Parasitic Flow Loops in Networks

M.Todinov

Department of Mechanical Engineering and Mathematical Sciences
Oxford Brookes University, Oxford, OX33 1HX, UK

***Abstract*—** Parasitic flow loops in real networks are associated with transportation losses, congestion and increased pollution of the environment. The paper shows that complex networks dispatching the same type of interchangeable commodity exhibit parasitic flow loops and the commodity does not need to be physically travelling around a closed contour for a parasitic flow loop to be present. Consequently, a theorem giving the necessary and sufficient condition for a parasitic flow loop on randomly oriented source-destination paths in a plane has been formulated and a simple expression has been obtained for the probability of a directed flow loop. A closed-form expression has also been derived for determining the probability of a parasitic flow loop on a fixed lattice with flows whose directions are random. The results demonstrate that even for a relatively small number of intersecting flow paths, the probability of a directed flow loop is very large, which shows that the existence of directed flow loops in large and complex networks is practically inevitable. Consequently, a simple and efficient recursive algorithm has also been proposed for discovering and removing parasitic flow loops in real networks. The paper also shows that for any possible number and for any possible orientation of straight-line flow paths on a plane, it is always possible to choose the flows in the paths in such a way, that no parasitic flow loops are present between the points of intersection.

In this paper, we also raise awareness of a fundamental flaw of algorithms for maximising the throughput flow published since 1956. They all leave highly undesirable parasitic flow loops in the optimised networks and are unsuitable for network optimisation without an additional stage aimed at removing them.

***Keywords* —** parasitic flow loops, flow networks, optimization, classical algorithms, maximising throughput flow.

---

1. ORIGIN OF PARASITIC FLOW LOOPS IN NETWORKS

Parasitic (directed) flow loops in cases where the transported commodity physically travels along a closed contour have already been reported in computer networks, where due to faults in the routing the packets, physically travel along a closed loop (Hengartner et al, 2002; Paxson, 1997). What has not been noticed in previous research, is that parasitic flow loops *could exist even if the transported commodity does not physically travel along a closed contour*. The example in Fig.1 illustrates this critical point.

Suppose that the network in Fig.1a transports interchangeable commodity (for example, the same type and quality of petrol, electricity, gas, chemicals, the same type goods etc.). The throughput capacity of each edge in the network if Fig.1a is 100 units. Assume that the commodity is transported along straight-line paths only, from sources s1,s2 and s3 to destinations t1,t2 and t3. The same type of interchangeable commodity is transported along path (1,2,3,4,5,6), along path (7,8,9,4,10,11,12) and along path (13,14,10,3,15,16). Despite that no commodity physically travels along a closed contour, a directed loop carrying 100 units of flow effectively appears between the intersection nodes 3,4 and 10. Removing 100 units of flow from the directed flow loop (3,4,10,3) turns the flow along edges (3,4), (4,10) and (10,3) into zero, without diminishing the amount of total flow sent from the source nodes to destination nodes (Fig.1b).

It needs to be pointed out that parasitic loops of flow appear even if there are no nodes at the intersections points of the flow paths (Fig.1c). This is for example the case of multi-level intersections of road and railway networks. Despite that the flow cannot be redirected at the intersection points, and essentially the parasitic flow loop ($i1$,$i2$,$i3$,$i1$) cannot be removed, a parasitic flow loop does exist and part of the flow of the network effectively moves around a closed contour. This feature of the parasitic loops has made them difficult to spot and this was probably one of the reasons why for so long they have evaded analysis.

Parasitic flow loops whether removable or non-removable are highly undesirable because: (i) they increase unnecessarily the cost of transportation of the flow in the network, (ii) they consume residual capacity from the edges of the network and (iii) energy is unnecessarily wasted for maintaining them. The presence of parasitic loops of flow in networks causes big financial losses in all affected sectors of the economy. In computer networks, parasitic loops of flow consume bandwidth capacity unnecessarily, increase data traffic and ultimately lead to congestion and delayed data transmission. This affects negatively the quality of service of the network. For supply networks, the existence of parasitic flow loops means high transportation costs, because energy is wasted on circulating commodities unnecessarily.

The probability of existence of a parasitic flow loop between the intersection points of random source-destination paths has not yet been considered in the literature, despite its importance.

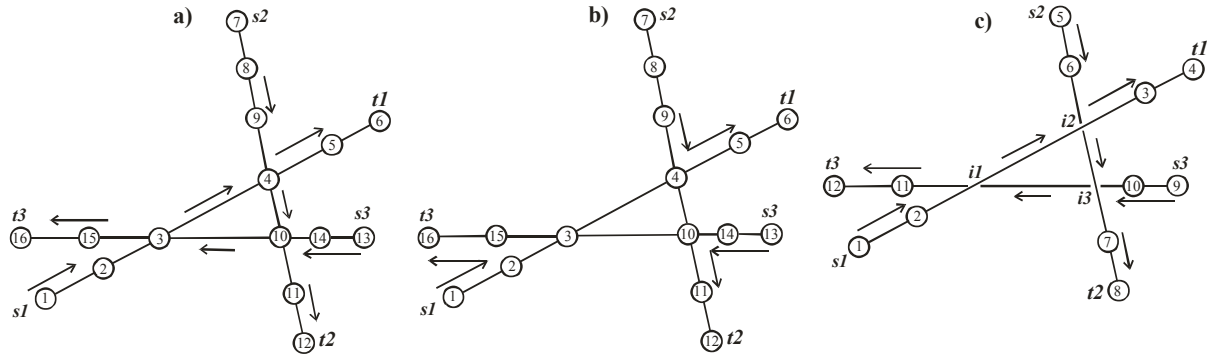

Figure 1. a) Parasitic flow loops may appear even if the transported commodity does not travel along a closed contour; b) Parasitic flow loops can be removed without affecting the throughput flow from sources to destinations; c) Parasitic flow loops essentially appear even if there are no nodes at the intersection points.

