Discussion Paper No. 646

COMPREHENSIVE DESIGN OF HIGHWAY NETWORKS

by

Dan Trietsch*

March 1985

---

*Visiting Assistant Professor, Department of Managerial Economics and Decision Sciences, J. L. Kellogg Graduate School of Management, Northwestern University, 2001 Sheridan Road, Evanston, Illinois 60201.

This paper reflects some results of the author's Ph.D. dissertation (1983, in Hebrew), written under the supervision of Dr. Gabriel Y. Handler, Faculty of Management, Tel Aviv University. The author is very grateful to Dr. Handler for his help.

Preface

No, this is not the first paper on the design of highway networks--not by far. Actually, there is a good 1974 book by Steenbrink [26], devoted to this subject and inlcuding a tentative implementation to the integral Dutch highway system. Two questions arise: (a) If planning methods exist, why do so many highway networks continue to "plan themselves"? (b) What is the inherent reason that the good people of the Netherlands could assume a leading role in the effort to implement a satisficing highway network design procedure? In the opinion of this author, the answer to both questions is basically the same: the procedures for the design of a single highway published to date are not yet adapted for fast computerized preliminary design, and all the models which assume given costs are doomed before they even start. That is, of course, if you do not happen to have a very flat country with very high transportation demands. If you do, you can aproximate projected highway costs rather accurately and you also have the necessary motivation to solve the problem.

In order to appreciate both the need for exact cost evaluations for single highways, and the justification for heuristic network design procedures, we refer to a paper by Pearman [24]. According to Pearman the network design problem is rich in good suboptimal solutions. The conclusion is that heuristics are apt to do very well, and in view of the complexity of the problem, which is one of the toughest NP-complete problems (see [17]), heuristics are our only choice here, at least for medium-size problems and up (see also [21]). What does that have to do with the need for exact cost estimation? Obviously if many different network configurations yield solutions with nearly equal values, as suggested by Pearman, then the solution must be highly sensitive to the input data! We may conclude that this is the

most important practical issue to solve. However, there is another question which needs to be asked: Why is the issue of optimally locating road junctions so neglected? The answer may be that even in the Euclidean case (i.e., flat country and no constraints), and with fixed flow-independent costs, this problem is very tough mathematically (see [22, 13, 6, 10, 29]), and it seems that the possible savings are not more than 2.5%, on average (see [19]). It has been conjectured that there is an upper bound of 13.4% on these savings (see [34]). However, 2.5% may represent an impressive amount of money here, and it seems that better planned junctions may have other transportational advantages as well. Hence, this issue should be addressed as well. Note, however, that if we pick up this particular glove, our candidate highways set is no longer even countable, not to mention finite.

It is the purpose of this paper to address the problem of a highway network design for any terrain, including treatment of the junctions location issue, by a method which can be computerized and solved at reasonable cost (i.e., the computer runs should not cost more than, say, 10% of the design budget). This approach would not be feasible without very fast mainframe computers (2 MIPS or so would be a rather low figure here). However, the problem certainly justifies the use of such mainframes. True, the procedure has some heuristic features, and it is therefore a satisficing solution and not an optimal one. But, since it is not very likely that we will live to see a practical and optimal solution for this problem, this can, perhaps, be tolerated.

1. Introduction

The general problem we discuss in this paper is as follows. For n given nodes (points, city centers, etc.), upon a not necessarily planar terrain, construct a highway network capable of satisfying all the bilateral transportation demands in such a manner that the construction costs (including earth moving, paving, right of way, ecological or social penalty costs, etc.), together with the projected capitalized users' costs (such as fuel, time value, accidents, mechanical wear and tear, etc.) and the capitalized projected maintenance costs, will be minimized. Extra nodes, i.e., road junctions, are to be located as beneficially as possible to that end. (We refer to these generalized Steiner points as G-Steiner points.)

Our formulation allows for constraints on the location of highways, since these can be taken care of by proper penalties. Other technical constraints can also be accommodated. A budget constraint on the actual construction and right-of-way costs (not including penalties) can be dealt with by varying the rate of return used for capitalization of future costs. This rate serves as a Lagrange multiplier.

The effect of the network design on the flow demand can be conceptually accommodated by iterative application until an economic equilibrium is achieved (i.e., a local minimum).

The problem (and the model we suggest) can be adapted to the (more important) case of improving an existing network. It should be noted that when G-Steiner points are allowed, the existing networks connection case is no longer a special case of the "regular" network design (see [28]), although without them it is so--e.g., see [7].

An important version we do not try to solve, but must mention, is that of a dynamic planning when a budget is spread over a few years, or (even worse)

if future budgets are anticiapted but not known in a deterministic sense. Since we have to settle for a satisficing solution anyway--i.e., use heuristics--this issue should be somewhere in the background when we evaluate them, however.

Much has been written about the network design problem, usually without considering G-Steiner points. A very extensive and illuminating survey of this problem (and problems to which it can be reduced directly), has been recently published by Magnanti and Wong [21], including a "three figures" references list. Magnanti and Wong describe in detail the integer programming formulation of the problem and several Bender's cuts which can be used to accelerate its solution. For small problems--up to 30 nodes and 100 arcs--they recommend solving the problem analytically. (Note that 30 nodes imply 435 possible bilaterial arcs in our case. Choosing a hundred arcs would mean, in effect, using a heuristic without admitting it!) For medium sized problems--up to 50 nodes, 150 arcs--they suggest similar methods, however, when the lower bound obtained by the Bender's cut is approached nearly enough, the solution may be considered satisficing. For large problems--50 to 100 nodes and 400 arcs, say--they recommend faster heuristics. Delete and/or add algorithms are mentioned by them in this context (and are recommended in this paper as well). Actually they say these may require excessive time, too! (This is true if we do not stick to pure deleting or pure adding strategies; pure strategies are polynomial by nature and for an important enough problem, such as ours, can be pursued until a local optimum is achieved.) Following Magnanti and Wong's paper it almost seems pointless to refer to earlier results which they cite, but we mention briefly a subset of these. Steenbrink [26] wrote a comprehensive book on the subject, where many earlier contributions are also mentioned, and includes a tentative application of a

heuristic he developed to the integral Dutch highway network. Transportation Research devoted a special issue to the problem, edited by Boyce [3]. Claus and Kleitman [5] discuss various heuristics to a similar problem where the flows are taken into account when the cost is calculated. ([7] and [26] consider flow dependent costs, too.) A budget constraint version (i.e., where we choose the best fixed cost arcs we can from a given set s.t. a budget), has been proven to be NP-complete by Johnson et al. [17], and some branch and bound analytic and heuristic procedures have been applied to it by Dionne and Florian [9].

As for the Steiner issue, it was usually dealt with in relation to minimal tree networks [22, 13, 6, 29], i.e., without flow-cost effects. A notable exception is a 1967 paper by Gilbert [12], recently followed by Trietsch and Handler [34]. Even without flow-cost considerations the problem is NP-hard [10]. The versions of the Steiner problem mentioned this far are with Euclidean distances. Other versions exist, i.e., Steiner trees (without flow-cost effects as above) embedded in graphs, introduced by Hakimi [14] and included in the NP-complete equivalence set from it's "birth" so to speak--i.e., included in Karp's seminal paper [18]. Yet another version, with rectilinear distances, has been introduced by Hanan [15] (before [14]), and may be considered as a special case of the graph version. This version, perhaps not surprisingly, is also NP-complete [11]. Magnanti and Wong [21] show that the flow dependent cost case (which they call the fixed charge case) is NP-complete, and we reiterate a similar result in this paper as well.

