Available online at www.sciencedirect.com
SCIENCE DIRECT®
Theoretical Computer Science 315 (2004) 191–208
Theoretical Computer Science
www.elsevier.com/locate/tcs

Minimal length test vectors for multiple-fault detection

Z. Füredi[a], R.P. Kurshan[b,*]

[a] *Department of Mathematics, University of Illinois at Urbana-Champaign, Urbana, IL 61801-2975, USA*
[b] *Cadence Design Systems, Murray Hill, NJ 07974, USA*

Abstract

A methodology for circuit testing is proposed for detecting multiple circuit faults in the course of a minimal length "guided tour" of the circuit transition structure. Deriving a test vector to guide this tour through an n state subsystem with at most I inputs possible in situ at each state, corresponds to solving an open tour multigraph version of the "Chinese Postman" problem, in which out-degrees are bounded by I. In this case, the length L of a minimal length open tour is shown to satisfy $L \leqslant In^2$; a minimal length open tour is computable in $O(n^3 + nI)$ steps for undirected multigraphs and $O(n^3 + (nI)^2 \log n/\log I)$ steps for directed multigraphs, both one-time costs, using weighted matching and bipartite weighted matching, respectively. An open tour can result in a test vector as much as $\frac{1}{2}$ shorter than the test vector associated with a closed tour, without any loss in error detection. Examples show that for a directed graph, the length of a minimal length open tour may be as great as $n^3/6$ for $I = n$, or $\Omega(n^2)$ when I is bounded, while in an undirected multigraph, a minimal length tour requires no more than $n - 3$ repeated state transitions. This mitigates in favor of "mixed" circuits in which certain transitions are reversible and need be tested in only one direction.

The practicality of this approach rests with the ability to apply it separately to small subsystems, in conjunction with symbolic testing of inter-subsystem coordination. The former is feasible with existing commercial technologies, such as electron beam scanning, while the latter is feasible with a finite-state model-checker.

In summary, the proposed methodology comprises three steps:

1. decompose a circuit into subsystems sufficiently small to be model-checked exhaustively;
2. perform symbolic tests of inter-subsystem coordination and conclude that if each subsystem is correctly implemented, then the entire circuit will behave as required;
3. for each circuit subsystem, exercise every realizable transition through a minimal length (open) tour, comparing the actual transitions with those of the specification.

Keywords: Chinese Postman; Testing; Fault detection

* Corresponding author.
E-mail address: rkurshan@cadence.com (R.P. Kurshan).

0304-3975/$ - see front matter © 2003 Elsevier B.V. All rights reserved.
doi:10.1016/j.tcs.2003.11.018

1. Introduction

The urgent need for more reliable, more efficient procedures to test complex synchronous digital hardware, especially VLSI, is now widely recognized [7,21]. A good survey of current circuit-testing methods may be found in [25]. With few exceptions, current testing methods test only for single "stuck-at" faults (a fault in a wire-segment characterized by a fixed logical value on that wire-segment, independent of whatever value is applied to it). While this often has proved adequate for small- and medium-scale nMOS integration, large-scale integration and utilization of CMOS technology have rendered this single stuck-at fault detection standard less than satisfactory, with many significant faults evading detection [13,25]. Furthermore, in large sequential circuits, the problem of "controllability and observability" of the circuit state (i.e., inherent limitations in the ability to set and observe all of a circuit's memory elements) can make the detection even of single stuck-at faults intractable [2]. This problem may be circumvented through "scan" designs such as "LSSD" and "scan/set logic" [25] wherein the entire memory (*total* state) of a circuit may be operated in a shift-register mode, allowing the circuit's total state to be read or written after any clock cycle, by sequentially shifting the memory in and out of the circuit. This "solves" the single stuck-at fault detection problem, in the sense that total controllability and observability of the circuit state permits application of Roth's *D*-algorithm [2], which detects any single stuck-at fault in a combinatorial circuit. However, there are three serious drawbacks to this approach, namely the (effective) inability to detect more general faults than single stuck-at faults (cf. above), the additional in-circuit hardware needed for scanning (up to a 20% increase [25]) and the time-consuming procedure of shifting states in and out of the circuit during testing. Other fault detection methods such as signature analysis [24], syndrome testing [17] and autonomous testing [12] can detect more general faults, but this is typically at the expense of fault localization (the ability to locate the source of a fault, especially in the presence of feedback loops [21,25]), further increased overhead in hardware and delay, and an uncertainty about the proportion of faults which are thus detected (e.g., signature analysis detects even single stuck-at faults with only small probability in programmable logic arrays and other circuits with large "fan-in" [21]). The simple and (hence) popular method of comparing the output of the circuit under test with the output of a "known correct" circuit or prototype, both exercised by identical randomly generated input sequences, generally does not provide a viable testing method, for reasons of inefficient and hard-to-quantify fault detection coverage [25] (for certain microprocessor designs, this method has been found to uncover only 20–30% of single stuck-at faults [7, p. 441]; cf. also [3]). Furthermore, there is significant uncertainty whether the "known correct" circuit is actually correct.

The proposed methodology can be summarized as follows. In classical fashion, the circuit is described as a finite state machine, e.g., [2] or Ref. [7]. However, the machine is first decomposed into a number of interacting machines, each of which is small enough to be searched exhaustively [10]. Once each subsystem has been shown to conform to its formal specification, the behavior of the system as a whole can be guaranteed by symbolic analysis of the coordination among the subsystems. Electron

beam technology can be used to exhaustively search each subsystem by means of a minimal length open tour. Tours have been used before in several applications, for example in [22] closed tours were proposed for protocol testing. In this paper, minimal length open tours are introduced, analyzed and shown to be twice as efficient as closed tours in this application.

2. Multiple-fault detection with electron beam scanning

We propose a general fault-detection method for synchronous circuits. It could be applied in practice using, for example, electron beam scanning, or any technology that can render all internal circuit states observable, and certain specially prepared latches controllable [18,26]; such technology is commercially available [4,19].

Our method entails logical isolation of pre-designated sequential subsystems, via electron-beam-programmable non-volatile latch/switches [18]. These switches are introduced into the circuit so that in the resulting partition of the circuit into subsystems, each subsystem is sufficiently small to permit exhaustive "in situ" testing (see below). The switches provide subsystem isolation and furthermore serve as subsystem input latches during testing. Switch placement is configured in a way that maintains the integrity of circuit timing [1]. The subsystems thus isolated are tested using electron beam switching of the subsystem input latches and electron beam sensing of the (total) state of the subsystem (while keeping the subsystem logically disconnected from the rest of the circuit) [18]. The sequential progression of (total) subsystem states is compared with that from a symbolic prototype whose behavior has been proved correct using formal, automated methods [10]; these formal methods have been implemented into a software system called COSPAN [8]. The derived testing procedure thus consists of sequentially writing input words from a fixed, predetermined list into the isolated subsystem, reading out the associated sequence of subsystem states, and comparing this state sequence with a fixed, predetermined list of states generated (once, for all tests) from the prototype. The subsystem tests "fault-free" if and only if the two state sequences are identical. Each write and associated read is accomplished in one clock cycle.