Finding the strongly connected components of a graph, which implies the existence of cyclic paths has been considered before (Sharir 1981). The question of identifying and removing parasitic flow loops in flow networks however, has been evading the attention of researchers for a very long time. This is evidenced by the fact that in spite of the years of intensive research on static flow networks, *the algorithms for maximising the throughput flow published since 1956 leave highly undesirable parasitic loops of flow in the "optimised" networks*. This shocking omission (discussed in Section 2) has been made despite the very large number of published algorithms for optimising the flows in networks. Research related to optimizing network flows for example, has been reviewed in (Ahuja et al. 1993; Asano and Asano 2000; Ford and Fulkerson 1962; Goldberg et al. 1990; Hu 1969; Tarjan 1983; Cormen et al. 2001; Kleinberg and Tardos 2006; Goodrich and Tamassia 2002; Papadimitriou and Steiglitz 1998 and Lawler 1976). Most of this research is related to determining the edge flows which maximise the throughput flow transmitted from a number of sources to a number of destinations (sinks).

There are two main categories of algorithms solving this problem. The augmentation algorithms preserve the feasibility of the network flow at all steps, until the maximum throughput flow is attained (Ford and Fulkerson, 1956; Dinic,1970; Edmonds and Karp, 1972; Elias et al.,1956). The central concept of the augmentation algorithms is the concept 'augmentable *s-t* path'. Augmentable *s-t* path is a sequence of forward and backward edges from the source *s* to the sink *t*, for which none of the forward edges is fully saturated with flow and none of the backward edges is empty. Such *s-t* path can be augmented with flow which consists of increasing the flow along all forward edges by a common amount Δ and decreasing the flow along all backward edges by Δ. The result is an increase of the throughput flow from the source *s* to the sink *t*, by the same amount Δ. For example, the Ford-Fulkerson algorithm (Ford and Fulkerson, 1956) works by augmenting available *s-t* paths until no more augmentable *s-t* paths can be found. According to the Ford-Fulkerson theorem (Ford and Fulkerson 1956), the maximum throughput flow is attained when no more augmentable *s-t* path can be found in the network. The Ford-Fulkeston algorithm may run very slowly on particular networks. The algorithm proposed by Edmonds and Karp (1972) avoids this difficulty by augmenting sequentially the shortest (in term of number of edges) augmentable paths and is superior to the Ford-Fulkerson algorithm.

The second major category of algorithms for optimising the throughput flow are based on the preflow concept proposed by Karzanov (1974) and subsequently used as a basis for the algorithms proposed by Sleator and Tarjan (1980) and Goldberg and Tarjan (1988). For the preflow, the sum of all edge flows going into a node is allowed to exceed the sum of all flows going out of the node. As a consequence, the flow conservation law at the nodes may be violated and the nodes may contain excess flow. The central idea behind the preflow-push algorithms is converting the preflow into a

feasible flow. The nodes are labeled with numbers indicating their 'height' with respect to the source node. Flow can only be sent from an excess node with a bigger height to a neighbouring node with a smaller height. If this cannot be done for a particular neighbouring node connected with an augmentable edge, a re-labeling operation is initiated. The height of the excess node is increased and subsequently, flow is pushed towards eligible neighbouring nodes. The process of pushing flow and relabeling continues until no internal node has excess flow. Attaining this state guarantees that the maximum throughput flow has been reached (Goldberg and Tarjan; Kleinberg and Tardos 2006).

2. THE CLASSICAL ALGORITHMS FOR MAXIMISING THE THROUGHPUT FLOW MAY LEAVE PARASITIC FLOW LOOPS IN THE "OPTIMISED" NETWORKS

Parasitic flow loops may be left after a network flow optimisation by classical augmentation algorithms. Consider the counter-example network in Fig.2, where, for the sake of simplicity, the capacities of the separate connecting edges are assumed to be equal to 100 units of flow per unit time. The classical Edmonds and Karp shortest-path algorithm proceeds with saturating the shortest (in terms of number of edges) path (1,5,9,14) with 100 units of flow, followed by saturating the next shortest path (1,2,4,9,12,13,14) with 100 units of flow and finally, with saturating the path (1,3,6,10,11,12,5,7,8,14) with 100 units of flow. As a result, a parasitic directed flow loop (5,9,12,5) appears, carrying 100 units of flow. The Edmonds and Karp algorithm leaves a directed flow loop in the network. The same critical flaw characterises the Ford-Fulkerson algorithm (1956), and other augmentation algorithms.

