DELTA-WYE TRANSFORMATIONS AND THE EFFICIENT REDUCTION OF TWO-TERMINAL PLANAR GRAPHS

THOMAS A. FEO
The University of Texas at Austin, Austin, Texas

J. SCOTT PROVAN
University of North Carolina, Chapel Hill, North Carolina

(Received January 1989; revisions received June 1990, November 1991; accepted December 1991)

A simple, $O(|V|^2)$ time algorithm is presented that reduces a connected two-terminal, undirected, planar graph to a single edge, by way of series and parallel reductions and delta-wye transformations. The method is applied to a class of optimization/equilibrium problems which includes max flow, shortest path, and electrical resistance problems.

In this paper, let $G = (V, E)$ be a connected planar undirected graph, in which we allow loops (edges of the form (v, v)) and parallel edges (edges having the same two endpoints), and let s and t be denoted as *terminals* of G. We allow the graph to be transformed by one of the following six rules.

T1: loop reduction: any loop can be removed from G;

T2: pendant edge reduction: if edge $e = (v, u)$ has u a nonterminal vertex of degree 1, then e and u can be removed from G;

T3: series reduction: if $e = (v, u)$ and $f = (u, w)$ with u a nonterminal vertex of degree 2, then e and f and v be replaced by the single edge (v, w);

T4: parallel reduction: if e and f are parallel edges, then they can be replaced by a single edge having the same endpoints;

T5: delta-wye transformation: if $e = (v, w)$, $f = (w, z)$, and $g = (z, v)$ are edges, then e, f, and g can be replaced by the three edges $e' = (u, v)$, $f' = (u, w)$, and $g' = (u, z)$, where u is a nonterminal vertex of degree 3;

T6: wye-delta transformation: the inverse of T6.

These reductions and transformations are illustrated in Figure 1. (Note that it is not necessary to require the vertices of these reductions to be distinct, because loops are allowed. This generalization actually simplifies some of the discussion, although degenerate transformations can be spurned without affecting the results of the paper.) The goal is to use the transformations **T1–T6** to efficiently reduce G to a single edge (s, t). A graph that can be reduced this way is called a *$\Delta Y\Delta$-reducible graph.*

For over a century, series and parallel reductions and delta-wye/wye-delta transformations have been used to simplify the analysis of electrical networks; for a survey of this, see Seshu and Reed (1961) or Brylawski (1977). Akers (1960) uses these operations to help solve the shortest path and maximum flow problems on undirected, two-terminal graphs. He proves that the application of these transformations preserves the optimal length or flow value. Lehman (1963) shows that the series and parallel reductions preserve reliability in the two-terminal, undirected, network reliability problem. He also presents approximations for the delta-wye transformations. These transformations have also been used to solve a variety of problems in statistical physics involving the evaluation of crystal lattice energy (Baxter 1982). They appear as well in the "Reidermeister moves" of Knot theory (Reidermeister 1948) and provide a method for solving other kinds of problems in combinatorial enumeration (Colbourn, Provan and Vertigan 1992).

Both Akers and Lehman conjectured that any connected, two-terminal, undirected, planar graph can be reduced to a single edge between its terminals by employing the above topological operations. This conjecture was proven by Epifanov (1966), using a proof which is ingenious but fairly obscure. Independently, Grunbaum (1967) provides a proof to a simplified version of the Akers-Lehman conjecture, namely, when the graph possesses no terminals. The result is

Subject classifications: Analysis of algorithms, data structures: efficient structure for solving many types of network problems. Networks/graphs, flow algorithms: solves a class of flow equilibrium problems. Networks/graphs, theory: reduction method for 2-terminal planar graphs.
Area of review: OPTIMIZATION.

0030-364X/93/4103-0572 $01.25

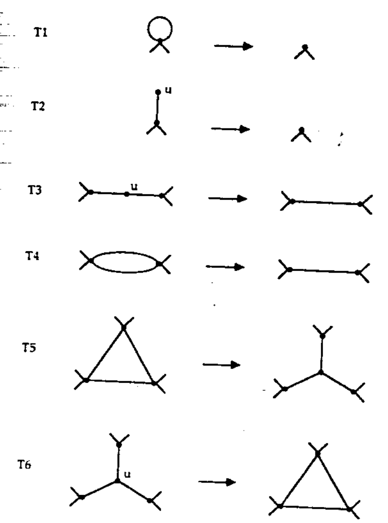


Figure 1. Topological reductions and transformations, where vertex u is not a terminal of the graph.

used by Grunbaum to shorten the proofs of several important theorems, including Steinitz's theorem characterizing the edge graphs of convex 3-polytopes. Recently, both Grunbaum's and Epifanov's proofs have been simplified considerably by Truemper (1989), making use of the fact that G can be embedded in a grid graph.

The computational complexity of reducing a connected graph using these operations has also been studied. When only transformations **T1–T4** are allowed then *series-parallel* graphs (graphs which do not contain a homeomorphic subgraph K_4) are precisely the class of graphs which can be reduced to a single edge, and in fact Valdes, Tarjan and Lawler (1982) give an $O(|V|)$ algorithm for recognizing and reducing series-parallel graphs. Several papers have considered the two classes of graphs reducible by the addition of either transformation **T5** or **T6** (called Δ to Y and Y to Δ *reducible graphs*, respectively), giving forbidden minor characterizations, linear time reduction algorithms, and applications to the computation of all-terminal reliability and other hard problems (Politof 1989, Politof and Satyarayana 1990, Arnborg, Proskurowski and Corneil 1990, El-Mallah and Colbourn 1990). The characterization c
reducible graphs remains an open question. E
a planar graph G there remains the question
many applications of the operations T1–
required to reduce G to a single edge. Epifanc
no indication as to how many reductions are re
The technique given by Truemper implic
$O(|V|^2)$ transformations are required. Actually
menting Truemper's technique in $O(|V|^2)$
problematic, for the size of the smallest grid g
which G can be embedded can grow as the squ
that of G itself, so that a naive implementatio
result in an $O(|V|^3)$ algorithm. In any case, Feo
provided the first $O(|V|^2)$ time algorithm for the
reduction of a two-terminal graph; his method—
on Epifanov's work—is complex and the pr
lengthy.