Under the proposed method, subsystems are designated so as to have a state space which is sufficiently small to permit exercising of all "circuit-reachable" subsystem state transitions. A subsystem state transition is *circuit-reachable* provided it can occur in situ, that is, in the context of the operation of the entire circuit. When the subsystem is viewed through a classical paradigm for sequential devices (Fig. 1), the total state of the subsystem has two vector components: v and x. The component x represents the subsystem electron-beam-programmable input while the component v represents the subsystem *internal* state, components from which produce the subsystem output to the rest of the circuit; (v,x) is the *total* state of the subsystem.

During normal (non-test) operation, the subsystem communicates with the rest of the circuit through its input x and the output components of its internal state v. In principle, if x is a binary vector of length k, for each value of v there are 2^k possibilities for x. However, due to coordination between the given subsystem and the rest of the circuit,

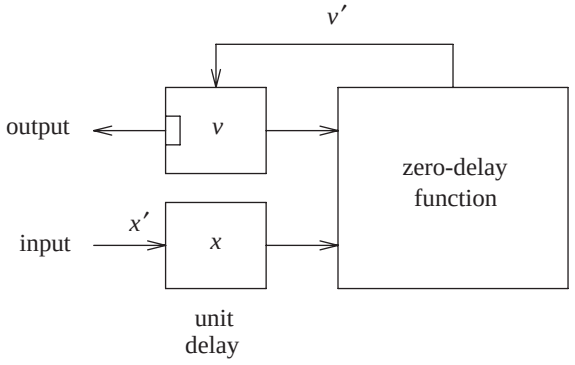


Fig. 1. The total state (v,x) of a subsystem at time i determines the internal state v' at time $i+1$, while the input x' at time i determines the value of the "input latch" at time $i+1$.

it may be that at a given v only $m<2^k$ values for x are ever actually possible, as inputs from the rest of the circuit. Limiting subsystem tests to those m values of x actually possible at each respective internal state v (as a function of v), is what is meant by "in situ" testing, and we will say that such an x is "possible in situ at v"; the transitions of the form $(v,x)\to(v',x')$ where x and x' are inputs possible in situ at v and v', respectively, are the *circuit-reachable* total state transitions.

Restriction of subsystem tests to circuit-reachable transitions can greatly reduce the number of tests required to exhaustively test the subsystem. The values of the inputs x possible in situ at each respective v are determined empirically in the prototype. Experience [10] shows that for properly designated subsystems, the average taken over v of the number of values of x possible in situ at v, may be much less than 2^k (where k is the dimension of x); thus, this approach can result in a significant saving in testing time. Specifically, when a subsystem implements high level control logic, it is typical for only a relatively few different input vectors x to be possible in situ at any given internal state v. On the other hand, when a subsystem implements general memory or an arithmetic unit, the resulting reduction may not be significant.

Testing all circuit-reachable subsystem state transitions has the effect of detecting and locating all subsystem behavioral faults. (While certain stuck-at faults may remain, these will not affect the required behavior of the subsystem.) It is proved symbolically in the prototype that the circuit will be behaviorally correct provided all of its subsystems are correctly implemented. Thus, interactions among the subsystems need not be tested in the implemented circuit, aside from verification that certain "design rules" have been correctly implemented. These design rules, which govern the interface interactions among the subsystems, are formally characterized in the course of symbolic analysis of the prototype. Examples of such design rules which must be checked in the implementation are generally limited to features which may be checked relatively easily through "parameter tests" [7] and include control of propagation delay, setup time and the like.

The problem of fault detection is thus reduced to a particularly simple form of the problem of finite state machine distinguishability. In general, as is well-known [14], if a bound on the actual number of states in the (possibly faulty) circuit under test is not known, it is not always possible to determine whether the circuit under test is faulty. On the other hand, if the circuit under test is known to have at most k states while the prototype is reduced and has $n \leqslant k$ states, it is possible to determine whether or not the circuit under test has any behavioral fault through the application of $O(n^2\, 2^{k-n})$ test vectors of length $O(n^2\, k 2^{k-n})$ [23, Theorem 1]; if $k = n$ and the prototype is strongly connected (and reduced), this test may be accomplished by a single test vector of length $O(n^4 \ln n)$ [23, Theorem 2] when only a component of the total state is observable. (An n state unreduced prototype may be reduced in time $O(n \log n)$ [9].)

We consider here the case of a strongly connected subsystem prototype and a circuit under test whose total state is observable and whose state space may be of arbitrary size in the presence of faults, but is identical to that of the prototype when fault-free. In this case, all behavioral faults in the circuit under test are detected by a single test vector which guides a "tour" of all circuit-reachable total states. This corresponds to testing every input possible in situ at every circuit-reachable internal state, checking that the resulting internal state transition is correct and that applied inputs are correctly "latched up" by the input latches. In the course of the tour, inputs are applied at successive internal states. Specifically, a tour of length L may be described as a vector

$$(v_0, x_0), \ldots, (v_L, x_L)$$

where $x_0, \ldots, x_L$ are input vectors possible in situ at respective internal states $v_0, \ldots, v_L$, where $\{(v_i, x_i) \mid 0 \leqslant i \leqslant L\}$ comprises the entire set of circuit-reachable total states and where, for $0 \leqslant i < L$, the correct internal state transition under application of the input x_i at the internal state v_i, is to v_{i+1}. The necessity of testing in the manner of a tour comes from the nature of the proposed electron beam testing methodology, wherein internal circuit states, while observable, are not directly controllable (i.e., internal states cannot be directly set externally). Thus, from a given state, the next possible internal circuit state is one which is adjacent to (i.e., one state transition away from) the given state. In particular, it is not generally possible to test consecutively several different inputs at one internal state v; as soon as one of the inputs causes an internal state change, the next of the several inputs can be applied to v only when the internal state of the circuit has returned to v.

Since each fabricated circuit may be subject to testing by the derived tour-guiding test vector, it can be of significant importance to derive such a test vector of minimal possible length. In Section 3, we describe how to derive such a minimal length tour-guiding test vector. If the subsystem prototype has n internal (circuit-reachable) states and I is the maximum number of inputs possible in situ at any internal state, we show that the length L of a minimal length test vector satisfies $L \leqslant In^2$. If each distinct input possible in situ at a given internal state results in an internal state transition to a distinct state, we say the subsystem circuit *differentiates* inputs. In this case, clearly $I \leqslant n$ and the given bound on L may be written as $L \leqslant n^3$ (independent of I).

This is an upper bound and depending upon the circuit, the minimal length test vector may be much shorter. However, we give examples which show that a circuit

can require a test vector of length $L > n^3/6$ (for $I = n$) and, even if I is bounded as a function of n, there are circuits which require test vectors of length $L = \Omega(n^2)$. (In Ref. [15], it is proposed to detect stuck-at faults through generation of a tour of each state transition, using random inputs to guide the tour and discarding redundant cycles to reduce the tour length. It is contended there that for any given n state machine, this method produces a tour of length $O(n \log n)$ "on the average"; this contention appears to be without foundation, at least for those circuits which we show have minimal tour length $L = \Omega(n^2)$.)