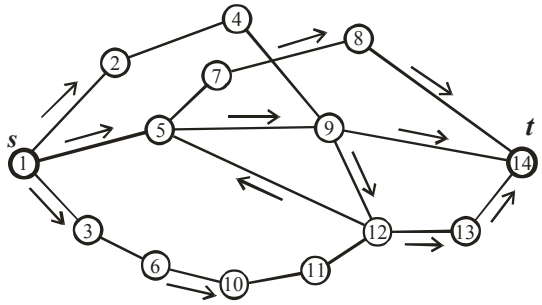

Figure 2. A counterexample network, demonstrating that the classical Edmonds and Karp (1972) shortest-path algorithm leaves a parasitic loop of flow (5,9,12,5) in the optimised network. All edges have a throughput capacity of 100 units of flow per unit time.

Parasitic flow loops may also appear after an optimisation by using preflow-push algorithms. Consider the network in Figure 3a. The network includes a single source *s* with capacity of 40 units and a destination node (sink) *t*. Suppose that the classical push-relabel algorithm (Cormen et al., 2001; Goldberg and Tarjan, 1988; Kleinberg and Tardos, 2006) is used to maximise the flow from the source *s* to the sink *t*. The first number on the labels of the edges denotes the capacity of the edge and the second number denotes the actual edge flow.

The preflow-push algorithm begins with assigning the highest label to the source *s* and zero to the rest of the nodes. The edge (1,2) coming out of the source *s* is fully saturated with flow (Fig.3b) and node 2 becomes an excess node. Edges (2,3) and (2,4) are augmentable; nodes 3 and 4 have the same label '0', and a relabeling operation is initiated for node 2. The label of node 2 is increased from 0 to 1, after which flow of 10 units is pushed towards node 3 and flow of 30 units towards node 4 (Fig.3c). Node 4 is the next selected excess node. Its label is '0', the same as the labels of the two eligible neighbours - nodes 3 and 7.

A relabeling operation is initiated, which increases the label of node 4 to 1. A flow of 20 units is then pushed towards node 7 and flow of 10 units is pushed towards node 3 (Figure 3d). Next, a relabeling operation is initiated for node 3 and its label is increased from 0 to 1, after which the excess flow of 20 units is sent towards node 5. Node 5 becomes an excess node with excess flow of 20 units. One of its eligible neighbours, node 4, has a label '1' and the other eligible neighbour, node 6, has a label '0'. A relabeling operation is initiated for node 5, whose label is increased to '1' and 10 units of flow are pushed towards node 6 (Fig.3e). Now, node 4 is an eligible neighbour towards which excess flow can be sent but it has the same label '1' as node 5. Following another relabeling operation, the label of node 5 becomes 2 and 10 units of excess flow are sent towards node 4 (Fig.3e). Now node 4 has 10 units excess flow which is pushed towards node 7. After pushing 30 units of excess flow from node 7 towards node 8 and 10 units of excess flow from node 6 towards node 8 there are no more excess nodes in the network. Figure 3f gives the final state of the edge flows. Despite

that the maximum throughput flow of 40 units has been reached, a parasitic flow loop (3,5,4,3) carrying 10 units flow appears in the network.

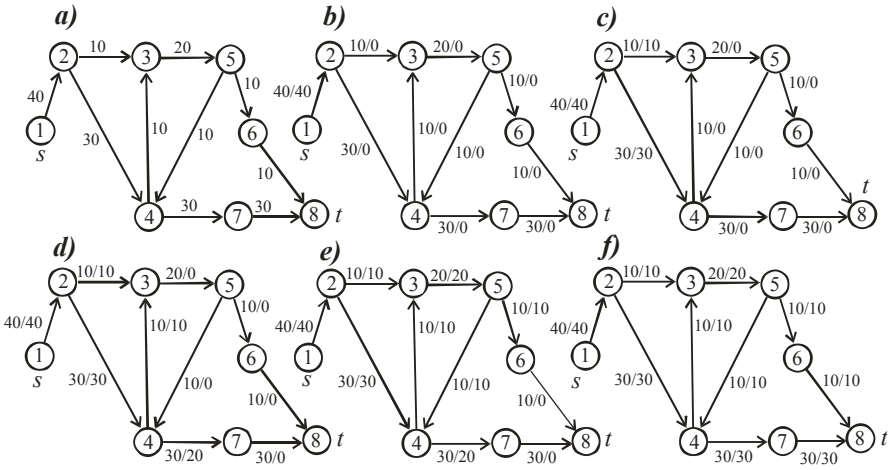

Figure 3. Maximising the throughput flow by using the preflow-push algorithm leads to a directed flow loop (3,5,4,3) carrying 10 units of flow.

3. ESTIMATING THE PROBABILITY OF PARASITIC FLOW LOOPS

The unexpectedly high probability of existence of parasitic flow loops will be demonstrated by considering two basic cases: (i) randomly oriented intersecting straight-line flow paths (Figure 4a) and (ii) the intersecting source-destination paths form a fixed grid with a random direction of the flow in each path (Fig.9).