With or without Steiner points, our problem is difficult enough as is--if what we want is the joy of facing difficult problems. However, any model we might have is of doubtful practical value if it does not address the issue of approximating the costs associated with the highways with some accuracy. In

this paper we use an adaptation of a highway preliminary design procedure described in [32], and based on some theoretical results by Trietsch and Handler [33], for this purpose. (The procedure can also serve as a basis for more exact alignment design, as described in [31]. Similarly, with the piecewise linear network we are going to obtain here, the same methods can serve for a more exact design.) Other published results about preliminary highway design include papers by Turner [35] and Parker [23] (or, see [32] for a discussion of these).

Appendix A is a summary of the particular method we use here.

In brief, this paper suggests the use of a honeycomb grid (see Figure 1)

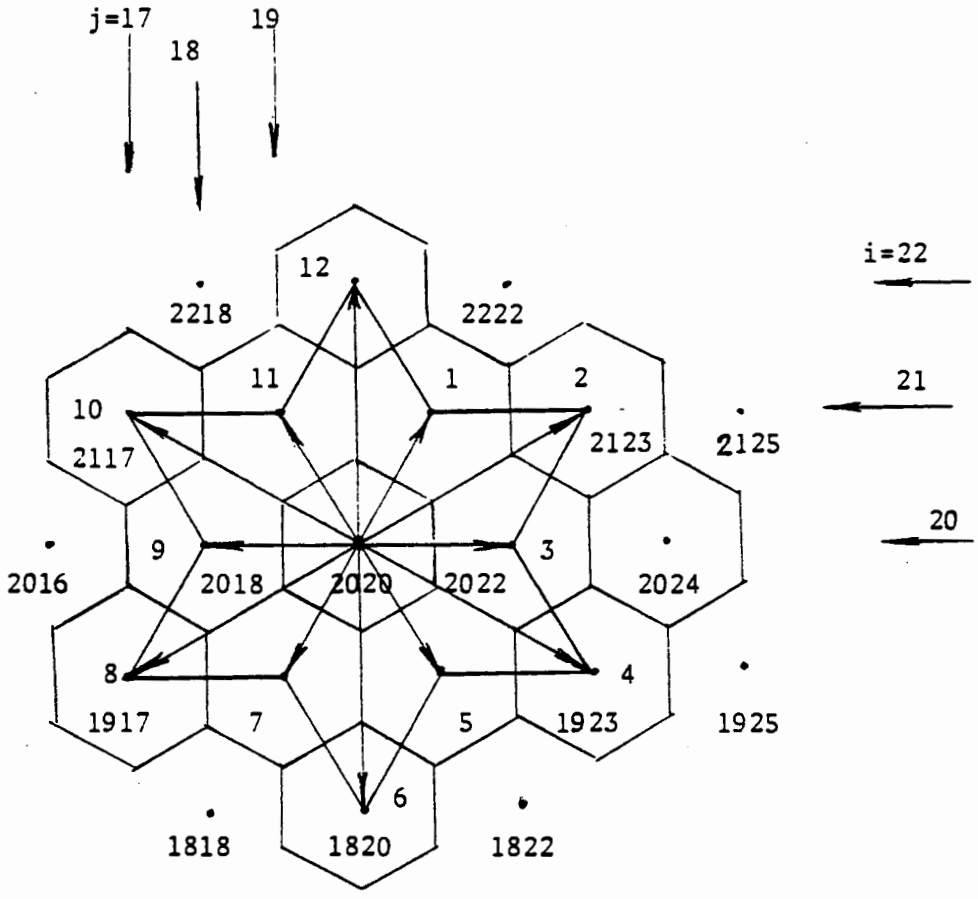

Figure 1

covering the whole area under consideration. For each hexagon we have 12 directions (which we prefer over 6, or over using a square, 8-directional grid as in [35, 23]); for each such direction we compute two parameters which serve as the intercept and the slope of a cost approximation function for a highway (there and in that direction) as affected by the flow projected on it (which, unfortunately, we cannot know in advance). Then heuristics are suggested to "extract" a "G-Steiner network" from the grid, complete with a flow assignment. This solution can then be refined iteratively by more exact highway design techniques. We also develop a lower bound on the cost of a highway which "justifies" the linear approximation, and can serve in turn for a lower bound on the value of the whole network. However, we use the linear approximation mainly for tractibility.

2. The Network Design Problem with Extra Nodes

Let N be a set of n nodes given on a surface, and $n(n-1)/2$ bilateral flow demands $q_{ij} \geq 0$ between the nodes also be given. Finally, for any pair of points X,Y (not necessarily in N) on the surface and for any $q \geq 0$, let a function $f(X,Y,q)$ be given such that

(1) $$f(X,Y,0) = 0; \ \forall\ X,Y,$$

(2) $$f(X,Y,\varepsilon) > 0; \ \forall\ \varepsilon > 0, \ X \neq Y.$$

The Problem: Choose a superset $P \supseteq N$, construct a network $G(P,A)$ (where A is the arc set, and each arc $a \in A$ connects two nodes of P), and assign all the flow demands q_{ij} to the arcs A in such a manner that the sum

(3) $$\sum_{a \in A} f(\underline{\bar{X}}^{1a}, \underline{\bar{X}}^{2a}, q_a)$$

will be minimized, where $X^{1a}, X^{2a} \in P$ are the endpoints of $a \in A$, and q_a is the sum of all the flows q_{ij} (or parts thereof) assigned to arc a.

Note that if we set $P \equiv N$, we get the regular network design problem with flow dependent costs, as described in [21], for instance. If we set

$$f(X,Y,q) = Kd(X,Y);\ \forall\ q > 0 \tag{4}$$

where d(X,Y) is the Euclidean distance between X and Y, and K is a given positive constant, we obtain the (Euclidean) Steiner tree problem. If we choose a function g(q) such that

$$g(0) = 0 \tag{5}$$

$$g(q) > 0;\ \forall\ q > 0 \tag{6}$$

$$g(q_1 + q_2) \leqslant g(q_1) + g(q_2);\ \forall\ q_1, q_2 > 0, \tag{7}$$

$$g(q + \varepsilon) \geqslant g(q);\ \forall\ \varepsilon > 0,\ q \geqslant 0, \tag{8}$$

and finally set

$$f(X,Y,q) = g(q)d(X,Y), \tag{9}$$

then we obtain a generalization of the Steiner problem first suggested by Gilbert [12], to which we refer as the Gilbert network problem, or alternatively as the generalized Euclidean Steiner network problem (GESN). (Clearly the solution should not be confined to trees here, and it is easy to construct examples with cycles; e.g., for many closely situated nodes on the circumference of a circle we would prefer the closed cycle to the regular

Steiner minimal tree.)

All the versions we mentioned are extremely complex. To complete the picture about their NP-completeness (the Steiner case is NP-hard, formally), we present the following (intuitively obvious) theorem (see also [21]).

Theorem 1: For P = N, the network design problem presented above is NP-complete.

Proof: Let $q_{in} \in \{0,1\}$; $\forall$ i < n, $q_{ij} = 0$; otherwise, and set f as in (4) with K = 1. To solve this special case of our problem we do not require cycles, and we can connect as many nodes of N by a tree as we wish, as long as we also connect all points such as i for which $q_{in} = 1$. This is an instance of the Steiner tree in a graph problem, which is (a "veteran") in the NP-complete equivalence set as proved by Karp [1]. []

(The proof cited above is due to Nimrod Meggido [private communication, 1981]. Another proof is given below, following Theorem 2, but it is irresistable to use a proof which is a direct link to Karp's original set, particularly to the Steiner tree there.)