In some circuits there may be certain cases of an internal state transition $v \to w$ caused by an input x possible in situ at v, which is *reversible* in the sense that some input y possible in situ at w causes the state transition $w \to v$ and furthermore, if the circuit performs either $(v,x) \to w$ or $(w,y) \to v$ correctly, then it may be presumed that it could perform the other test correctly as well. For example, if x is a vector, one component of which switches a simple latch while the other components leave the rest of the circuit invariant, it may be enough to test the latch by showing either that it "sets" from "clear" or "clears" from "set". When one or more state transitions may be reversible, we say the circuit is *mixed*; if all state transitions are reversible, we say the circuit is *undirected*, while if no state transitions are reversible, we say that the circuit is *directed*. In a mixed circuit, the definition of a tour (above) is relaxed to allow $\{(v_i, x_i) \mid 0 \leqslant i \leqslant L\}$ to include only one of (v,x) or (w,y) for each reversible pair of total states. This can result in a considerable saving in the length of a minimal length tour; in an undirected circuit, it is shown that a minimal length tour may be found with at most $n-3$ transitions in excess of the number of circuit-reachable total states, counting reversible pairs as only one state. By contrast, in a directed circuit, a minimal length tour may require as many as $\Omega(n^3)$ transitions in excess of the number of circuit-reachable total states. (This "excess" is the number of redundant transition tests required by the constraint that the tests be performed in the course of a tour.)

The cost of computing a minimal length tour-guiding test vector is shown to be that of finding a solution to an open tour, multigraph version of the "Chinese Postman" problem [5]. It is seen that in all cases (directed, mixed, undirected), a tour may be found in $O(n^3 + (nI)^2 \log n / \log I)$ steps using one form or another of "weighted matching" (a one-time cost for all associated circuit tests); in the directed and undirected cases, this algorithm can be used to find an (open) tour of minimal length, the undirected case requiring only $O(n^3 + nI)$ steps. An open tour can result in a saving of as much as $\frac{1}{2}$ of the test vector length of a minimal length closed tour, in both the directed and undirected cases. (The potential for such saving is lost if the initial state v_0 of all physically possible tours is fixed and if furthermore, from each circuit-reachable state v there is a "reset" input x possible in situ at v which causes the transition $(v,x) \to v_0$. However, typically, a "reset" entails propagation of the reset signal through several states, and the potential for saving with an open tour is actual.) If the initial state v_0 for any tour is given, then a minimal length tour is found subject to that constraint.

The assumption of strong connectivity in the prototype is trivially satisfied by any circuit design which includes a ("power-up") reset or clear, and thus this requirement is trivially satisfied in practice. Adding minimal memory to an initial design to effect

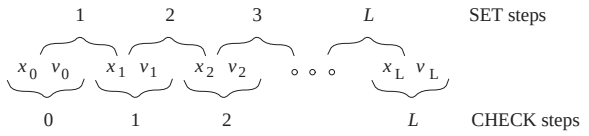


Fig. 2. The test vector of length L, illustrated here, is applied to the circuit through an alternating succession of CHECK and SET steps. During each SET step, the input latches are given a new value and the circuit internal state, with the old input value, is advanced one cycle. During each CHECK step, the value of the resulting circuit total state is checked.

subsystem isolation in such a way that consistent timing is maintained in the circuit can be accomplished in time quadratic in the number of subsystem nodes [1] (assuming constant complexity of enabling predicates at logical branch-points). Symbolic analysis of the prototype may be accomplished by a search algorithm which is linear in the number of system state transitions, in conjunction with applicable complexity-reduction techniques [10]. (All of the preceding are one-time costs.) Observability of the total state of the circuit subsystem under test can be guaranteed by electron-beam scanning [18]. (This is in contrast with conventional "shift-register" scanning methods such as LSSD, wherein certain types of faults may introduce extraneous memory, whose state component is thus unobservable [25].) Once a faulty state transition is located in the circuit under test, the cause of the fault is relatively easy to locate using conventional techniques for fault location in combinatorial circuits [2], since a test for the fault is given by the input vector associated with the faulty transition.

The proposed design for testability and specific circuit test is now summarized. During the design stage, special electron-beam-programmable latches are introduced into the circuit in a fashion which maintains the consistency of circuit timing and has the effect of partitioning the circuit into subsystems which are sufficiently small to be tested exhaustively. Formal methods are then used to prove that the entire circuit will perform properly, assuming proper implementation of each subsystem. These same methods are used to enumerate "design rules" (concerning mainly relative constraints on delay) necessary for the proper interface coordination of the subsystems. Proper implementation of each subsystem is checked by testing each implemented subsystem circuit in turn. Each subsystem circuit is tested by a single test vector which guides a minimal length tour of the subsystem transition structure. By the end of the tour, every input which is possible in situ at every respective circuit-reachable internal state has been applied, thus guiding the subsystem circuit through every possible internal state transition generated by every possible input.

The structure of this test is depicted in Fig. 2. The precomputed test vector $v_0, x_0, v_1, x_1, \ldots, v_L, x_L$ is applied to the subsystem circuit through successive, alternating CHECK and SET steps. During the ith CHECK step ($0 \leqslant i \leqslant L$), the input latches are checked (through electron beam scanning) to be in state x_i and the internal state of the circuit is likewise checked to be v_i. The circuit then is advanced one clock cycle, which corresponds to application to the circuit of input x_i at internal state v_i, and at the same time the input latches are set to the value x_{i+1}. This is the $(i+1)$st SET step.

It remains to show how to compute a minimal length tour-guiding test vector, for a mixed circuit. The problem and its solution may be recast in terms of finding a minimal length tour in a "mixed" multigraph (see the following section). The multigraph vertices correspond to the internal states of the given subsystem, while each multigraph edge from vertex v to vertex w corresponds to an input possible in situ at v which results in a transition (in the prototype) to w. In the case of a reversible pair $(v,x) \leftrightarrow (w,y)$, the directed edges (v,w) and (w,v) corresponding to x and y respectively, are paired to form a single undirected edge. If a circuit differentiates inputs, the associated multigraph problem reduces to a graph problem.

It may be worthwhile to observe that the same testing methodology can be applied to software testing, and this could be meaningful if input control is limited (and "print" commands take the place of electron beam scanning). In this direction, a proposal is given in [22] for high-level conformance testing of communication protocols through a (closed) tour of the entire state space (presuming the protocol differentiates inputs).