The purpose of this paper is two-fold. First,
$O(|V|^2)$ algorithm is given for reducing a planar
to a single edge using the reductions **T1–T6**.
algorithm is strikingly simple, relatively easy to p
and can be applied directly to the graph itself,
avoiding the problems of grid embeddings. Sec
an approach is given for looking at optimization
equilibrium problems which unifies the shortest
maximum flow, and electrical network problems
tioned above, as well as indicating how more com
related problems can be solved in this context.

1. THE DELTA-WYE REDUCTION (DWR) ALGORITHM

We assume that the reader is familiar with stand
graph terminology, especially with regard to pla
graphs (see, for example, Bollobás 1979, Section
The **DWR** algorithm transforms a two-terminal *pl*
graph G (planar graph with a fixed planar embeddi
to a single edge using the transformations **T1–T6**
is given in Figure 2 and consists of two parts. T
labeling procedure gives each edge and vertex a la
which will denote the *level* of that edge or vertex w
respect to the terminal s. A sample graph and
labeling is given in Figure 3. The *reduction* procedu
performs a modified set of transformations, call
positive transformations, which depend upon t
levels of the edges in the transformations. The positi
transformations are given in Figure 4. The transfo
mation **P2** is technically not allowed as a pendan
edge reduction when u is a terminal; it is, howeve
convenient to perform this reduction symbolically i
order to simplify the algorithm and the accompanyin
discussion. We note that all **P2** reductions of this typ
except the first are simply series reductions, and tha

Input: Connected plane graph G=(V,E), along with specified terminals *s* and *t*.

Output: Sequence of transformations T1-T6, which reduces G to the single edge (*s*,*t*).

Step 1: {Label}

Assign the label 0 to *s*, and declare all other vertices and edges unlabelled.

Set *l* := 1 {*l* = current level}.

do while (there are unlabelled vertices or edges),

To each unlabelled edge incident to a labelled vertex, assign the label *l*.

To each unlabelled edge sharing a face with a labelled vertex, assign the label *l*+1.

To each unlabelled vertex adjacent to a newly labelled edge, assign the label *l*+1.

Set *l* := *l*+2.

end while.

Step 2: {Reduce}

do while (G is not a single vertex),

Find and perform a positive transformation.

end while.

Figure 2. The **DWR** algorithm.

this transformation will have no affect on the allowable transformations available to the **DWR** algorithm (*modulo* possibly having to re-embed a pendant edge out of a loop, parallel, or delta region). The sequence of transformations now ends with the single vertex $t = s$; reversing the first **P2** transformation on t thus yields the edge (s, t) as required.

To prove the validity of the **DWR** algorithm, we first give some notation. A *contour* of level l is any of the isolated vertices, bridges, or 2-connected components in the subgraph of G consisting of all edges and vertices with the same level l, where l is even. A contour C may consist of a single vertex, a single edge and its incident vertices, or a loop or simple polygon (called a *simple* contour). Figure 3 gives examples of these, marked by C_1, C_2 and C_3, respectively. Thus, a contour defines a partition of $\mathbb{R}^2 \setminus C$ into two open regions (one being empty if C is not simple), exactly one of which contains only edges or vertices of level $l + 1$ or higher. This region will be called the *uphill region* of C, and the other region is the *downhill region*. Note that the contour at level 0 consists of the single vertex $\{s\}$, which is the only contour whose downhill region is empty. An *adjacent uphill contour* (AUC) of C is any contour of level $l + 2$ lying in the uphill region of C.

Intuitively, it is easiest to visualize the labelings and contours by reorienting the embedding of G so that s is on the exterior face and given level 0 (this will not change any vertex labels). Now for each even level l, we can recursively obtain the level $l + 2$ vertices and edges as the boundary of the graph obtained after removing the level l vertices and their adjacent edges. The contours will proceed to appear as a series of "layers" of even labeled vertices and edges, which are connected to adjacent layers by the appropriately odd labeled edges. (Odd labeled edges can also connect vertices of the same simple contour from the inside.) Each contour $C \neq \{s\}$ now has its uphill region the interior of C, which is empty if C is not simple, and its downhill region to be its exterior. Thus, the simple contours at level l produce a further partition of the contours of level $l + 2$, with each contour C of level $l + 2$ lying inside a unique simple contour of level l for which C is an AUC.

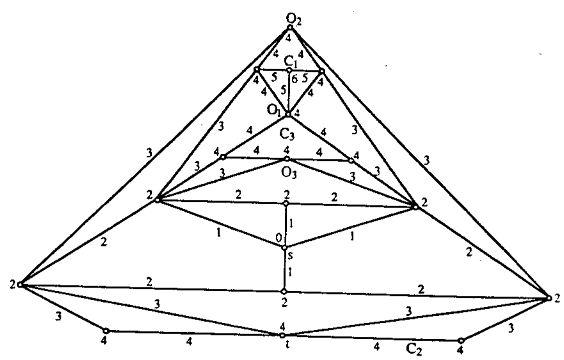


Figure 3. Example of a graph labeling.

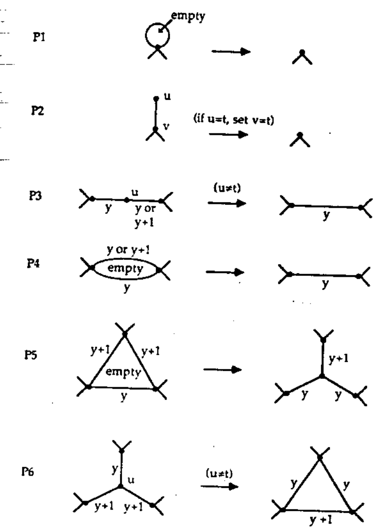


Figure 4. Positive transformations, where y and $y + 1$ denote level of the edge.