3. Approximating the True Highway Costs

Using the method outlined in Appendix A, with a hexagonal search grid (see Figure 1), and exact vertical alignment for each arc (highway) as in [31], we can claim that we have a good approximation to f(X,Y,q); $\forall$ X,Y,q and proceed to use one of the known network design techniques. Steiner points can be inserted at the next stage if we are willing to settle on a satisficing solution.

However, even without Steiner points, due to the fact that we do not know q in advance, that might mean a lot of work. Furthermore, it is highly likely

that the honeycomb grids which we would define for each pair, would overlap to a very large extent, and it would cause much duplicated effort. This motivates us to cover the whole area by one big honeycomb grid, for all the network needs combined. Since the hexagons we use may range in diameter from, say, 250 to 2000 meters, according to our specifications, the variability of the terrain, etc., we can expect that our network will "miss" most of the n points (which we would like to be in centers of hexagons, ideally) by 125 to 1000 meters in the worst case (about 70% of that in the median case). This is acceptable indeed.

Suppose we want to cover an area of a few thousand square kilometers by a highway network--a reasonable size for a problem of this type. This calls for thousands of nodes, and about six times that many arcs. For a given flow, on a, say, 8 MIPS computer, we may expect to compute each of them during a few seconds (recall that a month has about 2.6 million seconds, if the computer operates around the clock). Alas, we do not have the flows in advance, since their allocation is part of our problem. What we can do, however, is to sample some flows for each arc (those could the same flows across the board) and regress a flow cost function for it. When performing an additional calculation, however, we can expect some relative savings if we start with a "good guess" based on former calculations. With a sample of 2-5 values, we can use a linear regression model to approximate the cost f(q) associated with arc a by a linear function with a positive intercept such as

$$f_a(q) = \ell_a + m_a q; \ \forall\ a \in A. \tag{10}$$

As an aside we mention here that devoting several days or even a few weeks mainframe computer time to a single project such as ours is very

defensible. The monetical expenditure (around $100,000 per month in 1983 prices) and the time required are both very modest considering the magnitude of the project, and the expected design budget and duration.

As an example, following the Camp David accords, Israel planned a renewed highway network in the Negev, at an estimated cost of around $1 billion--certainly an investment of less than $1 million to support the design process would be acceptable; and it could be done for a lot less than that if the software would be there. (This example also serves to show that 2-3% savings may be impressive--hence G-Steiner points are attractive.)

In [33], it has been shown that for a given width choice and vertical alignment the combined fuel and time expenditures on a highway yield a positive nondecreasing convex function. (A result well-known for time alone, but false for the fuel alone. The result is based on the assumption that the driver's speed choice implies a lower bound on his time evaluation.) Since for this case the other costs, such as construction, are virtually fixed, it follows that f(q) is such a convex function as well. However, since we are in the design stage, there is no reason to assume the same vertical alignment or width choice. It has also been shown there that the vertical alignment problem is convex (i.e., it calls for minimizing a convex functional). It stands to reason that for larger flows we would be willing to pay more in earth moving and have a highway with milder slopes; however, if and when we decide to add a lane we may revert to a steeper design, since the traffic flow per lane is reduced, and the users suffer less from congestion. A similar logic applies to the horizontal alignment issue. In both cases, if we have flat terrain, we would be able to determine the alignment independently of the flow. To sum up, the more difficult the terrain, the more we can expect the traffic flow to influence the optimal design. We should stress here that we

are talking about optimal design, and not design as per some obsolete "cook book" filled with choking standards that must be adhered to blindly. (See [2] and [33] for further independent discussions of this issue.) To continue, we have actually demonstrated that for flat terrain we can expect f to be less linear than for a hilly one, since we cannot compensate for high flows by mild slopes or curvature, etc. For this "easy-bad" case we show that a lower bound exists in the format of (10). The bound is based on the assumption that the construction costs for this flat case is also in that format (as a function of the width choice) and that by adding a lane, we increase the capacity proportionally. The gritty details are deferred to Appendix B, but the result is depicted in Figure 2 here.

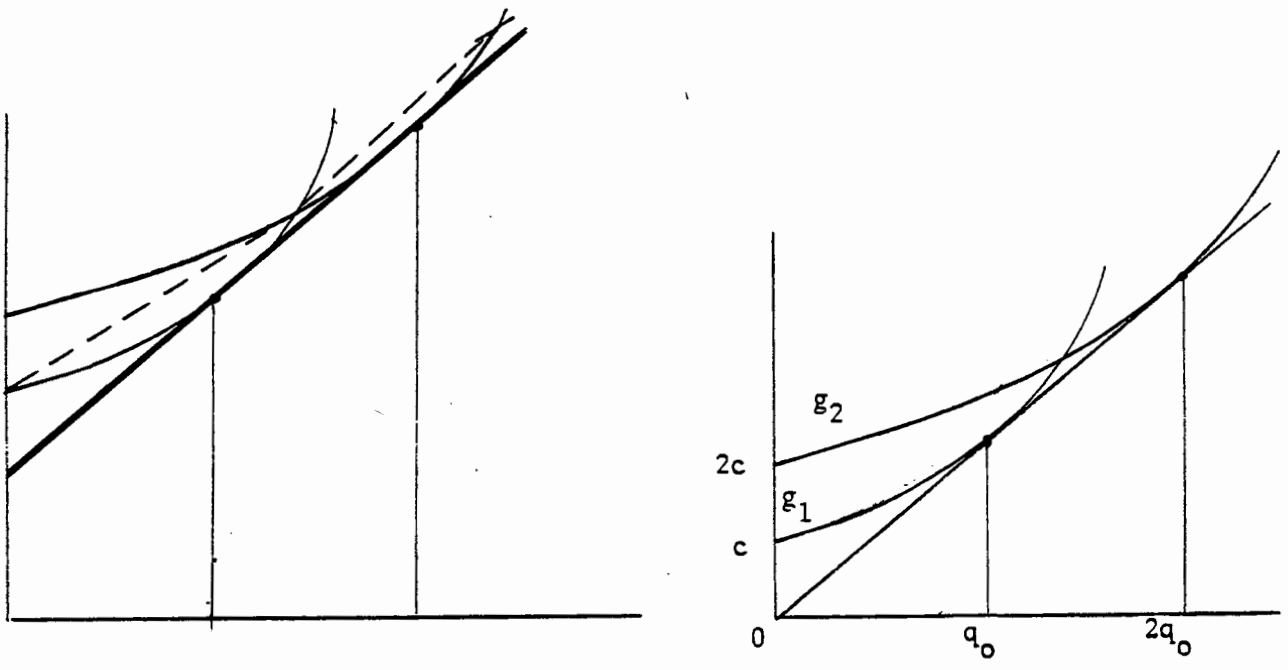

Figure 2

4. Designing a Network Embedded in a Graph

Having used a honeycomb grid with around 6 arcs per node on average, and where each arc has a flow cost function, (10), associated with it, we now wish to extract a network from it, connecting our n original nodes (which are just a small fraction of the grid's nodes), and using any additional nodes as required, not just along highways, but also as planned intersections.