3. Minimal length open tour of a multigraph

A ("mixed") *multigraph* G is an ordered pair (M_G, U_G), where M_G is a square matrix of non-negative integers, the *directed adjacency matrix* of G, and U_G is the *undirected adjacency function* of G, described below. The *vertices* of G are the indices of the rows (or columns) of M_G, presumed finite, the set of which is denoted by $V(G)$. If $v,w \in V(G)$ and the (v,w)th element of M_G, denoted by $M_G(v,w)$, satisfies $M_G(v,w) > 0$, then (v,w) is said to be a *directed edge* of G, with *multiplicity* $M_G(v,w)$; the set of directed edges of G is denoted by $E(G)$. U_G is any function defined on the set of unordered pairs of vertices of G, into the non-negative integers. If $U_G(e) > 0$ then e is said to be an *undirected edge* of G, with *multiplicity* $U_G(e)$; the set of undirected edges of G is denoted by $E^0(G)$. If $E^0(G) = \emptyset$, G is called *directed* while if $E(G) = \emptyset$, G is called *undirected*. If G is directed (undirected), we may write $M_G(U_G)$ in place of G. A *graph* is a multigraph, all of whose edges have multiplicity 1. The *content* of a multigraph G is

$$\kappa(G) \equiv \sum_{e \in E(G)} M_G(e) + \sum_{f \in E^0(G)} U_G(f),$$

the number of edges, directed and undirected, counting multiplicities. If G is a graph then $\kappa(G) = |E(G)| + |E^0(G)|$.

A multigraph H is a *subgraph* of G, written $H \subset G$, if $V(H) = V(G)$ and for all $v,w \in V(G)$, $M_H(v,w) \leqslant M_G(v,w)$ and $U_H(\{v,w\}) \leqslant U_G(\{v,w\})$. If $H \subset G$, then $H + G \equiv (M_H + M_G, U_H + U_G)$, where $M_H + M_G$ is the usual matrix addition and likewise, $U_H + U_G$ is defined by pointwise addition. For any multigraph G, let G^0 be the undirected multigraph defined by $V(G^0) = V(G)$, $U_{G^0}(\{v,w\}) = M_G(v,w) + M_G(w,v) + U_G(\{v,w\})$ for all $v,w \in V(G)$; let G^* be the directed multigraph defined by $V(G^*) = V(G)$, $M_{G^*}(v,w) = M_G(v,w) + U_G(\{v,w\})$ for all $v,w, \in V(G)$. Note that $\kappa(G^0) = \kappa(G) \leqslant \kappa(G^*)$.

A *path* of *length* L in a multigraph G is a vector $\mathbf{v}=(v_0,\dots,v_L)$ of $L+1$ vertices of G with the property that for $0\leqslant i<L$, either $(v_i,v_{i+1})\in E(G)$ or $\{v_i,v_{i+1}\}\in E^0(G)$. The path $\mathbf{v}$ is said to be *from* v_0 *to* v_L, and we write $L(\mathbf{v})=L$. A path $\mathbf{v}$ of length L in M is a *tour* of G if for each $e\in E(G)$ there are at least $M_G(e)$ distinct values of i, $0\leqslant i<L$, for which $(v_i,v_{i+1})=e$, and in addition to all these, for each $e\in E^0(G)$, there are at least $U_G(e)$ (additional) distinct values of i, $0\leqslant i<L$, for which $\{v_i,v_{i+1}\}=e$. The *edges* of $\mathbf{v}$ are the ordered pairs (v_i,v_{i+1}), $0\leqslant i<L$, the set of which is denoted by $E(\mathbf{v})$. (Note that $E(\mathbf{v})$ may not be a subset of $E(G)$.) The *multiplicity* of $e\in E(\mathbf{v})$ is the number of distinct values of i, $0\leqslant i<L$, for which $(v_i,v_{i+1})=e$. The path $\mathbf{v}$ is *closed* if $v_L=v_0$; otherwise $\mathbf{v}$ is *open*. A *cycle* is a closed path; a multigraph is *acyclic* if it admits of no cycles. A path is *simple* if each of its edges has multiplicity 1. A multigraph G is *strongly connected* if for each $v,w\in V(G)$ there is a path in G from v to w, or equivalently, if G admits of a closed tour; G is *connected* if for each v,w there is either a path from v to w or a path from w to v. Clearly, an undirected multigraph is strongly connected if and only if it is connected. If $\mathbf{v}$ is a tour such that for each pair $v,w\in V(G)$, the multiplicity m of (v,w) in $\mathbf{v}$ and the multiplicity n of (w,v) in $\mathbf{v}$ satisfy $a\equiv m-M_G(v,w)\geqslant 0$, $b\equiv n-M_G(w,v)\geqslant 0$ and $a+b=U_G(\{v,w\})$, then $\mathbf{v}$ is an *Eulerian* tour (which in this paper may be *open* as well as closed). For any path $\mathbf{v}$ in G let $\Gamma_G(\mathbf{v})$ be the directed multigraph whose edges and multiplicities are those of $\mathbf{v}$. Note that a tour $\mathbf{v}$ of G is an Eulerian tour of both $\Gamma_G(\mathbf{v})$ and $\Gamma_G^0(\mathbf{v})(\equiv(\Gamma_G^0(\mathbf{v}))^0)$, $M_G\subset\Gamma_G(\mathbf{v})$ and $G^0\subset\Gamma_G^0(\mathbf{v})$.

For $v\in V(G)$ the *out-degree* of v is

$$d_G^+(v)\equiv\sum_{w\in V(G)}M_G(v,w),$$

the *in-degree* of v is

$$d_G^-(v)\equiv\sum_{w\in V(G)}M_G(w,v),$$

and the *undirected degree* of v is

$$d_G^0(v)\equiv\sum_{w\in V(G)}U_G(\{v,w\}).$$

The *parity* of v is the parity of the integer $d_G^0(v)$; v is *symmetric* if $d_G^+(v)=d_G^-(v)$. Let

$$L(G)=\inf\{L\,|\,\text{there exists a tour of } G \text{ of length } L\}.$$

Note that if Z is a directed multigraph formed from a directed cycle on n vertices by duplicating exactly one edge (so as to give it multiplicity 2), then $L(Z)=n+1$ whereas every closed tour of Z has length at least $2n$. Similarly, an undirected linear graph l on $n+1$ vertices satisfies $L(l)=n$, while every closed tour of l has length at least $2n$. If G is strongly connected, every $v\in V(G)$ is even and $d_G^+(v)=d_G^-(v)$ for all $v\in V(G)$, then G admits of a closed Eulerian tour [11, 5.9.6], and conversely;

G admits of an Eulerian tour if and only if $L(G)=\kappa(G)$ (the content of G, defined above).

Now, let G be a fixed strongly connected multigraph with $n=card\ V(G)$, and let

$$I=\max\{d_G^+(v)+d_G^0(v)\mid v\in V(G)\}.$$

Note that

$$\sum_{e\in E(G)} M_G(e)+2\sum_{e\in E^0(G)} U_G(e)=\sum_{v\in V(G)}\sum_{w\in V(G)} M_G(v,w)+U_G(\{v,w\})$$
$$\leqslant \sum_v I=nI.$$

Proposition 1. $L(G)\leqslant In^2$.