3.1 Randomly oriented intersecting source-destination paths in a plane

The unexpectedly high probability of existence of parasitic flow loops will be demonstrated by considering the general case where the flow paths are randomly oriented intersecting straight lines (Figure 4a).

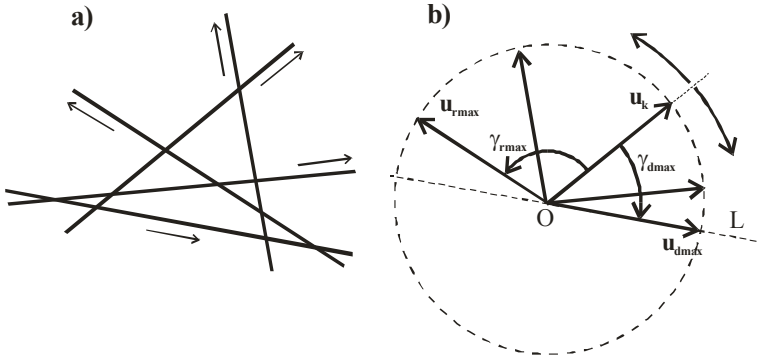

Figure 4. a) Randomly oriented intersecting source-destination paths; b) All direction vectors of the source-destination paths can be translated to start from a common point *O*.

All source-destination paths transport the same type of interchangeable commodity (e.g. petrol) and for each source-destination path, there is a particular direction of the flow (Fig.4a). For the sake of simplicity assume that all source-destination paths are characterised by the same throughput capacity and the flow along them fills the entire available capacity. It is also assumed that there are at least three source-destination paths; there are no parallel paths and no three paths intersect into a single point. These conditions are natural and common. Indeed, for randomly oriented straight-line flow paths on a plane, it is very unlikely to find two parallel paths or three paths intersecting into a single point. *The likelihood that a parasitic flow loop will be present in the network, given that the orientation of the intersecting flow paths is random, will be termed 'probability of a parasitic flow loop for random flow paths in a plane'.*

The existence of a parasitic flow loop anywhere between the points of intersection implies the existence of a triangular parasitic flow loop (Fig.5a). As a result, the existence of a triangular parasitic flow loop is a necessary condition for the existence of a parasitic flow loop. Conversely, the existence of a triangular parasitic flow loop is a sufficient condition for the existence of a parasitic flow loop. Consequently, the probability of a parasitic flow loop for randomly oriented source-destination flow paths can simply be estimated by estimating the probability of a parasitic triangular flow loop – the existence of three intersection points, between which the flow travels in the direction of traversing these points (Fig.5a).

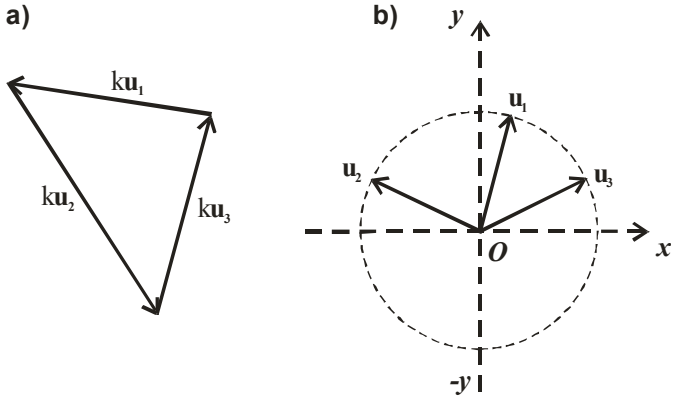

Figure. 5. a) A directed triangular flow loop; b) the direction vectors of the flow paths cannot possibly reside in a single half-plane.

A unit vector can be assigned to each flow path, whose direction is the same as the direction of the flow along the path (Fig.4b). The angle α the unit vector subtends with the horizontal axis (Fig.6a) gives the orientation of the source-destination path and the direction of the flow along the path. A random orientation of a flow path and the direction of its flow mean that the angle α the direction vector subtends with the fixed horizontal x-axis is uniformly distributed in the interval $(0,2\pi)$. In other words, all possible orientations are characterised by the same probability.

The unit vectors assigned to the source-destination paths will be referred to as 'direction vectors'. They can all be translated at the common origin *O*, as shown in Fig.4b. The following theorem then holds.

Theorem 1. *The necessary and sufficient condition for a parasitic flow loop in a network defined by randomly oriented straight-line flow paths is the non-existence of a half-plane where all direction vectors reside.*

Before proving this theorem a lemma will be stated and proved.

Lemma 1. *If no half-plane can be selected where all direction vectors reside, there is always a possibility to select three direction vectors which do not reside in a single half-plane.*

Proof. Let us select an arbitrary direction vector $\mathbf{u_k}$ and introduce counterclockwise and clockwise direction with respect to this vector to mark the angular positions of the rest of the direction vectors (Fig.6b). The angles γ_{di} mark the position of the unit vectors located in a clockwise direction from the selected vector $\mathbf{u_k}$ up to an angle equal to π (180°). The angles γ_{ri} mark the positions of the selected vectors located in a counter-clockwise direction from the unit vector $\mathbf{u_k}$ up to an angle equal to π. As a result, each of the angles γ_{di} and γ_{ri} is smaller than π ($\gamma_{di} < \pi ; \gamma_{ri} < \pi$) because no two direction vectors are parallel.