That the positive transformations will have labelings as given in Figure 4 can easily be seen in the context of the structure given above. We will consider transformations **P5** and **P6**; the transformations **P3** and **P4** are left to the reader. The only way a **P5** transformation can appear is when the edge with label y is on a simple contour (so that y is even) with the level $y + 1$ edges uphill of this contour. After the transformation, the two edges that connect the endpoints of the level y edges will be on the same contour as this edge, with the third edge again lying uphill. The only way a **P6** transformation can appear is when the two edges with level $y + 1$ are on the same simple contour or comprise two single-edge contours, with the level y edge connecting them to the adjacent downhill contour (so that y is odd). In this case, after the transformation the edge that connects the endpoints of the level $y + 1$ edges will remain at the same level, with the level y edge downhill. Thus, if the edges are given the labelings indicated by Figure 4, then the graph will continue to have labels consistent with that produced by the labeling procedure.

The inability of the **DWR** algorithm to perform the appropriate positive transformations at a given level turns out to be caused by *exposed vertices* on a contour and their associated *obstructions*. An *exposed vertex* of contour C is a vertex of C having no inciden
edges (with respect to C); C is called a *k-conto*
has k or fewer exposed vertices. It follows t
contour $\{s\}$ has no exposed vertices unless G is
(a single vertex with no edges), and that each
in any other nonsimple contour is exposed. Su
then, that $C \neq \{s\}$, and let D be the (unique) c
for which C is an AUC. Associated with each e
vertex v are three types of *obstructions* of v *with*
to C, which are illustrated in Figure 5. The ve
itself is a *type*-1 obstruction if it is contained in a
one other AUC of D. Type 2 and 3 obstructions
when v is not a type-1 obstruction. They are d
by a region P lying in the downhill region of
enclosed by two clockwise consecutive
of level $l - 1$ adjacent to v together with that p
Γ of D lying between them. P is a *type*-2 obstru
if it contains another AUC of D in its interior
type-3 obstruction if it contains no such AUC a
contains an exposed vertex. In Figure 3 obstruc
of types 1, 2, and 3 occur, respectively, at poin
O_2, and O_3. Note that the (single vertex) obstru
at O_1 is an obstruction *only* with respect to co
C_3, since it is not exposed in the AUC above it
obstruction at O_2 is comprised of the region ex
to the 4-edge polygon of level 3 edges at O_2 tog
with the two adjacent level 2 edges connecting
endpoints, and contains, for example, the AUC
The first result establishes that, barring a pos
transformation, each exposed vertex must be as
ated with at least one obstruction.

Lemma 1. *Let C be a contour of level l in a lab*
nontrivial graph that admits no positive transfor
tions. Then each exposed vertex of C must either
terminal or be incident to an obstruction of type
or 3.

Proof. Let v be an exposed vertex of C which is nei
a terminal nor an obstruction of type 1. Then v
have at most two adjacent level l edges, and
must have at least two adjacent level $l - 1$ ed
because otherwise v would immediately admit a p
itive transformation. Choose two of these edges e_1 a
e_2 that are consecutive in a clockwise sweep of
downhill edges incident to v. Let D be the contour
which C is an AUC, and consider the region P s
rounded by e_1, e_2, and the portion Γ of D betw
them. If there are any vertices inside P, then there
at least one AUC of D inside P, and hence P is a ty
2 obstruction. If there are only edges inside P, th
these must connect vertices of Γ. The fact that th
are no **P1** or **P4** transformations on D implies th

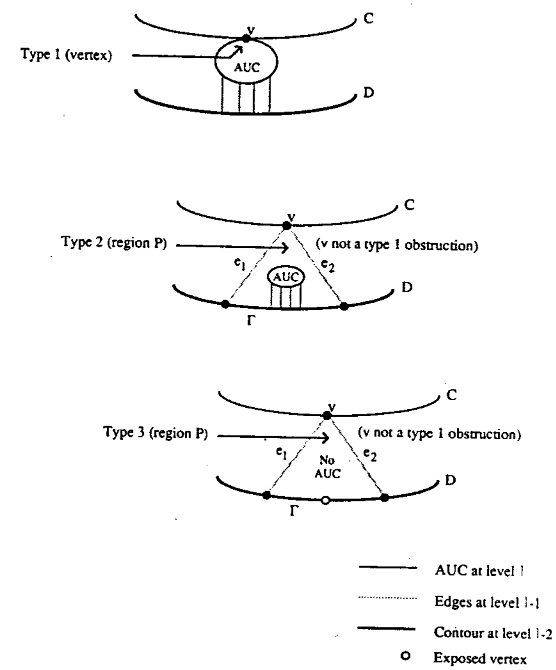


Figure 5. Types of obstructions to an exposed vertex.

the two nearest points of Γ adjacent through one of these edges cannot be the same or adjacent in Γ, and thus any vertex between them must be exposed. Finally, if P has empty interior, then the fact that e_1 and e_2 are not part of a **P4** or **P5** transformation implies that there is at least one vertex of Γ between them, which must again be exposed. In either of these last two cases P is a type-3 obstruction and the lemma follows.

Lemma 2. *Let C be a k-contour whose uphill region is vertex nonempty. Then at least one of the following two conditions holds:*

i. *There exists an AUC of C having no type-1 or 2 obstructions and at most k type-3 obstructions.*

ii. *There exist two AUCs of C each having exactly one type-1 or 2 obstruction and together having at most k type-3 obstructions.*

Proof. Let $D_1, \ldots, D_r$ be the set of AUCs of C, partitioned into connected components $B_1, \ldots, B_l$. We first prove that there exists a component B_{i_*} with no type-2 obstruction, or two components B_{i_*} and B_{j_*} with one type-2 obstruction each. Start with any component B_{i_0}. If B_{i_0} has no type-2 obstruction, we are done. Otherwise, let P_1 be any type-2 obstruction on B_{i_0}, and let B_{i_1} be any component contained in P_1. If B_{i_1} has two or more type-2 obstructions, then at least one of the obstruction regions does *not* contain B_{i_0} inside it, because the type-2 obstructions adjacent to B_{i_1} have disjoint interiors. Set P_2 to be this obstruction. Proceeding in the same manner, we produce a sequence $B_{i_0}, B_{i_1}, \ldots,$ of components, along with incident type-2 obstructions $P_1, P_2, \ldots,$ such that P_j contains no B_{i_q}, $q < j$. This sequence must, therefore, terminate in component $B_{i_*} \neq B_{i_0}$ having at most one type-2 obstruction. Now repeat the process, starting at $B_{j_0} = B_{i_*}$ and creating the sequence $B_{j_0}, B_{j_1}, \ldots,$ of components and associated type-2 obstructions. This sequence must likewise terminate in component $B_{j_*} \neq B_{i_*}$ having exactly one type-2 obstruction. Thus, we have produced the component or components, as required.