This G-Steiner network embedded in a graph is an obvious generalization of the Steiner tree in a graph problem. In this paper we confine ourselves to heuristic polynomial procedures for the solution. However, we can certainly solve it by an integer programming model, so we have exponential analytic solution procedures such as those mentioned in [21], with some modifications. This is even easier if we revert to a budget constraint version, which we do not choose to do (as we are trying to solve a real problem satisficingly, rather than discuss a less realistic model which does not lend itself to an efficient solution, either).

Some Lower Bounds: If we formulate the problem as an integer programming model, we may obtain lower bounds such as those discussed in [21]. However, we prefer to discuss here some simpler version (also mentioned there in the context of transforming the network design problem to well-known simpler problems). Clearly if we set $\ell_a = 0$; $\forall\ a \in A$, all we have to do is assign all the flows to their respective shortest paths, taking the length of a path as Σm_a. This can be solved rather efficiently by well-known methods (e.g., see [8]). To this we add the minimal Steiner tree in a graph for our n nodes, this time with $m_a = 0$; $\forall\ a \in A$, taking $\Sigma \ell_a$ as our objective function. Thus we obtain a lower bound on the "pure flow" costs and another lower bound on the "fixed" costs. The sum of those is a lower bound for our problem.

If we use such values of ℓ_a and m_a which constitute a lower bound on the

real $f_a(g)$, we now have a theoretically valid lower bound, except for any errors due to our piecewise linear representation.

However, there is a slight problem: for this bound we require the use of the NP-complete Steiner tree in a graph problem. So if we want a practical lower bound we have to look further.

If we substitute the regular minimal tree (see [25]) for the Steiner tree, then our bound will be very easy to calculate, but not solid theoretically. If we take half this value, we obtain a theoretically valid bound due to a theorem in [13]. This, however, might be a too low estimate. Practically we could settle, perhaps, on a heuristic result for the Steiner tree. We describe such a heuristic in some detail here, since we are going to suggest the use of a similar one for the network later. The heuristic is a greedy one (similar to a heuristic suggested by Chang [4] for Euclidean Steiner trees), and starts with the regular minimal tree as defined for the complete shortest paths graph for our n nodes (where the distances we use are still the ℓ_a parameters, similar to the former application for m_a which we used to get the flow cost's lower bound). This tree is the one we suggested halving before, but now, if we do not propose to do that, we may (or may not) find that some arcs belong to more than one path, and were counted more than once--which should be corrected as a first step. Note now that if some arcs were indeed joint ones, our regular minimal tree would already be a (not necessarily minimal) Steiner tree! Now look at the angles formed between adjacent links in the tree (especially the more acute angles, and those formed at intersections); some immediate gain may sometimes be achieved by inserting a Steiner point of rank three (i.e., choosing a node of the honeycomb within the angle), and connecting it to the three endpoints associated with the two paths involved (two paths have 2 + 2 endpoints, but one, the apex of the

angle, is mutual). In the case of an intersection not at a node (it can occur in the 12 directional grid, or the 8 directional one for that matter, but it would not happen for the 6 directional grid, or the 4 directional one, respectively), we have to go a little further and insert two Steiner points simultaneously, but we omit this technicality in this paper. The greedy heuristic is to look for the best such gain, insert the Steiner point associated with it, and reiterate until no further immediate gain is available--thus obtaining a local minimum.

Solution Heuristics: The procedure we suggest for the G-Steiner network is to find a regular network (with unintentional G-Steiner points perhaps), and then insert G-Steiner points as described above for the tree. We begin with the first part.

One of the heuristics we suggest for the first part is similar to an algorithm suggested by Barbier [1] (or see [21]), and we refer to it as the deletion procedure. The idea is to start with the "maximal" graph, i.e., the network which makes it possible to minimize the flow costs $\Sigma_a m_a q_a$, which we obtain as described above. If we compute the $\Sigma \ell a$ (fixed) costs associated with the arcs required in our maximal graph, we have an upper bound, since this maximal graph is a feasible solution. We also obtain a lower bound (the one with the heuristic solution of the Steiner tree is suggested). Now, define an index j, initialized at j = 0,; let U^j denote the upper bound (we now have U^0); Let L denote the lower bound and proceed to step (a).

(a) Let J = j + 1, and we are in the j^{th} iteration. If $U^{j-1} - L \leq \varepsilon_1$, stop. (Note: by setting $\varepsilon_1 = 0$ we may avoid using this stopping criterion, which we can do since our procedure is inherently finite.) Else, go to (b).

(b) By a proper method (discussed separately), choose a leaving path, adjust

all the other paths to their new minimum (if they shared arcs with the leaving path), reassign all the flows which used the leaving path, calculate U^j according to the new situation, and go to (b). IF there is no candidate to leave, stop.

(c) If $U^{j-1} - U^j \leqslant \varepsilon_2$ (or $U^{j-m} - U^j \leqslant \varepsilon_2$, for any m), stop. Else go to (a). (Again, we may choose $\varepsilon_2 = 0$.)

It remains to discuss proper methods to choose a leaving path. We discuss two methods suggested by Barbier, show counterexamples to both (which is not surprising; we are disuccsing heuristics here), and proceed to suggest two more heuristics. Both are better but require more computing power, especially one of them which could not have been seriously suggested for the computers Barbier could access in 1966. We still take care to have polynomial convergence, though.

Method A: Choose the path p (connecting two nodes of N) for which $m_p q_p / \ell_p$ is minimized (where $m_p = \Sigma m_a$, and $\ell_p = \Sigma \ell_a$ over the arcs in the path), provided there is an immediate gain.

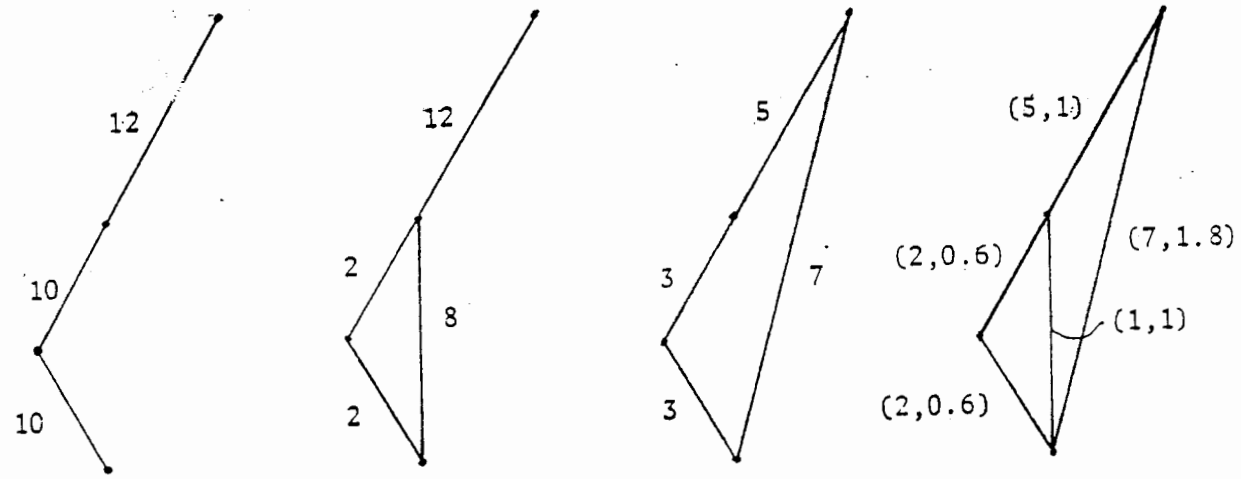

Figure 3

Counterexample A: In Figure 3 four possible topologies are depicted, where the data in parentheses in part (a) of the figure is $(q_{ij}, d(i,j))$; e.g., $q_{14} = 7$, $d(1,4) = 1.8$. Let f be as in (9), with $g(q) = 1.55 + q$; $q > 0$ (0; otherwise); e.g., if we assign a flow of 6 to arc (1,4), it would cost $(1.55 + 6) \times 1.8 = 13.59$; a flow of 7 would cost 15.39 there, etc. At the beginning we assign the flows to the direct arcs (in this example arcs and paths connecting pairs of N are identical), these being the shortest distances, to obtain $U^0 = 28.75$. (b), (c) and (d) would imply costs of 27.4, 27.36 and 27.4, respectively, so (c) is optimal; but according to Method A we have to delete arc $\overline{1,3}$, since it is the one which minimizes the ratio, and now we are stuck in (b), since there are no more candidates to leave.