Let $\gamma_{d\max}$ be the angle corresponding to the most extreme direction vector $\mathbf{u_{dmax}}$ in a clockwise direction and $\gamma_{r\max}$ be the angle corresponding to the most extreme direction vector $\mathbf{u_{rmax}}$ in a counterclockwise direction. The three vectors $\mathbf{u_k}$, $\mathbf{u_{dmax}}$ and $\mathbf{u_{rmax}}$ do not reside in a single half-plane (Fig.6b).

Indeed, suppose that the three vectors $\mathbf{u_k}$, $\mathbf{u_{dmax}}$ and $\mathbf{u_{rmax}}$ reside in a single half plane. As a result, $\gamma_{d\max} + \gamma_{r\max} \le \pi$ and the three vectors $\mathbf{u_k}$, $\mathbf{u_{dmax}}$ and $\mathbf{u_{rmax}}$ should reside in the half-plane defined by the line *L* (oriented along the direction vector $\mathbf{u_{dmax}}$) and vector $\mathbf{u_k}$. Because $\gamma_{d\max}$ and $\gamma_{r\max}$ are the extreme angles corresponding to the direction vectors, the rest of the direction vectors reside either within the angle $\gamma_{d\max}$ or within the angle $\gamma_{r\max}$. As a result, all of the direction vectors must also reside in the half-plane defined by the line *L* and the direction vector $\mathbf{u_k}$. This is however impossible because, according to our assumption, no half-plane can be selected where

all direction vectors reside. This contradiction shows that the selected direction vectors $\mathbf{u_k}$, $\mathbf{u_{dmax}}$ and $\mathbf{u_{rmax}}$ do not reside in a single half plane.□

Now, Theorem 1 can be proved.

Proof of Theorem 1. First note, that the existence of a directed flow loop implies the existence of at least one triangular directed loop (Fig.5a) because no two source-destination paths are parallel. Suppose that the source-destination paths with direction unit vectors $\mathbf{u_1}$, $\mathbf{u_2}$ and $\mathbf{u_3}$, define a triangular directed flow loop (Fig.5a). The vectors along the sides of the triangular loop can then be presented by $k_1\mathbf{u_1}$, $k_2\mathbf{u_2}$ and $k_3\mathbf{u_3}$, where k_1, k_2 and k_3 are some constants. The sum of the vectors $k_1\mathbf{u_1}$, $k_2\mathbf{u_2}$ and $k_3\mathbf{u_3}$ is zero and as result, the relationship

$$k_1\mathbf{u_1}, + k_2\mathbf{u_2} + k_3\mathbf{u_3} \quad =0 \tag{1}$$

holds. A similar relationship holds regarding the projections of the vectors $k_1\mathbf{u_1}$, $k_2\mathbf{u_2}$ and $k_3\mathbf{u_3}$ on any specified axis. Thus, for any axis defined by a direction vector **s,** by multiplying equation (1) by the vector **s**, the relationship

$$m_{1s}, + m_{2s} + m_{3s} = 0 \tag{2}$$

is always fulfilled for the projections $m_{1s}=k_1\mathbf{u_1}\mathbf{s}$, $m_{2s}=k_2\mathbf{u_2}\mathbf{s}$ and $m_{3s}=k_3\mathbf{u_3}\mathbf{s}$.

Suppose that there is a single half plane where the direction vectors of flow paths reside (Fig.5b). In this case, the direction vectors $\mathbf{u_1}$, $\mathbf{u_2}$ and $\mathbf{u_3}$ of the paths forming the directed triangular loop will also reside in the same half-plane. Suppose that the axis x and the half-axis y define this half-plane (Fig.5b). The projections of the vectors along the y-axis will violate equation (2), because there will be a non-zero sum of projections along the half-axis (O,y) and zero sum of projections along the half-axis (O,-y). As a result $m_{1y}, + m_{2y} + m_{3y} > 0$; we arrived at a contradiction. Therefore, if a directed flow loop exists, there is no half-plane where all direction vectors reside.

Now, suppose that there is no half-plane where all of the direction vectors $\mathbf{u_1}$, $\mathbf{u_2}$,..., $\mathbf{u_n}$ reside. According to Lemma 1, in this case, it is always possible to select three vectors $\mathbf{u_1}$, $\mathbf{u_2}$ and $\mathbf{u_3}$, which do not reside in a half-plane. For these three vectors, conditions (1) and (2) will be fulfilled. Because by assumption, no three source-destination pairs intersect in a single point, the flow paths which correspond to the selected direction vectors $\mathbf{u_1}$, $\mathbf{u_2}$ and $\mathbf{u_3}$ will form a triangular parasitic flow loop.□