We next find the desired contour or contours required by the lemma among those in B_{i_*} or B_{j_*}. If B_{i_*} (B_{j_*}) is a single contour, then it can have no type-1 obstruction. Otherwise, the contours in B_{i_*} (B_{j_*}) consist of the single-edge and simple contours of B_{i_*} (B_{j_*}) joined by *cutvertices* that are exactly the type-1 obstructions on B_{i_*}(B_{j_*}). Thus, B_{i_*} (B_{j_*}) forms a tree structure—called the *block-cutvertex* graph of B_{i_*} (B_{j_*})—with contours and cutvertices corresponding to the vertices of the tree and contour-cutvertex incidences defining the edges of the tree (see Bollobás, p. 51). It follows that there must be at least two degree 1 vertices of this tree (called *endblocks*), which must correspond to two contours D_{i_1} and D_{i_2}(D_{j_1} and D_{j_2}) each having exactly one type-1 obstruction.

From the above discussion, we obtain *one* of the following: a) one contour with no type-1 or 2 obstruction, b) two contours with one type-1 or 2 obstruction each, c) three contours with a total of four or fewer type-1 or 2 obstructions, or d) four contours with a total of six or fewer type-1 or 2 obstructions. If case a does not occur, then cases b–d imply that there must be at least two contours with one type-1 or 2 obstruction each. Finally, by the definition of type-3 obstruction (particularly that the associated vertex v is not a type-1 obstruction) it follows that each exposed vertex contributes to at most one type-3 obstruction on the AUCs of C. Since there are at most k exposed vertices, there can be a total of at most k type-3 obstructions among the contours chosen, and hence the contour or contours will satisfy either part i or ii, as required by the lemma.

We next give two lemmas which take care of special cases.

Lemma 3. *Let C be a simple 1-contour with vertex empty uphill region. Then C admits a positive transformation.*

Proof. We have that C contains at most one exposed vertex, and that all of its uphill edges connect points of C. If C has no uphill edges, then C is a loop and must admit a **P1** transformation. Otherwise let e be an uphill edge, cutting C into parts C_1 and C_2, and let C_1 be that part not containing an exposed vertex. Then the nearest pair of vertices on C_1 adjacent by an uphill edge must also be equal or adjacent on C_1 itself, and so a **P1** or **P4** transformation exists.

Lemma 4. *Let C be a contour satisfying o[…] following conditions:*

i. *$C = \{v\}$, $v = t$, and C has no obstructions[…]*
ii. *$C = \{v\}$, $v \neq t$, and C has at most one obs[…]*
iii. *$C = \{(u, v)\}$ $v \neq t$, and v is neither […] obstruction nor adjacent to a type-[…] obstruction.*

Then G admits a positive transformation.

Proof. Part iii follows from Lemma 1 (sin[…] exposed) and parts i and ii follow from essenti[…] same argument.

Theorem 1. *Let G be a two-terminal plane[…] which has been labeled according to the* **DW**[…] *rithm. Then G is either a single point or ad[…] least one positive transformation.*

Proof. Let C be a simple k-contour of highes[…] $k \leq 1$ having at most $1 - k$ terminals in its […] region. (There must exist at least one such […] because $\{s\}$ is such a contour.) First suppose t[…] has a vertex-empty uphill region. Then Lem[…] applies, and C admits a positive transform[…] Otherwise Lemma 2 applies, and we have two c[…]

Case 1. There exists an AUC D having no type[…] 2 obstructions and at most k type-3 obstructions[…] is a single vertex then Lemma 4i or ii applies, g[…] a positive transformation on D. If D is a single […] then one of the ends of this edge must be a no[…] minal vertex incident to no type-3 obstruction. […] Lemma 4iii applies, again giving a positive tran[…] mation. Finally, if D is simple and G admits […] positive transformation, then Lemma 1 implies […] D can have at most $k + r$ exposed vertices, where […] the number of terminals on C (and, hence, not i[…] uphill region). Thus, D is a contour of higher l[…] that satisfies the same properties as C, contradic[…] the choice of C.

Case 2. There exist two AUCs D_1 and D_2, each hav[…] exactly one type-1 or 2 obstruction and together h[…] ing at most k type-3 obstructions. Now D_1 and […] have at most one vertex in common, which then m[…] be a type-1 obstruction to both. Furthermore, […] definition each type-3 obstruction of $D_1 \cup D_2$ is as[…] ciated with exactly one vertex of either D_1 or D_2, b[…] not both. What this means is that D_1 and D_2 togeth[…] have at most 3 obstructions, and so one of D_1 or D[…] say D_1, has at most 1 obstruction. If t is downhill fro[…] D_1, then similar to case 1, Lemma 1 or 4 can […] applied to identify a positive transformation or contradiction. If t is downhill from D_2 but not downhill from D_1, then $k = 0$, so that D_2 has at most 1 obstruction and again Lemma 1 or 4 applies. Finally, if t is downhill from *neither* D_1 nor D_2, then t must be the type-1 obstruction to both. Thus, D_1 has at most 1 obstruction, t is not in the uphill region of D_1, and if D_1 is a single edge, then the nonterminal end of D_1 cannot be a type-1 obstruction. Again Lemma 1 or 4 applies, and this completes the proof of the theorem.

Corollary 1. *The* **DWR** *algorithm correctly reduces a 2-terminal plane graph to a single edge in time* $O(|V|^2)$.

Proof. The validity of the **DWR** algorithm follows from Theorem 1. To analyze the complexity we first note that the labeling procedure can be performed in linear time. Furthermore, since at least one new vertex is labeled at each even numbered level, the highest label any edge or vertex can take is $2|V|$. Define the *potential* of the graph at each stage of the algorithm to be the sum of the levels of the edges at that stage. Then the original potential is at most $2|V||E| = O(|V|^2)$, and each transformation reduces the potential by at least one. It follows that after $O(|V|^2)$ transformations the resulting graph has potential 0, implying that it is a single point. Each transformation or reduction requires constant time to perform. By keeping track of the degree of all vertices and faces, locating the positive transformations as they occur also requires constant time per transformation. Thus, the total running time of the **DWR** algorithm is $O(|V|^2)$.