Proof. Let $M=G^*$. Note that $I=\max\{d_M^+(v)\mid v\in V(M)\}$. For each $e\in E(M)$ let $\mathbf{z}_e$ be a simple cycle containing e (which exists because of the assumed strong connectivity of G); let $Z_e=\Gamma_G(\mathbf{z}_e)$. Set $Z=\sum_{e\in E(M)} M(e)Z_e$; then $M\subset Z$, $E(M)=E(Z)$, Z is strongly connected and $d_Z^+(v)=d_Z^-(v)$ for all $v\in V(Z)$. Hence, there is a closed Eulerian tour $\mathbf{v}$ of Z, and this constitutes a (closed) tour of M. Since $\mathbf{v}$ is an Eulerian tour of Z, $L(\mathbf{v})=\sum_{f\in E(Z)} Z(f)=\sum_{f\in E(Z)}\sum_{e\in E(M)} M(e)Z_e(f)=\sum_e M(e)\sum_f Z_e(f)\leqslant \sum_e M(e)n\leqslant In^2$. □

Example 1. Let G_n be the n vertex directed graph defined by

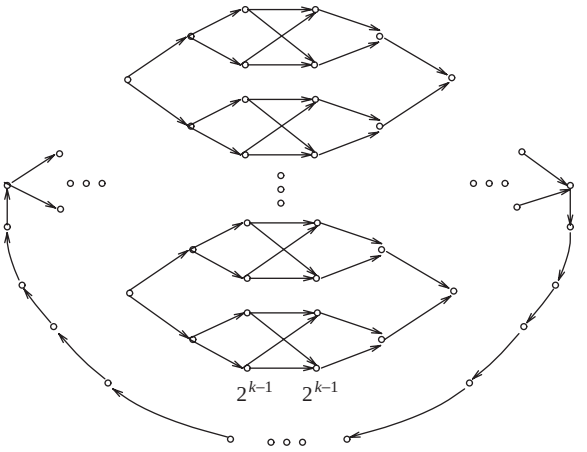
$$M_{G_n}(i,j)=\begin{cases}1 & \text{if } 1\leqslant j<1 \text{ or } j=i+1,\\ 0 & \text{otherwise}\end{cases}$$

and pictured below.

Say $(i,j)\in E(G_n)$ is a *backedge* if $j<i$. Let $\mathbf{v}$ be a tour of G_n. Then $\mathbf{v}$ traverses each backedge of G_n. In order to traverse the $i-1$ backedges emanating from vertex i, $\mathbf{v}$ must also traverse the edge $(j,j+1)$ j times for $1\leqslant j<i-1$ and the edge $(i-1,i)$ $i-2$ times, since intervening backedges in $\mathbf{v}$ from other vertices cannot decrease this repetition. Thus, for $n>2$,

$$L(G_n)=\sum_{i=2}^{n}\left[(i+1)+\sum_{j=1}^{i-2} j+(i-2)\right]=n(n+1)(n+2)/6-2n+1.$$

Example 2. Let G_n be the n vertex directed graph pictured below:



If the "expansion" part of G_n "expands" to 2^{k-1} vertices and then "contracts" symmetrically, while the linear portion of G_n (the long branchless chain at the bottom of the figure) contains 2^k-1 vertices, then $n=3(2^k-1)$. Note that each of the 2^k edges of the linear portion of G_n must be traversed once for each of the 2^k paths through the upper portion. Thus, while $d^+_{G_n}(v)\leqslant 2$ for all $v\in V(G_n)$, $L(G_n)>n^2/9$.

For a strongly connected multigraph G, let

$$\tau(G)\equiv L(G)-\kappa(G),$$

the *tour index* of G. This is the total number of edge traversals in excess of the respective edge multiplicities required by any tour of G. The preceding examples show that in general, the maximum value of $\tau(G)$ over all multigraphs with n vertices $\Omega(n^3)$, or $\Omega(n^2)$ if I is bounded. However, for undirected multigraphs, the tour index is substantially smaller, as the next proposition shows.

A *forest* is an undirected acyclic graph.

Lemma 1. *Let U be an undirected multigraph. Then there is a forest $F\subset U$ such that every vertex of $U+F$ is even. Such an F with fewest possible edges may be found in* $O(|V(U)|^3)$ *steps.*

Proof (Edmonds and Johnson [5]). Since $\sum_{v\in V(U)} d^0_U(v)=2\kappa(U)$, there are an even number of odd parity vertices of U, and the same applies for every connected component of U. Let these odd vertices be partitioned into pairs $\{v_1,w_1\},\{v_2,w_2\},\ldots$ such that the sum Z of the lengths of respective shortest paths P_i from v_i to w_i, $i=1,2,\ldots$ is as small as possible, and set $F=\sum_i \Gamma^0_U(P_i)$. If there are $2n$ odd vertices, then the length of the shortest path between each pair of them may be found in $O(n^3)$ steps using Dijkstra's algorithm (e.g., [20, Section 7.2]) and an optimal partition into pairs may be found using Edmonds' weighted matching algorithm (cf. [5, Section 3]),

in another $O(n^3)$ steps (cf. [20, Section 9.4]). If for $i \neq j, P_i$ is from v to w, P_j is from v' to w' and $\Gamma_U^0(P_i)$ and $\Gamma_U^0(P_j)$ share an edge e, then there is a path Q from v to v' (or w') and a path Q' from w to w' (or v') each of which avoids e, with the result that $L(Q) + L(Q') = L(P_i) + L(P_j) - 2$, contradicting the assumed minimality of Z. Thus, $F \subset U$. Since the odd vertices of F are precisely the endpoints of each P_i (which comprise the odd vertices of U), the parity of each $v \in V(F)$ is the same relative to F and U. Hence, every vertex of $U+F$ is even. If F contained a cycle, then its elimination would not affect the parity of the vertices of F, and thus its elimination could not leave any of the odd vertices of F unpaired; but its elimination would decrease the value of Z; thus, F is a forest, and by construction, it has a fewest possible number of edges. □

A forest F as in Lemma 1 is said to be an *evening* forest *for* U, the set of which is denoted by $E(U)$. Clearly, if $\mathbf{0}$ is an evening forest for U, then there can be no other, and $\tau(U) = 0$.

The *diameter* of a multigraph G is

$$\delta(G) = \max_{v,w \in V(G)} \min\{L(\mathbf{v}) \mid \mathbf{v} \text{ is a path in } G \text{ from } v \text{ to } w\}.$$

Proposition 2. *Let U be a connected undirected multigraph. Then*

$$\tau(U) \leqslant \min_{F \in E(U)} \{|E^0(F)| - \delta(F)\},$$

with equality if every $F \in E(U)$ is a tree.