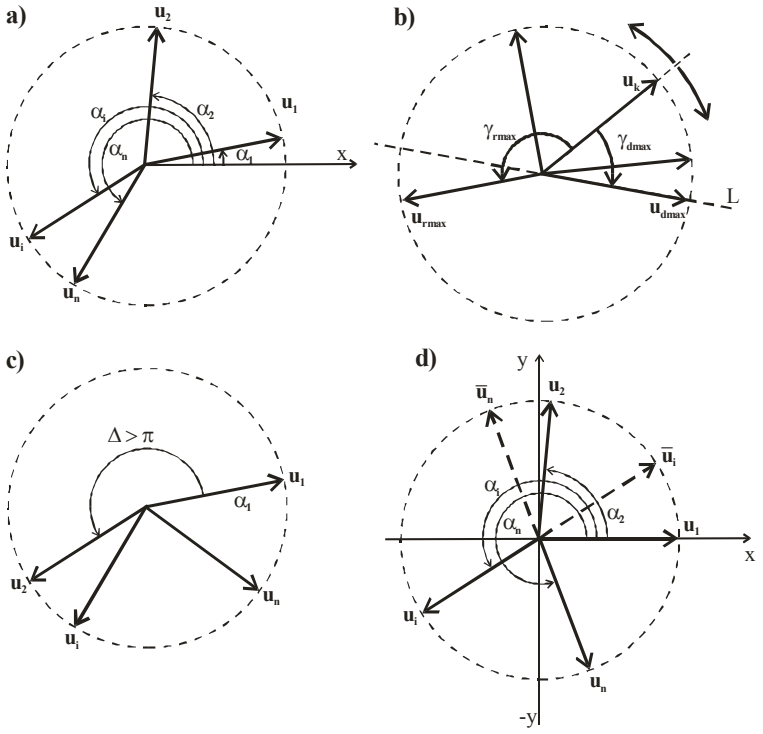

Figure 6. a) Ordering the direction vectors, according to the angle they subtend with the horizontal axis; b) If no half-plane can be selected where all direction vectors reside, there is always a possibility to select three direction vectors which do not reside in a single half-plane; c) a gap of size at least π between two random direction vectors; d) it is always possible to choose the direction of the flows in the intersection paths in such a way, that all direction vectors reside in the same half-plane.

Theorem 1 serves as a basis for calculating the probability of a parasitic flow loop. This probability can be determined by determining first the probability of the complementary event that no parasitic flow loop exists.

To calculate this probability, the random direction vectors are ordered in ascending order, according to the magnitude of the angle they subtend with the horizontal *x*-axis (Fig.6a).

The probability that there will be no parasitic flow loop is equal to the probability that all random direction vectors will lie in a single half-plane. All random direction vectors will lie in a single half-plane if and only if a gap of minimum length $\Delta = \pi$ exists between two random direction vectors (Fig.6c). There can be no more than a single gap of size $\Delta = \pi$, therefore, the random events corresponding to a gap between the first and the second direction vector, between the second and the third direction vector, etc., are mutually exclusive events. As a result, a single gap of minimum size $\Delta = \pi$ may be located in *n* distinct, mutually exclusive ways between the direction vectors.

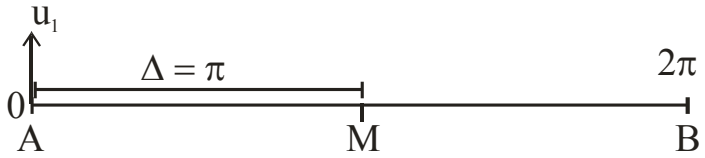

Figure 7. A gap of length Δ can be located in *n* distinct ways between the direction vectors.

Let the circumference of the direction vectors circle be represented by the segment with length 2π (Fig.7). A gap of length $\Delta = \pi$ between direction vectors u_1 and u_2, can only occur if the rest of the *n*-1 random locations fall in the segment MB and none of them falls in the segment AM (Fig.7). Because the probability that a random direction vector location will 'fall' on the segment MB is $\pi/(2\pi) = 1/2$, the probability that *n*-1 random vector locations will fall in the segment MB is $1/2^{n-1}$. Similarly, the probability of a gap between the second and the third direction vector is also $1/2^{n-1}$. As a result, the probability *p* of a gap of minimum size $\Delta = \pi$, between two direction vectors is a sum of the probabilities of these mutually exclusive events and becomes

$$p = \sum_{k=1}^{n} 1/2^{n-1} = n/2^{n-1} \tag{3}$$

which is also the probability that no parasitic flow loop will exist. Because there can either be a parasitic flow loop or no parasitic flow loop, the probability *q* of a parasitic flow loop becomes:

$$q = 1 - n/2^{n-1} \tag{4}$$

The probability of existence of a parasitic flow loop from equation (4) has been plotted in Fig.8.

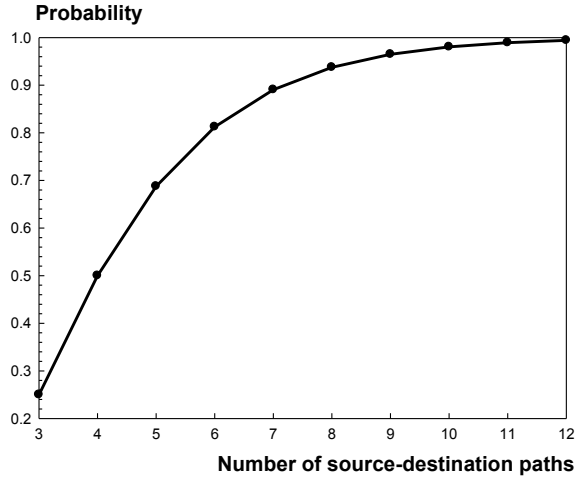

Figure 8. Probability of a parasitic flow loop as a function of the number of intersecting source-destination paths.