2. AN APPLICATION OF THE DWR ALGORITHM TO NETWORK OPTIMIZATION AND EQUILIBRIUM PROBLEMS

The kinds of reductions considered in this paper have an interesting application to problems of network optimization and electrical equilibrium. The general setup for these problems is as follows. We are given a graph $G = (V, E)$ with terminals s and t. Associated with each edge (i, j) is a real-valued *flow* f_{ij} and *potential* γ_{ij}. These two values are connected by a *potential-flow equilibrium relation* $R_{ij} \leqslant \mathbb{R}^2$ that describes the allowable pairs of values which can be taken on by f_{ij} and γ_{ij} when the network is in equilibrium. The R_{ij} can have a fairly general form, depending on the particular application considered. Figure 6 gives three such forms. The relations given in the

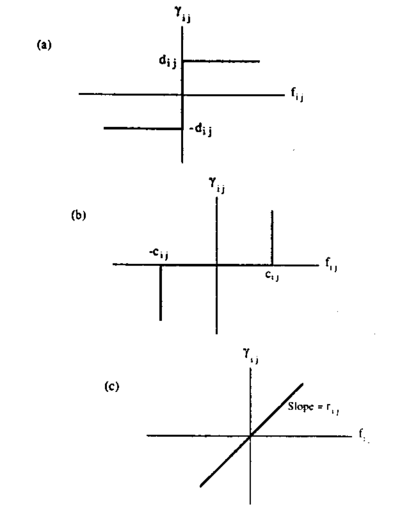


Figure 6. Three potential flow equilibrium relations.

figure happen to be *symmetric*, that is, negating both f_{ij} and γ_{ij} will result in the same relation. This, in essence, allows bidirectional flow without having to specify which direction corresponds to positive flow. In general, R_{ij} need not be symmetric and in this case a specific *orientation* must be given to the associated edge to distinguish the direction of positive flow. Asymmetric relations thus correspond to *directed* graph models.

The general equilibrium problem has as input G (along with edge orientations if the problem is asymmetric), relations R_{ij} for each edge (i, j), and either an *(s-t)-flow value* F, or an *(s-t)-potential value* Γ. The output is a collection of pairs (f_{ij}, γ_{ij}) for each edge (i, j) satisfying the following conditions.

1. $f_{ij} = -f_{ji}$, $\gamma_{ij} = -\gamma_{ji}$ for each edge (i, j).
2. The vector f constitutes a *valid flow* (of value F if F is given), that is,

$$\sum_{j:(i,j)\in E} f_{ij} = 0 \quad i \in V\setminus\{s, t\}$$

$$\sum_{j:(s,j)\in E} f_{sj} = \sum_{j:(j,t)\in E} f_{jt} = F \quad \text{if } F \text{ is given.}$$

3. The vector γ constitutes a *valid potential* (of value Γ if Γ is given), that is

$$\sum_{l=1}^{r} \gamma_{i_{l-1}i_l} = 0 \quad \text{for all cycles } i_0, (i_0, i_1), i_1, (i_1, i_2), \ldots,$$

$$i_{r-1}, (i_{r-1}, i_r) i_r = i_0 \text{ in } G;$$

$$\sum_{l=1}^{r} \gamma_{i_{l-1}i_l} = \Gamma \quad \text{for any (and hence every) } (s, t)\text{-path}$$

$$s = i_0, (i_0, i_1), i_1, (i_1, i_2), \ldots,$$

$$i_{r-1}, (i_{r-1}, i_r) i_r = t \text{ in } G \text{ (if } \Gamma \text{ is given).}$$

4. The pair $(f_{ij}, \gamma_{ij}) \in R_{ij}$ for each edge (i, j).

Three examples of equilibrium problems and their associated relations can be illustrated by the relations given in Figure 6.

Shortest Path Problem. Here we are given *distances* d_{ij} on the edges of G, and we wish to find the shortest (s, t)-path in G. To do this, we use the relation given in Figure 6a and set $F = 1$. Then the (s, t)-potential Γ associated with the equilibrium solution will be the minimum cost of moving one unit of flow from s to t, i.e., the length of a minimum cost (s, t)-path.

Maximum Flow Problem. Here each edge (i, j) of G has a *capacity* c_{ij}, and we are interested in finding the maximum flow which can pass from s to t. This uses the relation given in Figure 6b. By setting $\Gamma = 1$, the equilibrium (s, t)-flow F will be the maximum (s, t)-flow in G.

Electrical Resistance Problem. Here G represents an electrical network with each edge (i, j) having resistance r_{ij}. We wish to compute the resistance between two vertices s and t. Here f_{ij} represents the current in (i, j) and γ_{ij} represents the voltage across (i, j), and the relation given in Figure 6c is used. To compute (s, t)-resistance, it is simply a matter of determining the net potential = voltage between s and t necessary to maintain a flow = current of 1 from s to t. Thus, again we set $F = 1$, solve the associated equilibrium problem and determine the resulting Γ. We could also compute *conductance* between s and t, which is the inverse of resistance, by setting Γ to 1 and solving for F.

The relations given in Figures 6a and 6b are simply the "kilter diagrams" associated with the maximum flow and shortest path problems, respectively. That the equilibrium conditions produce the optimal solution can be seen easily from noting that the equilibrium flow values comprise the optimal primal solution

for the associated linear program, and that the eq rium potential value on each edge is equal t difference between the cost and the reduced co that edge at optimum. By using more complex tionships, one can as well model nonlinear elec resistance problems or minimum cost flow prob as well as many other equilibrium situations.