Proof. Suppose U admits of a non-zero evening forest F. Let $\mathbf{v}$ be a simple path in F of length $\delta(F)$, from (say) v to w and let $V = \Gamma_U^0(\mathbf{v}) \subset F$. Let $\mathbf{w}$ be a maximal length path of U satisfying $\Gamma_U^0(\mathbf{w}) \subset U + F$, which starts at v, and from each vertex u of which "uses" an edge $e \in E^0(V)$ only if all edges of $U + F - V$ have already been used, counting multiplicities; in this case, the edge is required to be oriented in the same "direction" (i.e., in $E(\mathbf{v})$ or not) as any edge in $E^0(V)$ previously "used" in this sense (for its last traversal). We claim $\mathbf{w}$ is in fact an open Eulerian tour of $U + F - V$. Indeed, since all vertices of $U + F$ are even, the first edge $e \in E^0(V)$ "used" in $\mathbf{w}$ (in the above sense) will be incident either to v or w (thus determining the direction of all future edges "used" from $E^0(V)$). By the same argument, the order in which the edges of $E^0(V)$ are "used" is either the natural order in which they appear in $\mathbf{v}$, or else the reverse of that order. Since an edge of $E^0(V)$ is "used" only when necessary, and $V \subset F \subset U$, by the time $\mathbf{w}$ is forced to end, all the edges of $U + F - V$ have been traversed exactly once, counting multiplicities. Thus, $\mathbf{w}$ is a tour of U and $L(\mathbf{w}) = \sum_{e \in E^0(U)} U(e) + |E^0(F)| - L(\mathbf{v})$ so $\tau(U) \leqslant |E^0(F)| - \delta(F)$. Now, suppose $\mathbf{w}$ is a tour of U which satisfies $L(\mathbf{w}) - \sum_{e \in E^0(U)} U(e) = \tau(U)$, and let $F = \Gamma_U^0(\mathbf{w}) - U$ (so $\tau(U) = \kappa(F)$). Since $\mathbf{w}$ is a minimal length tour of $U, \mathbf{w}$ is not a cycle (otherwise, the multiplicity of any edge of $\mathbf{w}$ could be reduced). Hence, if $\mathbf{w}$ is from v to w, then $w \neq v$. Let $\mathbf{d}$ be a simple minimal length path in U from w to v. Then $\mathbf{w}$ followed by $\mathbf{d}$ is a closed Eulerian tour of $\Gamma_U^0(\mathbf{w}) + \Gamma_U^0(\mathbf{d}) = U + F + \Gamma_U^0(\mathbf{d})$, so

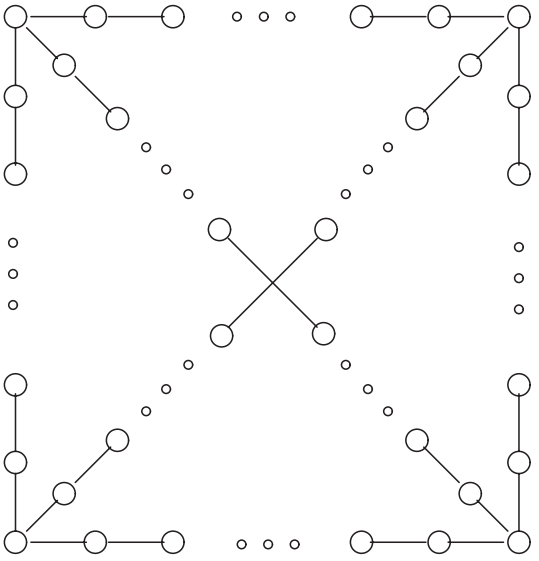
the odd vertices of U are precisely the odd vertices of $F + \Gamma_U^0(\mathbf{d})$. Let $f \subset F + \Gamma^0(\mathbf{d})$ be an evening forest of $F + \Gamma^0(\mathbf{d})$. Then f is an evening forest of U as well, and thus is a tree. Furthermore, since $L(\mathbf{d})$ is minimal, $\delta(f) \geqslant L(\mathbf{d})$. Thus, $\tau(U) = \kappa(F) + L(\mathbf{d}) - L(\mathbf{d}) \geqslant |E^0(f)| - L(\mathbf{d}) \geqslant |E^0(f)| - \delta(f)$. The required equality follows. □

Corollary. *Let U be a connected undirected multigraph with $n \geqslant 3$ vertices. Then $\tau(G) \leqslant n - 3$.*

Proof. If there are no odd vertices then U admits of an Eulerian tour and so $\tau(U) = 0$. Otherwise, let $F \subset U$ be a non-zero evening forest. If $\delta(F) = 1$ then $|E^0(F)| = 1$ so by Proposition 2, again $\tau(U) = 0$. Otherwise, $\delta(F) \geqslant 2$ while $|E^0(F)| \leqslant n - 1$ since F is a forest, so again by Proposition 2, $\tau(U) \leqslant (n-1) - 2 = n - 3$. □

Example 3. Let U be an n vertex undirected "star" graph: $U(\{1, i\}) = 1$ for $1 < i \leqslant n$, $U(\{i, j\}) = 0$ for $i, j > 1$. Then $\tau(U) = n - 3$.

Example 4. The undirected graph U pictured below, on $n+4$ vertices (with $n/6$ vertices per "long edge"), has $d_U^0(v) \leqslant 3$ for every vertex v, yet $\tau(U) = (n/6) + 1$:



The generalization of evening forest in an arbitrary multigraph G is *acyclic complement*, a $|V(G)|$-state acyclic multigraph H such that $E(H) \subset E(G)$, $E^0(H) \subset E^0(G)$ and for each $v \in V(G)$,

$$d^0_{G+H}(v) - |d^+_{G+H}(v) - d^-_{G+H}(v)|$$

is even and non-negative. The set of acyclic complements of G is denoted A (G). The effect of an acyclic complement H of a multigraph G is to produce a multigraph $G + H$ in which, for each vertex v, the number of potential "entrances to" v and

"exits from" v are equal; since a directed edge can be used only for an entrance or only for an exit, any in- or out-going deficiency at v must be compensated by undirected edges, after which the remaining number of undirected edges at v must be even.

Unfortunately, unlike the undirected case in which every minimal content acyclic complement of U is an evening forest $F \subset U$, in the general or directed cases (as Examples 1 and 2 show), edges in an acyclic complement may require higher multiplicity than the same edge in the original multigraph. Furthermore, an acyclic complement exists if and only if each connected component is strongly connected.

We prove first the existence of acyclic complements in the strongly connected directed case. Given a directed multigraph M, a directed $|V(M)|$-vertex acyclic multigraph S is a *symmetric complement* of M if every vertex of $M+S$ is symmetric; the set of all symmetric complements of M is denoted by $\mathrm{S}(M)$; *the deficiency* of a vertex $v \in V(M)$ is

$$d_M(v) \equiv d_M^-(v) - d_M^+(v).$$

Note that since $\sum_v d_M^+(v) = \sum_v d_M^-(v)$, $\sum_v d_M(v) = 0$; thus, for $I = \max\{d_M^+(v) \mid v \in V(M)\}$ and $n = |V(M)| \sum_v \max\{0, d_M(v)\} = -\sum_v \min\{0, d_M(v)\} \leqslant nI$.

