the optimum, other combinations led to substantially higher costs (up to 5.5 percent in the case of Compton).

Since a network is realizable only in integer trunk-group sizes, the optimal (noninteger) solution must be rounded in some way. As indicated by the results in Table II, rounding of optimal trunk-group sizes to the nearest integers is an attractive alternative: It is simple; it yields an essentially unique solution; and, although it generally does not lead to the optimal integer solution, it yields networks whose costs are only slightly higher (from 0.2 to 0.44 percent in the three cases examined) than those of the optimal, noninteger solutions. (Subsequent studies[10] have shown that, among several practical approaches, rounding the optimal solution to the nearest integers, or to the nearest multiples of the module size, is indeed the policy most likely to minimize cost.)

The cluster-busy-hour networks are included to provide some perspective. It is evident that while the cost of any solution obtained by the coordinate-search algorithm is substantially lower than the cost of the cluster-busy-hour solution, additional nonneligibile capital savings may be obtained by computing the optimal solution via the descent algorithm.

A comparison of the rounded optimal solution with the cluster-busy-hour solution reveals that these two networks are not very different. (The average absolute difference in high-usage group sizes is only 0.8 trunks, while the average group size for the rounded optimal solution is 7.1 trunks.) The cost of the cluster-busy-hour network, however, is 11.7 percent higher than that of the integerized optimal solution. The sensitivity of the cost to relatively small changes in trunk-group sizes is a consequence of the "shape" of the multihour cost function. As Fig. 2 suggests, the contours of this function are long and narrow, and the slope is steep in directions normal to the intersection between the two elementary cost functions.

In contrast, $C(x)$ is much less sensitive to changes in x along the intersection of the graphs of $C_1(x)$ and $C_2(x)$. It is for this reason that the

Table I — High usage trunk group sizes—Gardena

Trunk Group	Offered loads (CSS)		Number of Trunks				
			Descent algorithm		Coordinate-search algorithm		Cluster-busy-hour
	Hour 1	Hour 2	Optimal	Rounded	Low	High	
1	60	140	4.42	4	4	6	3
2	119	9	5.25	5	2	6	6
3	82	20	3.78	4	2	4	4
4	305	76	11.97	12	10	12	12
5	30	0	1.47	1	1	2	2
6	59	7	2.81	3	1	3	3
7	102	56	4.64	5	5	5	5
8	256	161	10.32	10	9	11	11
9	366	230	14.10	14	12	15	15
10	469	310	17.57	18	14	18	18
11	115	115	5.37	5	5	5	5
12	144	34	6.20	6	3	7	7
13	206	335	10.81	11	10	13	9
14	310	650	18.58	19	16	22	13
15	284	319	12.14	12	12	13	12
16	93	152	5.43	5	5	6	4
17	17	24	1.08	1	1	1	1
18	74	325	8.92	9	6	13	4
19	102	158	5.74	6	5	7	5
20	137	322	9.36	9	8	13	6
21	222	247	9.78	10	10	10	9
22	252	390	12.58	13	7	15	11
23	445	194	16.73	17	12	17	17
24	176	86	7.41	7	6	8	8
25	83	29	3.83	4	2	4	4
26	98	21	4.43	4	4	5	5
27	158	74	6.75	7	6	7	7
28	124	36	5.44	5	5	6	6
29	54	25	2.64	3	2	3	3
30	38	1	1.86	2	1	2	2
31	31	17	1.60	2	1	2	2
32	140	46	6.06	6	6	6	6
33	96	30	4.35	4	3	5	5
34	122	62	5.40	5	4	6	6
35	163	57	6.92	7	5	7	7
36	163	72	6.93	7	5	7	7
37	296	238	11.84	12	11	12	12
38	33	28	1.77	2	2	2	2
39	240	3	9.70	10	6	10	10
40	136	7	5.90	6	4	6	6
41	54	4	2.59	3	2	3	3
42	52	35	2.61	3	2	3	3
43	206	9	8.48	8	3	9	9

Table II — Network costs

Office	Cost ($000)				
	Descent algorithm		Coordinate-search algorithm		Cluster-busy-hour
	Optimal solution	Integerized solution	Low	High	
Compton	402.9	403.7	406.8	425.0	488.6
Gardena	385.5	386.6	388.2	397.9	431.9
Melrose	271.7	272.9	273.7	276.0	305.3

largest cost difference between the 20 sample solutions generated by the coordinate-search algorithm is only 4.5 percent. (As suggested by Fig. 3, all termination points for the coordinate-search algorithm lie on this intersection in this particular example.)

The high sensitivity of network cost to trunk variations in some directions is not, of course, a consequence of engineering a network to a multihour (minimum-cost) solution. The cost function $C(x)$ as defined by eq. (10) represents the actual cost of the network. The multihour method is simply the one which recognizes this actual cost during the sizing process.