When any of the transformations given in Se 1 are performed (other than loop or pendant reductions) it is necessary to define appropriate tions on the new edges so that the equilibrium v for the remaining edges are identical to those v could occur before the transformation takes plac define these equations we use the following not for relations R and S:

$$-R = \{(x, y): (-x, -y) \in R\}$$

$$R + S = \{(x, y + z): (x, y) \in R, (x, z) \in S\}$$

$$R \vee S = \{(x + y, z)(x, z) \in R, (y, z) \in S\}.$$

One important property of relations involves description relative to the two orientations of an

If e is an edge with endpoints i and j and R relation for the orientation (i, j), then $-$ the relation for the orientation (j, i).

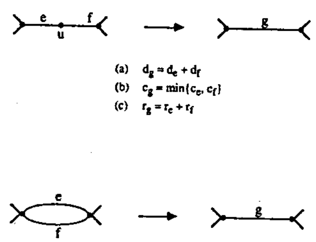
It is immediate that the relations used for an rected network must have the symmetry $R_{ij} =$ and this is true of the relations given in Figure 6 will henceforth give the relation in the orient which is most convenient.

The series and parallel reductions have straightforward relational transformations, name

If (v, u) and (u, w) are replaced by the edge $(v,$ a series reduction, then $R_{vw} = R_{vu} + R_{uw}$;

If e and f are parallel edges replaced by the edg a parallel reduction, then $R_g = R_e \vee R_f$, the relations are all taken with respect to the orientation.

Figure 7 gives the relations of the edges resulting a series or parallel reduction performed on edges relations given by Figure 6, in terms of the rel parameters. Thus, in principle the equilibrium lem for series-parallel graphs can be solved for ar of relations on the edges, and in practice this ca done efficiently for relations which are piecewis ear. It is easy to verify that the amount of work in computing $R + S$ and $R \vee S$ is linear in the number of piecewise-linear parts of R and S. By the linear time series-parallel reduction algorith Valdes, Tarjan and Lawler, we get the following r



(a) $d_g = d_e + d_f$
(b) $c_g = \min\{c_e, c_f\}$
(c) $r_g = r_e + r_f$

(a) $d_g = \min\{d_e, d_f\}$
(b) $c_g = c_e + c_f$
(c) $r_g = r_e r_f / (r_e + r_f)$

Figure 7. Series and parallel transformations of potential equilibrium relations.

Theorem 2. *The general equilibrium problem with continuous, piecewise-linear, nondecreasing, potential-flow equilibrium relations can be solved on a series-parallel graph in time $O(k|V|)$, where k is the total number of piecewise-linear parts in the potential-flow equilibrium relations. In particular, any minimum cost flow problem with convex, differentiable, piecewise-quadratic edge cost functions can be solved on a series-parallel graph in time $O(k|V|)$, where k is the total number of piecewise-quadratic parts in the edge cost functions.*

The delta-wye and wye-delta transformations involve a more complex analysis of the associated relations. Consider a transformation between a delta with edges (v, w), (w, z), and $($ $u)$, and a wye with edges (u, v), (u, w), and (u, z). To see how the relations are interconnected, we equate the composite relation linking, respectively, v and w, w and z, and z and v in the *wye* and the delta. Using the series and parallel transformations given above, this yields the following three relational equations:

$$R_{vu} + R_{uw} = R_{vw} \vee (R_{vz} + R_{zw})$$

$$R_{wu} + R_{uz} = R_{wz} \vee (R_{wv} + R_{vz})$$

$$R_{zu} + R_{uv} = R_{zv} \vee (R_{zw} + R_{wz}).$$

To derive the relations associated with a delta-wye or wye-delta transformation, it is necessary to solve for R_{uv}, R_{uw}, R_{uz} in terms of R_{vw}, R_{wz}, and R_{zv}, or vice versa. For general relations this is difficult, and in most cases impossible, and furthermore it does not guarantee agreement of values when 3-way flow is present. For the relations given in Figure 6, however, there are solutions, and the linearity of potential-flow equilibrium relations between extreme values of f_{ij} or γ_{ij} guarantees that they will jointly satisfy the equilibrium conditions. The equations (in terms of the relevant parameters defining the relations) are

$$d_{uv} + d_{uw} = \min\{d_{vw}, d_{wz} + d_{zv}\}$$

$$d_{uw} + d_{uz} = \min\{d_{wz}, d_{zv} + d_{vw}\}$$

$$d_{uz} + d_{uv} = \min\{d_{zv}, d_{vw} + d_{wz}\}.$$

$$\min\{c_{uv}, c_{uw}\} = c_{vw} + \min\{c_{wz}, c_{zv}\}$$

$$\min\{c_{uw}, c_{uz}\} = c_{wz} + \min\{c_{zv}, c_{vw}\}$$

$$\min\{c_{uz}, c_{uv}\} = c_{zv} + \min\{c_{vw}, c_{wz}\}.$$

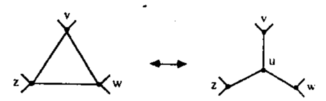
$$r_{uv} + r_{uw} = \left(\frac{1}{r_{vw}} + \frac{1}{r_{wz} + r_{zv}}\right)^{-1}$$

$$r_{uw} + r_{uz} = \left(\frac{1}{r_{wz}} + \frac{1}{r_{zv} + r_{vw}}\right)^{-1}$$

$$r_{uz} + r_{uv} = \left(\frac{1}{r_{zv}} + \frac{1}{r_{vw} + r_{wz}}\right)^{-1}.$$

The solutions are given in Figure 8. Corollary 2 now follows the **DWR** algorithm of Section 2.