As can be verified, with increasing the number of intersecting random source-destination paths, the probability of a parasitic flow loop increases significantly. For five intersecting source-destination paths, the probability of a parasitic flow loop is already 69%. The results from equation (4) have been confirmed by Monte Carlo simulations.

Note from equation (4) that no matter how large the number *n* of intersecting source-destination paths is, the probability of parasitic flow loop is always smaller than 1. This important result suggests the next theorem:

Theorem 2. *For any possible number and for any possible orientation of straight-line flow paths on a plane, it is always possible to orient the flows along the paths in such a way, that no parasitic flow loops exist between the points of intersection.*

Proof. For a given orientation of *n* straight-line flow paths, consider a particular orientation of the flows along the paths. There are two mutually exclusive cases: (i) there are no parasitic flow loops between the points of intersection and (ii) there are parasitic flow loops. If case (i) is present, the existing flow directions represent the solution. Assume that case (ii) is present. Let us orient the *x*-axis along the direction vector $\mathbf{u_k}$ and arrange the rest of the direction vectors according to the angle α_i they subtend with x-axis (Fig.6d). Now, for all $\alpha_i > \pi$, the direction of the flow is changed in the opposite direction. As a result of this operation, all direction vectors will reside in a single half-plane determined by the *x*-axis and the positive *y*-semi-axis (Fig.6d). According to Theorem 1, there will be no parasitic flow loops for such orientation of the flows.□

3.2 A fixed grid of flow paths with a random direction of the flow in each path

Consider now a fixed lattice consisting of *n* vertical flow paths x_{ai}, x_{bi}, intersected with *n* horizontal flow paths y_{ai}, y_{bi} (Fig.9). All edges and flow paths have the same throughput capacity and the magnitude of the flow along each path is equal to the throughput capacity of the path. For each flow path, the direction of the flow in the path is random. Thus, for a horizontal flow path, the flow can travel with equal probability (0.5) rightwards or leftwards. Similarly, for a vertical flow path, the flow can travel with equal probability (0.5) upwards or downwards.

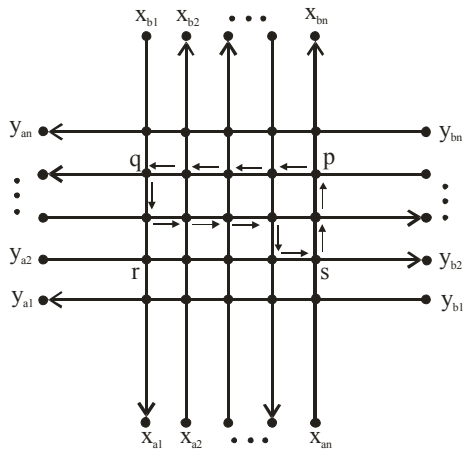

Figure 9. A lattice with intersecting source-destination paths with random directions of the flows creates the possibility of parasitic flow loops.

The likelihood that the flow will travel in the same direction (Fig.9) along a closed contour will be denoted 'probability of a parasitic flow loop in a lattice network'.

To reveal the probability of a parasitic flow loop in a lattice network, where the flows are with random directions, note that any parasitic flow loop in the lattice defined by the intersections always implies a parasitic flow loop around its tight rectangular envelop. Thus, the parasitic flow loop denoted by arrows in Fig.9, implies a parasitic flow loop along the tight rectangular envelope p,q,r,s,p. Conversely, the existence of a rectangular flow loop implies the existence of a flow loop. Consequently, the probability that a parasitic flow loop will exist, can be assessed simply by assessing the probability that a rectangular parasitic flow loop will exist.

To derive an analytical expression for the probability of a rectangular parasitic flow loop, the state of the flows along *k* vertical flow paths can be represented by a chain of *k* bits which will be referred to as V-chain. Each bit of the V-chain represents the direction of the flow in the corresponding flow path. Thus, a bit '0' stands for a downward direction of the

flow while a bit '1' stands for upward flow. Similarly, a chain of k bits can also represent the horizontal flow paths. This chain of bits will be referred to as 'H-chain'. A bit '0' in the H-chain codes a leftward orientation of the flow while a bit '1' codes a rightward orientation. For example, the vertical flows in Fig.9, starting from left to right are coded by the V-chain '01101' and the horizontal flows, starting from top to bottom are coded by the H-chain '00110'.

The number of all possible combinations of flow directions for k vertical paths is therefore 2^k. The number of all possible combinations of flow directions for k horizontal paths is also 2^k. All distinct combinations of flow directions in vertical and horizontal direction is therefore equal to $2^k \times 2^k = 2^{2k}$. A directed flow loop in clockwise direction requires the existence of at least a single bit combination 10 somewhere in the V-chain and the same bit combination 10 somewhere in the H-chain. This is because for a clockwise flow loop to be present there must be a flow in upward direction (bit '1'), a flow in downward direction (bit '0'), a flow in rightward direction (bit '1') and a flow in leftward direction (bit '0').