Method B: To overcome that problem, Barbier suggested the following method (which would indeed work properly for the case above): delete the arc with the maximal flow first, if there is an immediate gain.

Counterexample B: As before, but $g(q) = 1 + q$, yielding 26, 25.2, 25.6 and 26.2 for (a) through (d), respectively. Now (b) is optimal, but Method B would lead us to (c)!

The following method is not much more expensive than either of the above, and it is really the least we can do.

Method C: On each step choose according to Method A or B, whichever yields a larger immediate gain.

And if the problem justifies some more expense, and ours certainly does, then Method D should perhaps be chosen.

Method D: Check all arcs, one by one, and choose the one with the best immediate gain.

In our simple example, Method C and D would outperform A or B equally well. In more complex situations D should tend to do better than C. Since we chose to delete arcs successively, without considering interim additions, the polynomial convergence of any of them is assured. Note that n is the number of concern to us here, and not the number of nodes in the honeycomb grid--although that number affects the length of the minimal path searches.

Sometimes, it should be noted, it is clear that a certain arc must be part of the solution. E.g., if the flow demand q_{ij} is high, and if the shorteset path without (i,j) is long in terms of Σm_a, i.e., if m_p is the minimal m distance between i and j when a = (i,j) is excluded, and if

$$(11) \qquad m_p\, q_{ij} \geq \ell_a + m_a q_{ij},$$

then a should be part of any optimal solution. This would tend to introduce large flow "central" arcs to the "fixed" set. On the other hand, points on the periphery, with small total flow demand may be connected through a nearest "inside" neighbor, again fixing some arcs, and changing the demand of the neighbor (in effect we can then forget these nodes). The fact that some arcs are fixed may enable us to fix other arcs in a second wave, etc. We would do that if the fixed arc is on the shortest path of the m costs for a neighboring node(s), and the additional ℓ costs required are low relative to the higher cost implied by not using the fixed arc for that flow (using the same rationale of (11)). Incidentally, such considerations can also serve with branch and bound techniques, where the inclusion of certain arcs may imply others, thus pruning the search tree a bit.

The second heuristic we consider is based on additions (versus deletions before). Now we start with a "good" spanning tree, and add arcs to it as

required. According to Dionne and Florian [9], a similar algorithm did not perform very well. However, they use the budget constraint version, and start with the regular minimal spanning tree--not necessarily the best heuristic choice for the initial solution. They also allow deletions interleaved with additions (which should improve the result, but jeopardize the polynomiality). In some more detail, starting with $j = 0$, a spanning tree of some kind (discussed further, below), with an upper bound U^0 implied by it, and L as before. Proceed to (a).

(a) Let $j = j + 1$, and we are in the j^{th} iteration. If $U^{j-1} - L \leq \varepsilon_1$, stop. Else, go to (b).

(b) Choose the best path, $\overline{i,j}$; $i,j \in N$, to improve U^j. If there is no candidate, stop. Else perform the addition, calculate U^j and go to (c).

(c) If $U^{j-1} - U^j \leq \varepsilon_2$ (or $U^{j-m} - U^j \leq \varepsilon_2$, etc.), stop. Else go to (a).

If it seems that the addition heuristic is more attractive than the deletion one (if!), it may be due to the fact that it imitates the natural "no-design" network design process, where new highways are added, hopefully as per (b), whenever a budget can be found. However, with some reflection we conclude that the deletion heuristic can be easily adapted to serve as well, and give a better picture of the "future," by going backwards from the optimal solution and continue to delete arcs according to a minimal loss criterion.

Another important issue we should discuss further is how to choose the initial tree. This problem is not trivial at all for the flow dependent costs case! E.g., we may think (not correctly perhaps), that the minimal spanning tree is a good start (as in [9]). But, although it is true that if the minimal total distance spanning tree (i.e., the regular one) is sought, then the regular algorithm suggested by Prim [25] can serve well to locate it; the

real problem of finding the minimal spanning tree with flow dependent costs is tougher, as per the next theorem.

Theorem 2: Even if we restrict the solution of the network design problem to be a tree, we still have an NP-complete problem.

Proof: Set $q_{ij} = 1$; $\forall$ i,j $\in$ N, and $\ell_a = 0$ (or any other constant); $\forall$ a, and we have an instance of the minimal total paths tree problem, which is NP-complete [17]. []

Note that if we set $\ell_a = M$, where M is large enough, then the tree restriction is satisfied automatically! Hence, this proof suffices for Theorem 1 as well.

In addition to the minimal spanning tree, other possibilities should be explored, and even if we want to confine ourselves to "polynomial" ones, our choice is rather large. The regular shortest path tree for one of the "central" points may be attractive in some situations; we may like to enter large direct flow arcs first; criteria such as m_a/ℓ_a^β could be used (where β is some positive parameter), and the minimal spanning tree for these values may serve, arcs implied to be fixed by (11) may be the basis of the tree, etc.

Since we have not yet introduced the Steiner points except, perhaps, some which made their way in "uninvited," so to speak, we should also consider the SALMOF method by Steenbrink [26], the Dantzig et al. [7] decomposition method, the method suggested by Dionne and Florian [9], after adaptation to the flow dependent cost case (though it is exponential), and so on, as elaborated by Magnanti and Wong.

Inserting Steiner Points: Briefly stated, the method described above, for the Steiner tree in a graph (in connection with the lower bound), can serve for

the network case equally easily. This can be done interactively very efficiently and though there is no reason not to do it automatically, the required programming effort may not be justified. It would be so if we compared many networks after the Steiner insertion, but so far our procedure is to first design a regular network and then insert the Steiner points. Incidentally, in the Euclidean Steiner tree case an analog procedure [4, 19], saves about 2.5% on average. In the flow weighted case there is some evidence that the savings may be slightly less over the optimal costs (see [34]), but some examples show that it will not be significantly less even if we do start with the optimal network. However, starting with the optimal network is easier said then done, and if we start with a suboptimal one, we can sometimes save more; and in the network case we may have to start with a suboptimal network by necessity, as opposed to the tree case. In addition, there are some reasons to prefer the Y junctions implied by the Steiner insertion (see Figure 4). Appendix C deals with that issue.