Corollary 2. *For any of the relational forms given in Figure 6 the associated equilibrium problem can be solved by the delta-wye reduction method in time $O(|V|^2)$.*



(a) $d_{vw} = d_{uv} + d_{uw}$, $d_{wz} = d_{uz} + d_{uw}$, $d_{zv} = d_{uz} + d_{uv}$

$d_{uz} = (d_v + d_w - d_z)/2$
$d_{uv} = (d_z + d_w - d_v)/2$
$d_{uw} = (d_z + d_v - d_w)/2$, where
$d_v = \min\{d_{wz}, d_{vw} + d_{zv}\}$
$d_w = \min\{d_{zv}, d_{vw} + d_{wz}\}$
$d_z = \min\{d_{vw}, d_{wz} + d_{zv}\}$

(b) $c_{vw} = (c_v + c_w - c_z)/2$
$c_{wz} = (c_z + c_w - c_v)/2$
$c_{zv} = (c_z + c_v - c_w)/2$, where
$c_v = \min\{c_{uv}, c_{uz} + c_{uw}\}$
$c_w = \min\{c_{uw}, c_{uz} + c_{uv}\}$
$c_z = \min\{c_{uz}, c_{uv} + c_{uw}\}$

$c_{uz} = c_{wz} + c_{zv}$
$c_{uv} = c_{vw} + c_{zv}$
$c_{uw} = c_{vw} + c_{wz}$

(c) $r_{vw} = r_Y / r_{uz}$
$r_{wz} = r_Y / r_{uv}$
$r_{zv} = r_Y / r_{uw}$, where
$r_Y = r_{uz}r_{uv} + r_{uz}r_{uw} + r_{uv}r_{uw}$

$r_{uz} = r_{wz} r_{zv} / r_\Delta$
$r_{uv} = r_{zv} r_{vw} / r_\Delta$
$r_{uw} = r_{vw} r_{wz} / r_\Delta$, where
$r_\Delta = r_{vw} + r_{zv} + r_{wz}$

Figure 8. Delta-wye transformations of potential equilibrium relations.

In the context of the min-cost flow problem, moreover, we get the following interesting result.

Theorem 3. *A minimum cost uncapacitated (s, t)-flow of specified value F can be found on a planar undirected network with quadratic edge costs of the form $c_e(x) = r_e x^2$, $r_e > 0$, $e \in E$, in time $O(|V|^2)$ using the delta-wye reduction method.*

3. EXTENSIONS AND CONCLUSIONS

The purpose of this paper was to present the **DWR** algorithm, an $O(|V|^2)$ algorithm for reducing a planar, 2-terminal graph to a single edge using the transformations **T1–T6**, and to show the application of the **DWR** algorithm to two-terminal equilibrium problems relating to shortest path, maximum flow, and electrical resistance. To conclude we will mention briefly two further uses of the **DWR** algorithm for solving related problems. The first of these involves the extension of the various equilibrium problems to situations involving more than two terminals. A good example has been illustrated by Feo who extended the maximum (s, t)-flow application to *multicommodity flows*, where several commodities flow between multiple sources and sinks in a network subject to capacity constraints on total flow through each edge. Feo showed that the same transformations given in Figures 7b and 8b can be used for the multicommodity case, so long as the vertex u is not a terminal. Presumably multiterminal extensions exist for other types of equilibrium problems as well.

The multiterminal extension raises an important question, namely: Does the **DWR** algorithm reduce graphs with more than two terminals to "small" final graphs through positive transformations? The answer is no, even for the case of three terminals, as illustrated in Figure 9. In this graph there exists no positive transformation (with respect to s), and hence the algorithm as it is given would halt on this graph. One may ask further whether there exists *any* series of transformations **T1–T6** which can reduce a given 3-terminal planar graph to a single delta (or wye). This question has recently been answered in the affirmative (Gitler 1991), although an $O|v|^2$ algorithm is not known. When four or more terminals are present there may be *no* way of reducing the graph by **T1–T6** transformations, as illustrated in Figure 10. Here all vertices and nonterminal vertices are of degree four, and thus no transformation of type **T1–T6** can be performed. The **DWR** algorithm, however, can provide

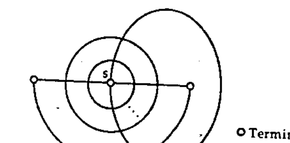


Figure 9. Three-terminal graph admitting no transformation.

a good method for obtaining a substantial re in the size of an arbitrary graph by, for e applying it successively, using each terminal as s, until no further positive transformations found. Preliminary empirical studies by Feo sh a careful application of transformations T1– often produce a substantial reduction in the s multiterminal graph. In solving multicommod problems, for example, this results in a graph ir the commodity flows can be found either dire by solving a reasonably small linear program.

Another useful application of the **DWR** alg is to the computation of source-to-sink connect reliability in a planar graph with randomly and pendently failing edges. Although this prob known to be NP-hard (Provan 1983), a nice ap mation scheme was suggested by Lehman, wh edge failure reliability transformations associate each of the transformations **T1–T6** used in the algorithm. Although the delta-wye and wye transformations do not preserve reliability e: they offer remarkably good approximations t actual (s, t)-connectedness reliability (Chari, Fe Provan 1992). These transformations apply i multiterminal case as well, and thus suggest a approximation heuristic for the problem of comp the probability that a given set K of vertices ca connected pairwise by operating paths in the net (called the *K-terminal reliability* problem).

One final open question concerns a lower bou

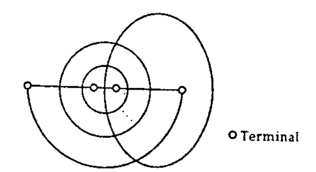


Figure 10. Four-terminal graph admitting no transformation.

the complexity of reducing a 2-terminal planar graph via the transformations **T1–T6**. The **DWR** algorithm requires $O(|V|^2)$ such transformations, but there are compelling reasons to think that $O(|V|^{3/2})$ is the smallest possible order. This problem also appears to be quite difficult, and therefore we leave this problem for future research.

ACKNOWLEDGMENT

This research was partially supported by the Air Force Office of Scientific Research under grant AFOSR-84-0140; reproduction in whole or part is permitted for any purpose of the United States Government. This work was done while the first author was visiting the Department of Operations Research at the University of North Carolina at Chapel Hill in spring semester, 1988.

REFERENCES

AKERS, S. B. 1960. The Use of Wye-Delta Transformations in Network Simplification. *Opns. Res.* **8**, 311–323.

ARNBORG, S., A. PROSKUROWSKI AND D. G. CORNEIL. 1990. Forbidden Minors Characterization of Partial 3-Trees. *Disc. Math.* **80**, 1–19.

BAXTER, R. J. 1982. *Exactly Solved Models in Statistical Mechanics*. Academic Press, New York.

BOLLOBÁS, B. 1979. *Graph Theory: An Introductory Course*. Springer-Verlag, New York.

BRYLAWSKI, T. 1977. A Determinantal Identity for Resistive Networks. *SIAM J. Appl. Math.* **32**, 443–449.