Lemma 2. *Let M be a strongly connected directed multigraph. Then $S(M) \neq \emptyset$. For $n = card\{v \in V(M) \mid d_M(v) \neq 0\}$ and $m = \sum_{v \in V(M)} \max\{0, d_M(v)\}$, an element $S \in S(M)$ with $\kappa(S)$ minimal may be found in $\mathrm{O}(n^3 + m^2 \log m)$ steps.*

Proof (Papadimitriou [16]). Let V^+ be a set containing $d_M(v)$ "copies" of v for each $v \in V(M)$ for which $d_M(v) > 0$, and let V^- be likewise for those v with $d_M(v) < 0$. Then $|V^+| = |V^-|$. As in the proof of Lemma 1, use Dijkstra's algorithm to find the respective lengths of shortest paths from each $w \in V^-$ to each $v \in V^+$, in $\mathrm{O}(n^3)$ steps; then let $(v_1, w_1), (v_2, w_2), \ldots$ be a "matching" between V^- and V^+ (i.e., each $v \in V^-$ is equal to exactly one v_i and each $w \in V^+$ is equal to exactly one w_i) so that the sum of the lengths of respective shortest paths P_i from v_i to w_i, $i = 1, 2, \ldots$ (non-vacuous since M is strongly connected) is as small as possible; set $S = \sum_i P_i$. The "Hungarian" bipartite weighted matching algorithm finds such a minimal matching in $\mathrm{O}(m^2 \log m)$ steps [20, Theorem 8.13; Section 9.1]. As in the proof of Lemma 1, S is acyclic. □

Proposition 3. *Let G be an arbitrary strongly connected multigraph. Then $A(G) \neq \emptyset$.*

Proof. Start with a multigraph A_0 such that $M_{A_0} \in \mathrm{S}(M_G)$ *and* $U_{A_0} \in E(U_{A_0})$, *as given by Lemmas* 1 *and* 2. *Then every vertex of $M_G + M_{A_0}$ is symmetric, so for all $v \in V(G)$, $|d^+_{G+A_0}(v) - d^-_{G+A_0}(v)| = 0$,* while every vertex of $U_G + U_{A_0}$ is even. If A_0 is acyclic, then $A_0 \in A(G)$. *If not, let* $\mathbf{v}$ *be a cycle of A_0 and let $Z \subset A_0$ be defined as follows: for each $e \in E(\mathbf{v})$, if $M_{A_0}(e) > 0$ let $M_Z(e) = 1$ and $U_Z(e) = 0$; otherwise, let $M_Z(e) = 0$ and $U_Z(e) = 1$; for all $v, w \in V(A_0)$ with $(v, w) \notin E(\mathbf{v})$, let $M_Z(v, w) = U_Z(\{v, w\}) = 0$. Set $A_1 = A_0 - Z$. By construction, for each $v \in V(G)$, $d^0_{G+A_1}(v) - |d^+_{G+A_1}(v) - d^-_{G+A_1}(v)|$ is even and non-negative, while $\kappa(A_1) < \kappa(A_0)$. In this way we produce a strictly decreasing chain of multigraphs $A_0 \supset A_1 \supset \cdots$ which must terminate (after a finite number m of terms) in a multigraph $A_m \in A(G)$.*

Note: While Proposition 3 gives an algorithm to find an element of A (G) in $O((nI)^3)$ steps, the problem of finding an element of A (G) with *minimal* content is NP-complete, even if G is taken to be planar, with $d_G^0 + d_G^+ + d_G^- \leqslant 3$ [16]. However, in view of Proposition 1 and Example 2, the tour produced using this algorithm (see below), in the worst case has length of the same order as the minimal length tour.

The analog to Proposition 2 for directed multigraphs is given next. The value of τ may be much less favorable in this case, as every symmetric complement of M may fail to be a subgraph of M, as already noted.

Proposition 4. *Let M be a strongly connected directed multigraph. Then*

$$\tau(M) \leqslant \min_{S \in S(M)} \{\kappa(S) - \delta(S)\}$$

with equality when every symmetric complement is connected.

Proof. This is completely analogous to the proof of Proposition 2.

For a general multigraph G, the analogous bound on $\tau(G)$ obtains; however, computationally, the best one may hope for is the somewhat worse second bound. The proof again follows that of Proposition 2, and is omitted. □

Proposition 5. *Let G be an arbitrary strongly connected multigraph. Then*

$$\begin{aligned}\tau(G) &\leqslant \min_{A \in A(G)} \{\kappa(A) - \delta(A)\} \\ &\leqslant \min_{S \in S(M_G)} \{\kappa(S) - \delta(S)\} + \min_{F \in E(U_G)} \{|E^0(F)| - \delta(F)\}.\end{aligned}$$

4. The complexity of finding an optimal open tour

As mentioned above, to find an optimal tour in a mixed graph is an NP-complete problem. However, using powerful known methods, one can find effectively an optimal open tour in a (purely) directed or undirected graph.

Given an arbitrary strongly connected multigraph G with $|V(G)| = n$, the problem of how to find a minimal length tour of G and to compute $\tau(G)$, may be decomposed into three subproblems. The first is to find the lengths of all minimal length paths between ordered pairs of vertices; for this we use Dijkstra's algorithm, requiring $O(n^3)$ steps. The second is to find an acyclic complement $A \in$ A (G) with $\tau(G) = \kappa(A) - \delta(A)$. We will give solutions to this problem in the cases in which G is undirected or directed, using variations of Edmonds' general weighted matching and Hungarian bipartite weighted matching, cited in the proofs of Lemmas 1 and 2, respectively; these variations have the same respective complexities as the original algorithms and apply as well to the constrained problem, in which the tour must begin at a given vertex. This also serves to find an acyclic complement A in the general case, which satisfies the bound given in Proposition 5, although this acyclic complement may not have minimal possible content.

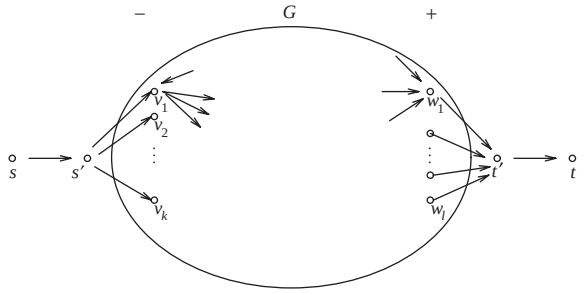
Given an acyclic complement A, a tour of G is constructed, of length $\kappa(G)+\kappa(A)-\delta(A)$. This is done by constructing an (open) Eulerian tour of $G+A-V$, where $\kappa(V)=\delta(A)$, using the variation of the van Aardenne–Ehrenfest/deBruin spanning arborescence algorithm for mixed graphs presented in Ref. [5] (which applies directly to mixed multigraphs as well), in conjunction with the algorithm of Proposition 2; this requires $O(nI)$ steps.

The crucial step, then, is to find an acyclic complement A with $\tau(G)=\kappa(A)-\delta(A)$. We can solve this problem for the cases in which G directed or undirected. (For general G, although this problem is NP-complete, one may find a suboptimal tour by applying this step twice: once for M_G and once for U_G; the resulting tour will satisfy the bound of Proposition 5.) The problem which this step solves may be restated as follows. In the undirected case (cf. the proof of Lemma 1), partition the odd vertices into pairs $\{v_1,w_1\},\ldots,\{v_k,w_k\}$, such that the sum of the lengths of the respective shortest paths from v_i to w_i for $1\leqslant i<k$ is as small as possible. Let us call this the *truncated* weighted matching problem.