Final Touches: At this stage we have a piecewise linear network, with the deterministic given flow demands assigned to it. This is the time to apply such methods as described in [31] to obtain a network with smooth curvature-contraint-abiding highways. (But see [32] for a discussion of the curvature issue in the piecewise linear version as well. The solution suggested there extends directly to our case, without any problem whatsoever.) When we obtain these smooth highways, we may find that the Steiner points have to be shifted around for a while until a local optimum is obtained. However, the solution is not very sensitive to the exact location of these, since in the vicinity of the optimum the function concerned is rather flat usually. At this (more) exact design stage, we may wish to use Wardrop's second criterion, rather than the first (i.e., user optimized path choice as opposed to system optimal one,

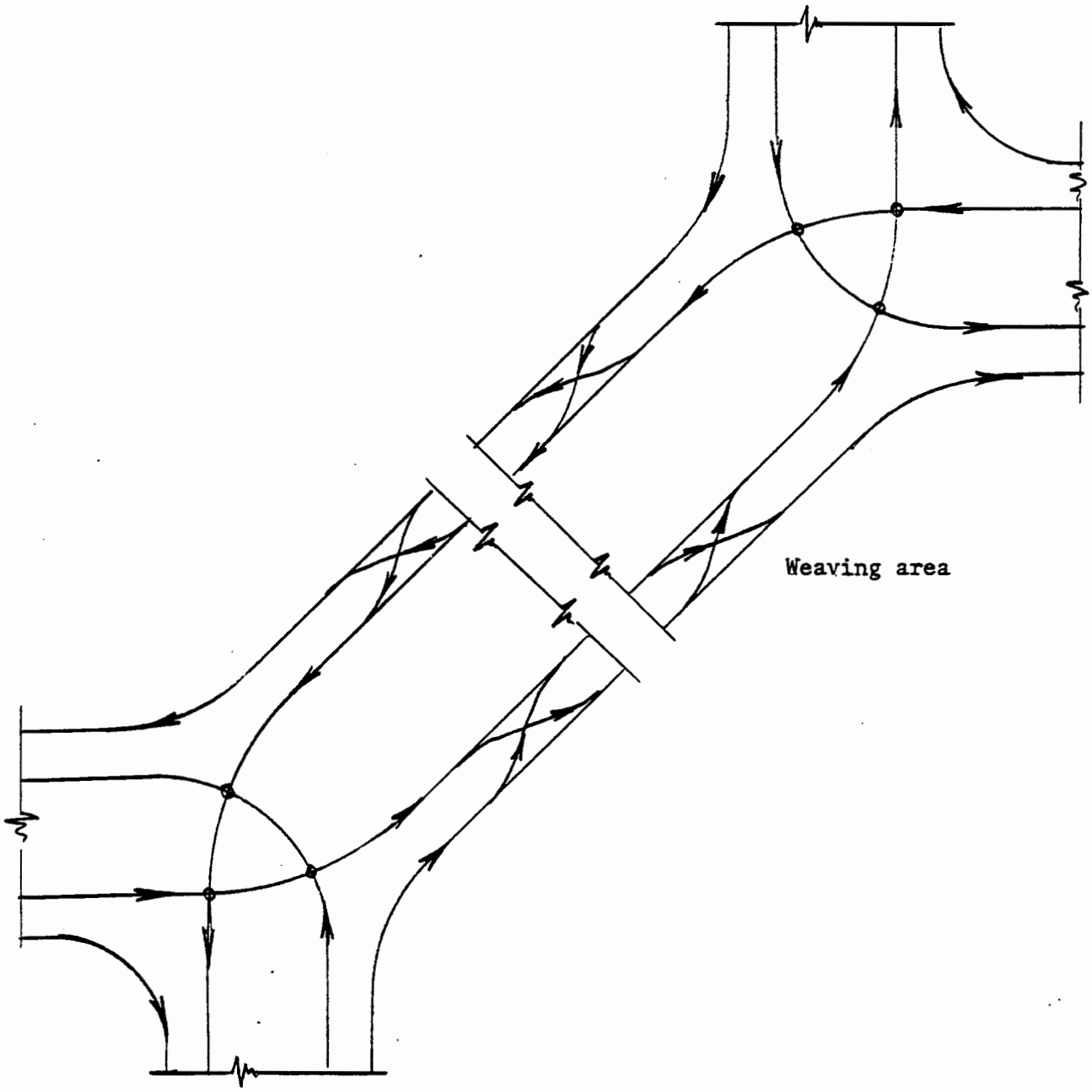

Figure 4

which may be unrealistic). The issue does not arise with linear (or concave) cost functions for the preliminary design stage, but it does arise after we settle on a width choice and the users confront personal costs which are not identical to society's costs. We can also consider the suply-demand effect on q_{ij} now.

With some effort the method can also be adapted to augmenting existing networks [27], though in the presence of Steiner points this is no longer a case of regular network design. The problem is somewhat complicated since connections can be made along highways, and not just to nodes of N. In the honeycomb network representation distances along existing highways should now reflect the costs of changes rather than of new highways, and for some flow values it may mean no intercept value. Also, these existing highways will usually not be deleteable (though due to the Braess' paradox this may be indicated sometimes).

Appendix A

Designing a Single Highway Between Two Points for Given Traffic Flow

In [32], it has been shown that the vertical alignment problem (also refered to as the "red line" problem), including considerations such as earth moving cost and projected capitalized and users' costs (mainly fuel and time value, but also accidents and mechanical wear and tear) is to minimize a convex functional. Constraints on the curvature can be accommodated, and do not change the picture as far as convexity is concerned. Constraints on the slopes are not required any more, since the users' costs are a function of the slope (and of the traffic flow), and the choice of the best slopes is best left to the optimizing procedure itself. (Note that really excessive slopes, which some vehicles cannot negotiate, imply infinite users' costs, and are thus automatically avoided. Another interesting feature here is that for similar traffic flow and terrain the design may be different for different countries due to different time values, rates of return, and so on. The need for such differentiation and of taking the traffic flow into account in a direct manner was independently noted recently in [2], where the issues addressed are curvature and slow lanes.) In [31] a method is shown how to solve this convex functional by natural cubic spline functions and a quasi-Newtonian search method with derivatives. A similar method, but without derivativess (due to the nature of the problem), is extended there to the horizontal alignment problem, where the input is a piecewise linear trajectory, and the output is a smooth, curvature-constraint-abiding curve in the neighborhood, which is locally optimal. Other constraints on the

trajectory, to the amount they are not taken care of by the picewise linear trajectory, can be accommodated with ease. (Sometimes such constraints are a blessing in the manual design process; in the automated process, with computing costs being what they are today--or even less in the future, most probably--we can take it or leave it.)

It remains to discuss how to obtain the piecewise linear trajectory required for the process described. For short stretches, it may very well be appropriate to start with a straight line. This is certainly true if the distance is much less than twice the allowed radius of curvature (see the discussion of a bound on the search area in [32]). However, in our case here, where we are ultimately concerned with the design of very long stretches and a complete network of these, we need a search grid of some kind. E.g., Turner [35] suggested a square grid with eight directions from each node (the nodes are placed in the squares' centers). All kinds of costs are assigned to the squares (he does not dwell much on how this is done), and a shortest path is sought. In [32] several other methods are also discussed, in addition to the square grid method mentioned, specifically and a honeycomb grid, with twelve directions associated with the hours of a clock is introduced there (see Figure 1).