CHARI, M., T. A. FEO AND J. S. PROVAN. 1992. The Delta-Wye Approximation Procedure for (s, t)-Connectedness Reliability. Technical Report No. UNC/OR/TR-92-18, Department of Operations Research, University of North Carolina, Chapel Hill.

COLBOURN, C. J., J. S. PROVAN AND D. VERTIGAN. 1992. A New Approach to Solving Three Combinatorial Enumeration Problems on Planar Graphs. Technical Report No. UNC/OR/TR92-19. Department of Operations Research, University of North Carolina, Chapel Hill.

EL-MALLAH, E. S., AND C. J. COLBOURN. 1990. On Two Dual Classes of Planar Graphs. *Disc. Math.* **80**, 21–40.

EPIFANOV, G. V. 1966. Reduction of a Plane Graph to an Edge by a Star-Triangle Transformation. *Doklady* **166**, 13–17.

FEO, T. A. 1985. Efficient Reduction of Planar Networks for Solving Certain Combinatorial Problems. Ph.D. Dissertation (Part II). Department of Industrial Engineering and Operations Research. University of California, Berkeley.

GITLER, I. 1991. Reduction of a Three-Terminal Graph to K_3 by $\Delta \Leftrightarrow Y$ Transformations. Preprint. Department of Combinatorics and Optimization, University of Waterloo, Ontario.

GRUNBAUM, B. 1967. *Convex Polytopes*. Interscience, London.

LEHMAN, A. 1963. Wye-Delta Transformations in Probabilistic Network. *J. SIAM* **11**, 773–805.

POLITOF, T. 1989. A Characterization and Efficient Reliability Computation of Δ-Y Reducible Networks. Ph.D. Thesis, University of California, Berkeley.

POLITOF, T., AND A. SATYARAYANA. 1990. A Linear Algorithm to Compute the Reliability of Planar Cube-Free Networks. *IEEE Trans. Rel.* **R-39**, 557–563.

PROVAN, J. S. 1983. The Complexity of Reliability Computations in Planar and Acyclic Graphs. *SIAM J. Comp.* **15**, 694–702.

REIDERMEISTER, K. 1948. *Knot Theory*. BSC Associates, Moscow, Idaho.

SESHU, S., AND M. B. REED. 1961. *Linear Graphs and Electrical Networks*. Addison-Wesley, London.

TRUEMPER, K. 1989. On the Delta-Wye Reduction for Planar Graphs. *J. Graph Theory* **13**, 141–148.

VALDES, J., R. E. TARJAN AND E. L. LAWLER. 1982. The Recognition of Series Parallel Digraphs. *SIAM J. Comp.* **11**, 298–313.

SUBOPTIMAL POLICIES, WITH BOUNDS, FOR PARAMETER ADAPTIVE DECISION PROCESSES

WILLIAM S. LOVEJOY
Stanford University, Stanford, California

(Received November 1990; revisions received July, November 1991; accepted December 1991)

A parameter adaptive decision process is a sequential decision process where some parameter or parameter set impacting the rewards and/or transitions of the process is not known with certainty. Signals from the performance of the system can be processed by the decision maker as time progresses, yielding information regarding which parameter set is operative. Active learning is an essential feature of these processes, and the decision maker must choose actions that simultaneously guide the system in a preferred direction, as well as yield information that can be used to better prescribe future actions. If the operative parameter set is known with certainty, the parameter adaptive problem reduces to a conventional stochastic dynamic program, which is presumed solvable. Previous authors have shown how to use these solutions to generate suboptimal policies with performance bounds for the parameter adaptive problem. Here it is shown that some desirable characteristics of those bounds are shared by a larger class of functions than those generated from fully observed problems, and that this generalization allows for iterative tightening of the bounds in a manner that preserves those attributes. An example inventory stocking problem demonstrates the technique.

A parameter adaptive decision process is a sequential decision process where some parameter or parameter set impacting the rewards and/or transitions of the process are not known with certainty. Signals from the performance of the system can be processed by the decision maker as time progresses, yielding information regarding which parameter set is operative. In the control theory literature, a control policy for problems with incomplete information is called *adaptive* if it incorporates new information dynamically through time. Hence, seeking an optimal adaptive policy with unknown system parameters is a *parameter adaptive* problem. Active learning is an essential feature of these processes, and the decision maker must choose actions that simultaneously guide the system in a preferred direction, as well as yield information that can be used to better prescribe future actions (this feature is called *dual control* in the control theory literature).

In this paper all relevant sets, including the set of possible parameter sets, are assumed to be finite, and it is assumed that the problem can be cast so that rewards, transitions, and messages have a Markov property. A parameter adaptive decision process satisfying these assumptions is closely related to a finite partially observed Markov decision process, or POMDP. A POMDP is a generalization of a Markov decision process (MDP) that allows for noise-corrupted information regarding the state of the system. See Heyman and Sobel (1984) for a discussion of MDPs, Bertsekas (1976) for an introduction to POMDPs, and Monahan (1982) for a review of POMDP applications. The intuition and results derived by two early works significantly influence the results below: Smallwood and Sondik (1973) for the structure of finite POMDPs, and Van Hee (1978) for suboptimal control in the parameter adaptive context.

POMDPs are theoretically equivalent to MDPs with state-spaces equal to the set of all probability distributions on the partially observed states. However, this theoretical result does not translate into practical solution methods, for reasons described below. Exploiting special structure is currently the key to solving large POMDPs. Bertsekas reviews results for POMDPs with linear dynamics, quadratic reward functions, additive stochastic noise, and unconstrained actions, for which closed-form solutions are available. In the general case, some analytical results are available for systems with only two partially observed states. General POMDPs with more than two partially observed states are more difficult to solve. Despite some significant recent advances, general POMDPs with more than about 20 partially observed states are currently intractable. See Lovejoy (1991b) for a survey of currently

Subject classifications: Decision analysis: Bayesian dynamic programming. Dynamic programming: parameter adaptive decision processes. Inventory/production: policies under uncertainty.
Area of review: STOCHASTIC PROCESSES AND THEIR APPLICATIONS.

0030-364X/93/4103-0583 $01.25