The imposition of modular engineering rules tends to diminish the capital savings of multihour engineering over single-hour engineering, by blurring some of the fine structure of the networks. A substantial portion of the savings can still be realized, however, as long as the module size is not large relative to the average group size. For example, in the three networks studied here, where the average group size is 7.4, rounding the high-usage group sizes to the nearest multiples of six trunks resulted in a reduction of the original savings by approximately one-fourth.

V. SUMMARY

The cost of a traffic network which gives a minimum specified grade of service on the last-choice routes in more than one hour can be approximated by a strictly convex, although possibly nondifferentiable, function of the high-usage trunk-group sizes. A descent algorithm, which can be shown to converge to the unique, noninteger minimum-cost network, has been developed. The noninteger solution is subsequently rounded to the nearest allowable integer (or modular) solution to yield a realizable network. The main advantage of this algorithm relative to the coordinate-search method is that the uniqueness of the solution prevents unnecessary, expensive rearrangements from being undertaken as traffic loads change with time. A secondary advantage is a small possible additional saving in network capital cost.

The results obtained from applying the descent algorithm to three numerical examples (and to others not discussed here) demonstrate that even after only a few iterations a sufficiently high degree of precision can be achieved to ensure the reproducibility of the results and hence the stability which motivated the design of the algorithm.

VI. ACKNOWLEDGMENTS

The author wishes to acknowledge helpful discussions with E. J. Messerli and R. Saigal.

APPENDIX

The steepest-descent direction for the multihour cost function

Consider the problem of finding a vector $y^* \in R^n$ with the property that

$$C'(x, y^*) = \min_{\|y\| \le 1} C'(x; y)$$

$$= \min_{\|y\| \le 1} \max_{\mu \in \mathcal{J}(x)} \langle y, \nabla C_\mu(x) \rangle \tag{40}$$

where $C'(\cdot\,;\cdot)$, $C_\mu(x)$ and $\mathcal{J}(x)$ have all been defined in Section III. Without loss of generality, let $\mathcal{J}(x) = \{1, \ldots, m\}$ and let Z be the convex hull of the set of all the inner products $\langle y, \nabla C_j(x) \rangle$, $j = 1, \ldots, m$:

$$Z = \left\{ z : z = \sum_{j=1}^{m} \lambda_j \langle y, \nabla C_j(x) \rangle, \sum_{j=1}^{m} \lambda_j = 1, \quad \lambda_j \ge 0 \right\}. \tag{41}$$

In other words, Z is the shortest closed segment of the real line which contains all the inner products $\langle y, \nabla C_j(x) \rangle$, $j = 1, \ldots, m$. Consequently, the maximal inner product is also the maximal element in Z:

$$\max_{j \in \mathcal{J}(x)} \langle y, \nabla C_j(x) \rangle = \max_{z \in Z} z. \tag{42}$$

Since the inner product is a linear operator, the elements z defined by eq. (41) can be rewritten as

$$z = \left\langle y, \sum_{j=1}^{m} \lambda_j \nabla C_j(x) \right\rangle, \sum_{j=1}^{m} \lambda_j = 1, \quad \lambda_j \ge 0. \tag{43}$$

Now define a new set, $\mathcal{G}$, as the convex hull of the gradients of the elementary cost functions:

$$\mathcal{G} = \left\{ g : g = \sum_{j=1}^{m} \lambda_j \nabla C_j(x), \sum_{j=1}^{m} \lambda_j = 1, \quad \lambda_j \ge 0 \right\}. \tag{44}$$

The relationship between the sets $\mathcal{G}$ and Z is evidently

$$Z = \{ z : z = \langle y, g \rangle, g \in \mathcal{G} \} \tag{45}$$

and thus

$$\max_{z \in Z} = \max_{g \in \mathcal{G}} \langle y, g \rangle \tag{46}$$

Combining eqs. (40), (42), and (46) yields

$$C'(x; y^*) = \min_{\|y\| \le 1} \max_{g \in \mathcal{G}} \langle y, g \rangle. \tag{47}$$

Since the two constraint sets on the right-hand side of eq. (47) are convex and compact, the minimization and maximization operations can be interchanged:[9]

$$C'(x; y^*) = \max_{g \in \mathcal{G}} \min_{\|y\| \le 1} \langle y, g \rangle. \tag{48}$$

For any $g \in \mathcal{G}$ with $g \neq 0$,

$$\begin{aligned} \min_{\|y\| \le 1} \langle y, g \rangle &= \left\langle -\frac{g}{\|g\|}, g \right\rangle \\ &= -\|g\|. \end{aligned} \tag{49}$$