Now, such a grid (square or honeycomb) can take care effectively of important costs such as right of way (including such constraints which actually mean prohibitive right of way costs), ecological and social penalty costs, earth quality, etc. Some other important costs depend on the length of the highway and possibly its general location but are not very sensitive to the exact alignment, e.g., pavement costs and capitalized maintenance costs; these costs can also be accounted for satisfactorily by a grid. The problem arises when we wish to approximate the exact-alignment-sensitive costs, and

these include earth moving costs and capitalized users costs as affected by the highway slope. It is in order to approximate these costs satisfactorily that the honeycomb grid is indeed superior to the simpler square one, as elaborated in [32]. A possible way to take these costs into account is by solving the exact alignment problem for each arc of the grid. To ensure compatibility of the endpoints elevations we can choose several ways: (i) extend the arcs to both sides before solving for the alignment, and then chop off the extra margins--this will tend to make the conditions in the neighboring hexagons affect the design properly; (ii) solve for a set of possible elevations (that is, obtain the arcs' costs as functions of the end elevations); or finally (iii) impose the endpoints elevations from the beginning by a proper heuristic. Be that as it may, the problem is tractable (when we consider its importance).

Needless to say, without some good DTM system, such methods are useless. It does seem that DTMs are alive and well these days, though, so this should not be a hindrance.

Apopendix B

A Lower Bound for f(q) in Flat Terrain Conditions

Assume that the construction costs CC are

(B1) $$CC = b + k \cdot c;\ b,c, > 0,$$

where k is the number of lanes in each direction, b is a fixed cost (right of way, design, etc.), and c is the marginal cost here. (Except for the case of slow truck lanes in upgrades, we may nowadays assume that there is an equal number of lanes in each direction; and for the flat case we are discussing there are no such lanes.) Further, assume low lane changing activity, but enough to obtain equal load on the lanes. (This assumption is more and more valid when the traffic increases, and that is when we really care anyway.)

Now clearly for any positive q we have to invest b + c at least, for a single lane (in each direction). As for the users' costs, the cost for each user is a positive nondecreasing convex function of the traffic flow in the lane. The function is fairly constant for low traffic, and then starts climbing faster. (See [33] for a detailed description. Note that we are not talking about time alone here, where these claims would be very basic indeed. A similar result holds for these costs as a function of the slope, but not necessarily for a function of these two variables together.) Denote that function as TC(q), and since have q such users, the users' cost associated with it, given that we just have one lane per direction, QC(q), is as follows

(B2) $$QC(q) = C \cdot q \cdot TC(q)$$

where C is a capitalization factor. Now, q is convex and monotone, as is TC(q). It follows that QC(q) is likewise (to verify, take the second derivative). Let $TC_k(q)$ denote the users' cost for k lanes carrying a total traffic of q, and similarly for $QC_k(q)$. Then, under our second assumption, for k lanes we get

(B3) $$QC_k(q) = C \cdot q \cdot TC_k(q) = C \cdot q \cdot TC_1(q/k).$$

Clearly (B2) is a special case and should be written with indices of 1. Adding our fixed costs we obtain a function $f_k(q)$ as follows

(B4) $$f_k(q) = b + k \cdot (c + C \cdot (q/k) \cdot TC_1(q/k)).$$

Now, the average cost per user, except for her or his share of b (which can only decrease with q) is

(B5) $$(f_k(q) - b)/q,$$

and we try to minimize it by taking the derivative by q. For k = 1, we get

(B6) $$\frac{d}{dq}[(f_1(q) - b)/q] = \frac{d}{dq}[c/q + C \cdot TC_1(q)] = C \cdot TC_1'(q) - c/q^2.$$

And this derivative is zero (or changes sign if $TC_1(q)$ is not continuously differentiable, a possibility we neglect henceforth) for q_0 such that

(B7) $$q_0 = \arg\{C \cdot TC'(q_0) \cdot q^2 = c\}.$$

This value exists and is unique, due to the convexity of TC. The geometrical interpretation is that the tangent to $f_1(q_0)$ passes through the intercept b. But, extend this tangent further, and it turns out to be the tangent of $f_2(2q_0)$, $f_3(3q_0)$, etc., i.e., it supports the whole set of functions $\{f_k(q)\}_{k=1,2,\ldots}$ and it does that at $k \cdot q_0$. As advertised, this is a lower bound on our total costs function

(B9) $$f(q) = \min_k\{f_k(q)\}_{k=1,2,\ldots}.$$

Figure 2 depicts this result.

Appendix C

Are Two Y's Better than one Regular Cross(roads)?

Yes. Figures C1 and 4 may suffice to make the point.

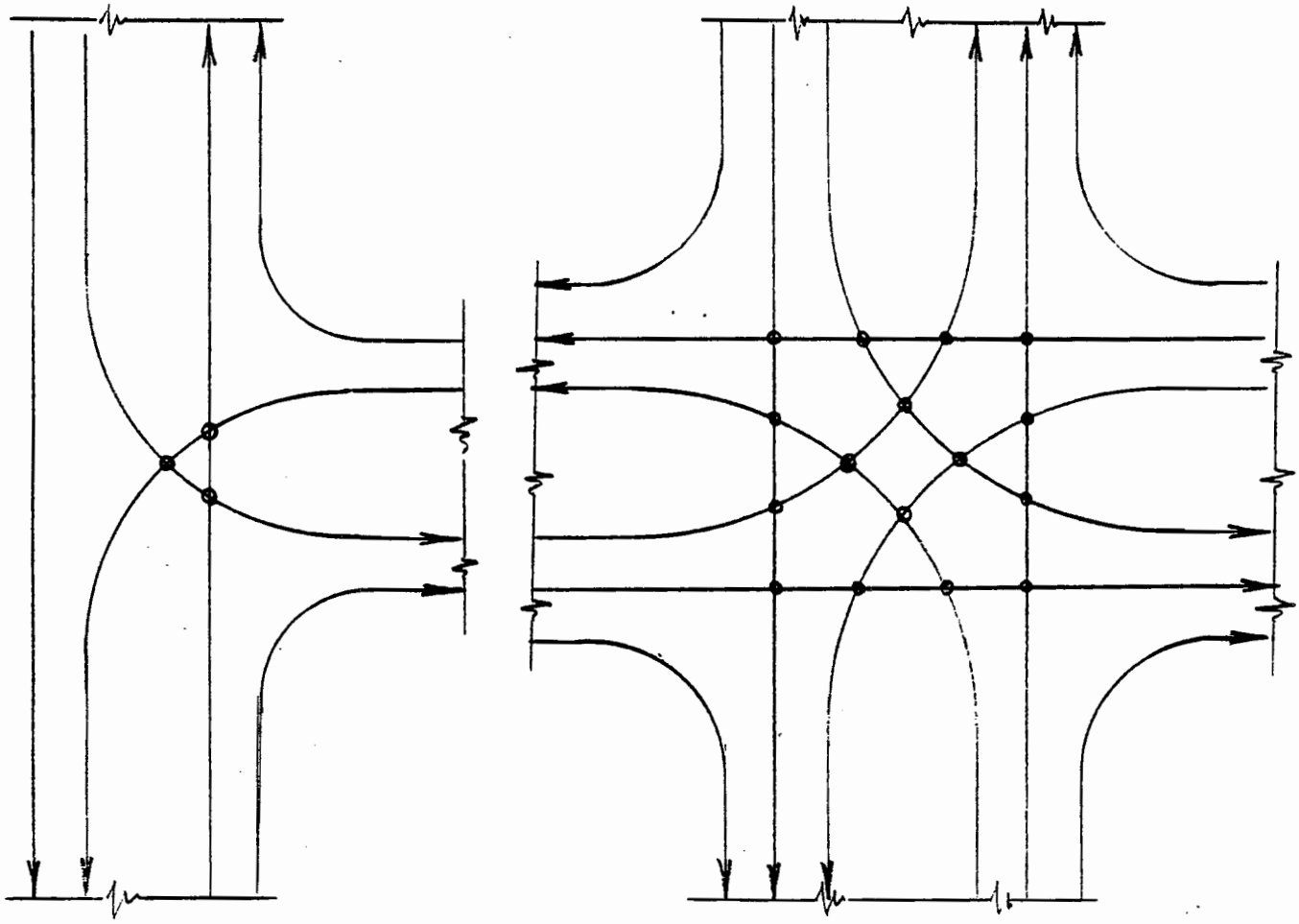

Figure C1

By splitting a regular crossroads into two Y junctions the number of full conflicts is reduced from 16 to a total of 6. Even considering the weaving which results, the picture with regard to half conflicts is still rosy as