Proposition 6. *The truncated weighted matching of 2k points may be solved in* $O(k^3)$ *steps.*

Proof. Run the (non-truncated) weighted matching algorithm [6] through the $(k-1)$th augmentation step, and stop. From the proof of [20, Theorem 8.12], it follows that the corresponding truncated matching problem is solved.

In the directed case, let us denote the vertices of G with negative (positive) deficiencies by $v_1,\ldots,v_k$ ($w_1,\ldots,w_l$, respectively). (If all the vertices are balanced, then G has a closed Euler tour [5].) First, we define a network N (see figure) with source s, sink t on the set $\{s,s',t',t\}\cup V(G)$ with integer capacities and costs in the following way: $\mathrm{cap}(s',v_i)=$ deficiency of v_i, similarly $\mathrm{cap}(w_j,t')=|$deficiency of $w_j|$ $\mathrm{cap}(s,s')=\mathrm{cap}(t',t)=\sum_i \mathrm{cap}(s',v_i)-1$, and all the edges of G have capacity ∞. The cost of an edge of G is 1; the additional edges have cost 0.



Now consider a minimal length open tour $\mathbf{w}$ of G. Let A denote the graph formed by the edges of G with multiplicities $\Gamma_{\mathbf{w}}(e)-G(e)$. We call A a *truncated acyclic complement* of G. Then every vertex of G is balanced in A except one v_i and w_j with deficiencies -1, $+1$, respectively. So for every A, we can associate a flow in the

network N with maximum capacity. Thus, in order to solve the problem in the directed case, it is enough to find a minimum cost flow (with maximum capacity) in N.

Proposition 7. *A truncated acyclic complement of G with minimum number of edges can be found in* $O(n^2I^2 \log n/\log I)$ *steps.*

Proof. In general, in an integer network with non-negative costs and capacities, one can find a minimum cost flow in $O(m|f^*| \log_{(2+m/v)} v)$ steps using the so-called minimum cost augmentation method (see, [20, Theorem 8.13]). Here m is the number of edges in the network, v is the number of vertices and $|f^*|$ is the value of the maximum flow. In our case, $m \leqslant nI$, $|f^*| = -1 + \Sigma$ deficiencies of $v_i \leqslant nI$, so we have $O(n^2I^2 \log n/\log I)$ steps. □

When a minimal length tour is required subject to the constraint that it begins at a given vertex v_0, the associated truncated matching problem is reduced to an ordinary weighted matching problem among $2k-1$ points in the undirected case. The directed case can be handled by a slight modification of N.

Acknowledgements

One of the authors, R.P. Kurshan thanks R.E. Tarjan for pointing him to Edmonds' weighted matching algorithm, and discussing its application to the open postman problem.

References

[1] S.N. Bhatt, F.R.K. Chung, A.L. Rosenberg, Partitioning circuits for improved testability, Proc. 4th MIT Conf. in Advanced Res. in VLSI, 1986, pp. 91–106.
[2] H.Y. Chang, E. Manning, G. Metze, Fault Diagnosis of Digital Systems, Wiley, NY, 1974.
[3] C.K. Chin, E.J. McCluskey, Test length for pseudorandom testing, IEEE Trans. Comput. C-36 (1987) 252–256.
[4] M. Cocito, D. Soldani, Evaluation of the performance of an electron beam testing system based on commercially available subsystems, Scanning Electron Microscopy (1983) 45–54.
[5] J. Edmonds, E.L. Johnson, Matching, Euler tours and the Chinese Postman, Math. Programming 5 (1973) 88–124.
[6] H.N. Gabow, An efficient implementation of Edmonds' algorithm for maximum matching on graphs, J. Assoc. Comput. Mach. 23 (1976) 221–234.
[7] L.A. Glasser, D.W. Dobberpuhl, The Design and Analysis of VLSI Circuits, Addison–Wesley, UK, 1985.
[8] R.H. Hardin, Z. Har'el, R.P. Kurshan, COSPAN, Proc. CAV'96, Lecture Notes in Computer Science, Vol. 1102, 1996, pp. 423–427.
[9] J.E. Hopcroft, An *n*log*n* Algorithm for Minimizing States in a Finite Automaton, in: Z. Kohavi, A. Paz (Eds.), Theory of Machines and Computations, Academic Press, New York, 1971, pp. 189–196.
[10] R.P. Kurshan, Computer-aided Verification of Coordinating Processes: The Automata-Theoretic Approach, Princeton University Press, Princeton, NJ, 1994.
[11] L. Lovász, Combinatorial Problems and Exercises, North-Holland, Amsterdam, 1979.

[12] E.J. McCluskey, S. Bozorgui-Nesbat, Design for autonomous test, IEEE Trans. Comput. C-30 (1981) 866–875.
[13] K.C.Y. Mei, Bridging and stuck-at faults, IEEE Trans. Comput. C-23 (1974) 720–727.
[14] E.F. Moore, Gedanken-experiments on sequential machines, in: C.E. Shannon, J. McCarthy (Eds.), Automata Studies, Annals of Mathematical Studies, Vol. 34, Princeton University Press, Princeton, NJ, 1956, pp. 129–153.
[15] S. Naito, M. Tsunoyama, Fault detection for sequential machines by transition-tours, Proc. 11th Ann. IEEE Internat. Symp. Fault-Tolerant Comput., 1981, pp. 238–243.
[16] C.H. Papadimitriou, On the complexity of edge-traversing, JACM 23 (1976) 544–554.
[17] J. Savir, Syndrome-testable design of combinatorial circuits, IEEE Trans. Comput. C-29 (1980) 442–451.
[18] D.C. Shaver, Techniques for electron beam testing and restructuring integrated circuits, J. Vac. Sci. Technol. 19 (4) (1981) 1010–1013.
[19] K. Smith, Electron beam tests dense VLSI chips, Electronics, April 19, 1984, pp. 86–88.
[20] R.E. Tarjan, Data Structures and Network Algorithms, SIAM, Philadelphia, 1983.
[21] R.E. Tulloss, Automated board testing: coping with complex circuits. IEEE Spectrum, July 1983, pp. 38–43.
[22] M.U. Uyar, A.T. Dahbura, Optimal test generation for protocols: the Chinese Postman algorithm applied to Q. 931, Proc. IEEE Global Telecomm. Conf., December 1986, pp. 68–72.
[23] M.P. Vasilevskii, Failure diagnosis of automata, Cybernetics 9 (1973) 653–665 [translated from Kibernetika 4 (1973) 98–108].
[24] E. White, Signature analysis–enhancing the serviceability of microprocessor-based industrial products, Proc. IECI, 1978, pp. 68–76.
[25] T.W. Williams, K.P. Parker, Design for testability–a survey, Proc. IEEE 71 (1983) 98–112.
[26] E. Wolfgang, R. Lindner, P. Fazekas, H. Feuerbaum, Electron beam testing of VLSI circuits, IEEE Trans. Electron Dev. ED-26 (1979) 549–559.