If $g = 0$, then

$$\min_{\|y\| \le 1} \langle y, g \rangle = 0. \tag{50}$$

Equation (48) is then equivalent to

$$\begin{aligned} C'(x; y^*) &= \max_{g \in \mathcal{G}} (-\|g\|) \\ &= -\min_{g \in \mathcal{G}} \|g\| \end{aligned} \tag{51}$$

Let g^* be the vector with minimum norm in $\mathcal{G}$, i.e.,

$$\|g^*\| = \min_{g \in \mathcal{G}} \|g\|. \tag{52}$$

The desired result is now given by

$$C'(x; y^*) = -\|g^*\| \tag{53}$$

$$y^* = \begin{cases} 0 & , g^* = 0 \\ -\dfrac{g^*}{\|g^*\|} & , g^* \neq 0. \end{cases} \tag{54}$$

The set $\mathcal{G}$ and its elements g are called the "subdifferential" and the "subgradients," respectively, of the convex function $C(x)$.[4] The steepest-descent vector y^* is then in the negative direction of the minimum-norm subgradient of $C(x)$. Note, incidentally, that $C'(x^*; y^*) = 0$—the necessary and sufficient condition for $C(x^*)$ to be the minimum—is equivalent to the condition that $0 \in \mathcal{G}$ at x^*.

Explicit solutions for y^* can now be found for the cases $m = 1$ and $m = 2$, as follows:

(*i*) $m = 1$:

In this case we simply have

$$g = \nabla C_1(x) = \nabla C(x) = g^*. \tag{55}$$

(*ii*) $m = 2$:

The subdifferential is now given by

$$\mathcal{G} = \{g: g = \beta \nabla C_1(x) + (1 - \beta)\nabla C_2(x), \quad 0 \le \beta \le 1\}. \tag{56}$$

If $0 \in \mathcal{G}$, then $y^* = g^* = 0$. Suppose now that $0 \notin \mathcal{G}$. The unconstrained minimum of $\|g\|$ is found by setting its derivative (with respect to β) equal to zero:

$$\frac{d}{d\beta} \|g\| = 0. \tag{57}$$

Since

$$\|g\|^2 = \langle g, g \rangle, \tag{58}$$

the relationship

$$2\|g\| \frac{d}{d\beta} \|g\| = \frac{d}{d\beta} \langle g, g \rangle \tag{59}$$

is obtained. Thus, since $\|g\| \neq 0$, eq. (57) is equivalent to

$$\frac{d}{d\beta} \langle g, g \rangle = 0. \tag{60}$$

Let $\beta = \hat{\beta}$ satisfy eq. (60). Expanding the inner product, taking the derivative, and solving for $\hat{\beta}$ yields

$$\hat{\beta} = \frac{\|\nabla C_2\| - \langle \nabla C_1, \nabla C_2 \rangle}{\|\nabla C_1\|^2 + \|\nabla C_2\|^2 - 2\langle \nabla C_1, \nabla C_2 \rangle}. \tag{61}$$

However, since β is constrained by $0 \le \beta \le 1$, the minimum-norm subgradient g^* is given by

$$g^* = \beta^* \nabla C_1 + (1 - \beta^*)\nabla C_2 \tag{62}$$

where

$$\beta^* = \begin{cases} 0, & \hat{\beta} < 0 \\ \hat{\beta}, & 0 \le \hat{\beta} \le 1 \\ 1, & \hat{\beta} > 1 \end{cases} \tag{63}$$

REFERENCES

1. M. Eisenberg, "Engineering Traffic Networks for More Than One Busy Hour," B.S.T.J., *56*, No. 1 (January 1977), pp. 1–20.

2. E. J. Messerli, "Proof of a Convexity Property of the Erlang-B Formula," B.S.T.J., *51*, No. 4 (April 1972), pp. 951–953.
3. C. J. Truitt, "Traffic Engineering Techniques for Determining Trunk Requirements in Alternate Routing Trunk Networks," B.S.T.J., *39*, No. 2 (March 1954), pp. 277–302.
4. R. T. Rockafellar, *Convex Analysis*, Princeton, New Jersey: Princeton University Press, 1970.
5. M. D. Cannon, C. D. Cullum, Jr., and E. Polak, *Theory of Optimal Control and Mathematical Programming*, New York: McGraw-Hill, 1970.
6. G. Zoutendijk, *Methods of Feasible Directions*, Amsterdam: Elsevier, 1960.
7. M. Eisenberg, "Multihour Engineering in Alternate-Route Networks," Eighth International Teletraffic Conference, Melbourne, Australia, November 10–17, 1976.
8. Y. Rapp, "Planning of Junction Network in a Multiexchange Area," (Part I), Ericsson Technics, *20*, No. 1 (1964), pp. 77–130.
9. D. G. Luenberger, *Optimization by Vector Space Methods*, New York: Wiley, 1969, p. 208.
10. W. B. Elsner, unpublished work.