well. Furthermore, half conflicts are not half-dangerous when compared to full conflicts, since the relative velocity involved is very low! (A popular stunt in movies is the side-ram, where two high speeding cars collide clashingly, with non-fatal dents the only damage usually. What the nonengineering minded audience is not aware of is, that although the cars are speeding, their relative velocity is low.)

However, if we do split crossroads to Y-pairs, we must maintain a distance of 300-2000 meters between them depending on the traffic flow, to allow for comfortable weaving [16, Chap. 7]. Another point we should mention is that when we use fully separated intersections the advantage is that no full conflicts exist, but half conflicts persist. For a certain traffic flow and up, these expensive separated intersections are justified, and then the advantages of splitting disappear, but it may still yield a better and shorter network.

References

[1] M. Barbier, Le Future Reseau de Transports en Region de Paris, Cahiers de l'Institute d'Amenagement et d'Urbanisme de la Region Parisienne, 4-5, No. 4, 1966.

[2] M. Ben-Akiva, M. Hirsh and J. Prashker, Probabilistic and Economic Factors in Highway Geometric Design, to appear in Trans. Sci., March 1985.

[3] D. E. Boyce (ed.), Transportation Network Design, Trans. Res. 13B, 1, 1979 (a special issue).

[4] S. K. Chang, The Generation of Minimal Trees with A Steiner Topology, J. ACM 19, 1972, pp. 699-711.

[5] A. Claus and D. J. Kleitman, Heuristic Methods for Solving Large Scale Network Routing Problems: The Telpaking Problem, Studies in Applied Mathematics, Vol. LIV, No. 1, 1975, pp. 17-29.

[6] E. J. Cockayne, On the Efficiency of the Algorithm for Steiner Minimal Trees, SIAM J. Applied Math. 18, 1970, pp. 150-159.

[7] G. B. Dantzig, B. P. Harvey, Z. P. Landsowne, D. W. Robinson, and S. F. Maier, Formulting and Solving the Network Design Problem by Decomposition, Trans. Res. 13B, 1, 1979, pp. 5-17.

[8] R. Dial, F. Glover, D. Karney and D. Klingman, A Computational Analysis of Alternative Algorithms and Labeling Techniques for Finding Shortest Path Trees, Networks 9, 1979, pp. 215-248.

[9] R. Dionne and M. Florian, Exact and Approximate Algorithms for Optimal Network Design, Networks 9, 1979, pp. 35-59.

[10] M. R. Garey, R. L. Graham and D. S. Johnson, The Complexity of Computing Steiner Minimal Trees, SIAM J. Appl. Math. 32, 1977, pp. 835-859.

[11] M. R. Garey and D. S. Johnson, The Rectilinear Steiner Tree Problem is NP-Complete, SIAM J. Appl. Math. 32, 1977, pp. 826-834.

[12] E. N. Gilbert, Minimum Cost Communication Networks, Bell System Technical Journal 46, 1967, pp. 2209-2227.

[13] E. H. Gilbert and H. O. Pollak, Steiner Minimal Trees, SIAM J. Appl. Math. 16, 1968, pp. 1-29.

[14] S. L. Hakimi, Steiner's Problem in Graphs and its Implications, Networks, 1, 1971, pp. 195-207.

[15] M. Hanan, On Steiner's Problem with Rectilinear Distance, SIAM J. App. Math. 14, 1966, pp. 255-265.

[16] Highway Research Board, Highway Capacity Manual, Special Report 87, Washington, 1965.

[17] D. S. Johnson, J. K. Lenstra and A. H. G. Rinooy Kan, The Complexity of the Network Design Problem, Networks 8, 1978, pp. 279-285.

[18] R. M. Karp, Reducibility Among Combinatorial Problems, Complexity of Computer Computations, R. E. Miller and J. W. Thatcher (eds.), Plenum Press, New York, 1972, pp. 85-104.

[19] P. Korhonen, An Algorithm for Transforming a Spanning Tree into a Steiner Tree, Computing Center, University of Helsinki, 1976.

[20] H. W. Kuhn, "Steiner's" Problem Revisited, Studies in Optimization, Studies in Mathematics 10, G. B. Dantzig and B. C. Eaves (eds.), The Mathematical Association of America, 1974.

[21] T. L. Magnanti and R. T. Wong, Network Design and Transportation Planning: Models and Algorithms, Trans. Sci. 18, 1984, pp. 1-55.

[22] Z. A. Melzak, On the Problem of Steiner, Canadian Math. Bull. 4, 1961, pp. 143-148.

[23] N. A. Parker, Rural Highway Route Corridor Selection, Transportation Planning and Technology, 3, 1977, pp. 247-256.

[24] A. D. Pearman, The Structure of the Solution Set to Network Optimization Problems, Trans. Res. 13B, 1979, pp. 81-90.

[25] R. C. Prim, Shortest Connection Networks and Some Generalizations, Bell System Technical Journal 36, 1957, pp. 1389-1401.

[26] R. A. Steenbrink, Optimization of Transport Networks, John Wiley and Sons, 1974.

[27] D. Trietsch, On the Transportation Network Design Problem, Ph.D. Dissertation, 1983 (in Hebrew).

[28] D. Trietsch, Augmenting Euclidean Networks--The Steiner Case, to appear in SIAM J. Appl. Math.

[29] D. Trietsch, An Improved Algorithm for Steiner Trees, Discussion Paper

No. 635, The Center for Mathematical Studies in Economics and Management Science, Northwestern University, 1984.

[30] D. Trietsch, Connecting Euclidean Networks--The Steiner Case, Discussion Paper No. 640, The Center for Mathematical Studies in Economics and Management Science, Northwestern University, 1985.

[31] D. Trietsch, Some Notes on Natural Cubic Spline Functions, Nonlinear Regression, and the Elastica, Discussion Paper No. 639, The Center for Mathematical Studies in Economics and Management Science, Northwestern University, 1985.

[32] D. Trietsch, A Family of Methods for Preliminary Highway Design, Discussion Paper No. 645, The Center for Mathematical Studies in Economics and Management Science, Northwestern University, 1985.

[33] D. Trietsch and G. Y. Handler, On Highway Fuel and Time Expenditures, to appear in Trans. Sci. August 1985.

[34] D. Trietsch and G. Y. Handler, The Gilbert and Pollak Conjecture--A Generalization, to appear in Networks.

[35] A. K. Turner, A Decade of Experience in Computer Aided Route Selection, Photogrammetric Engineering and Remote Sensing 44, 12, 1978, pp. 1561-1576.