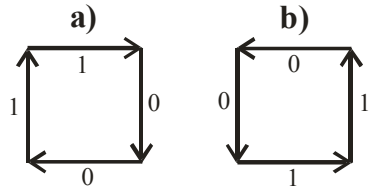

Figure 10. The two basic types of loops (clockwise and counter-clockwise loop) and the bits indicating the directions of the flows for the vertical and the horizontal flow paths.

The total number of V-chains which have at least a single bit string '10' somewhere, can be obtained by subtracting from the total number 2^k of possible V-chains, the number of V-chains which do not have a bit string '10'. For a 4-bit chain for example, the only bit chains which *do not* have a bit string '10' are: 0000, 0001, 0011, 0111 and 1111. This result is easily generalized and for a k-bit chain, the number of chains which do not have a bit string '10' is equal to k+1. As a result, the total number of V-chains which have the bit string '10' is $2^k - (k+1) = 2^k - k - 1$. In order to have a clockwise directed loop, at least one bit string '10' must also be present in the H-chain. Because H-chains and V-chains are combined independently, the total number of flow path configurations in which a clockwise directed flow loop exists is

$$N_1 = (2^k - k - 1)^2 \tag{5}$$

From Fig.10b it can be seen that for a counterclockwise directed loop to be present, at least one bit string '01' is required in the V-chain and at least one bit string '01' is required in the H-chain. With a reasoning very similar to the one used for deriving the total number of configurations with a clockwise directed loop, the total number of configurations in which a counter-clockwise flow loop exists is also

$$N_2 = (2^k - k - 1)^2 \tag{6}$$

Note that while estimating the numbers of clockwise and counter-clockwise flow loops, some of the configurations have been counted twice because they contain both clockwise and counter-clockwise loop. Both flow loops (clockwise and counterclockwise) are present if both bits strings '10' and '01' are present in the V-chain and both bit strings '10' and '01' are also present in the H-chain. The number of V-chains containing both bit strings '10' and '01' can be obtained by subtracting from the total possible number 2^k of distinct V-chains, the number of chains which do not contain both bit strings. For a 4-bit V-chain for example, the bit chains which do not contain both bit strings ('10' and '01') are: '0000', '0001', '0011', '0111', '1111', '1000', '1100', '1110'. This result is easily generalized for a k-bit chain. The number of chains which do not have both a string '10' and a string '01' is equal to $2k$. As a result, the total number of V-chains which have both bit strings ('10' and '01') is $2^k - 2k$. In order to have both: clockwise oriented directed loop and a counterclockwise oriented loop, both bit strings '10' and '01' must also be present in the H-chain. Because H-chains and V-chains are combined independently, the total number of flow path configurations in which both a clockwise and a counterclockwise directed flow loop exists is

$$N_{12} = (2^k - 2k)^2 \tag{7}$$

The total number flow path configurations with a loop in clockwise direction or with a loop in counterclockwise direction therefore are equal to $N_1 + N_2 - N_{12}$, where N_1 is given by equation (5), N_2 is given by equation (6) and

N_{12} by equation (7). The probability of a directed flow loop (in clockwise or counterclockwise direction) as a function of k vertical and k horizontal paths is therefore given by:

$$p = \frac{2 \times (2^k - k - 1)^2 - (2^k - 2k)^2}{2^{2k}} \tag{8}$$

The results from equation (8) have been confirmed by Monte Carlo simulations.

Equation (8) has been used for determining the probability of a parasitic flow loop for a different number of intersecting flow paths. The results have been presented in Fig.11. From the graph in Fig.11, it can be seen that the probability of a directed flow loop increases quickly with increasing the number of intersecting flow paths and even for 4 horizontal and 4 vertical intersecting paths (8 intersecting paths), the probability of a parasitic flow loop is nearly 70%. For 6 horizontal and 6 vertical intersecting pairs (12 intersecting paths), this probability is already nearly 93%. These results mean that in real lattice networks (transportation networks, manufacturing networks, electrical networks, computer networks), which include many intersecting flow paths, the existence of parasitic loops of flow is practically inevitable.

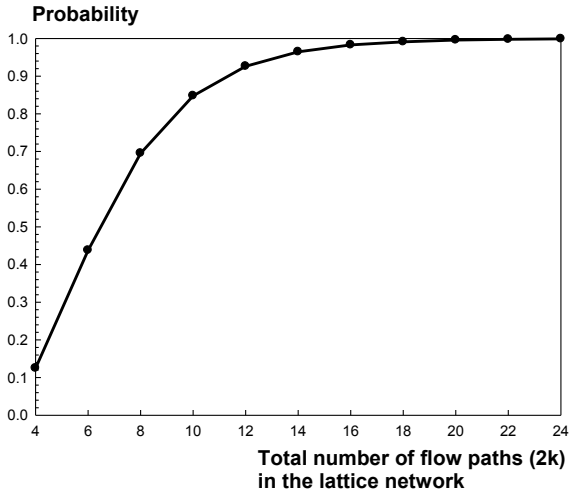

Figure 11. Results from equation 8, showing the variation of the probability of a parasitic flow loop on a fixed lattice, with random directions of the flows along k horizontal and k vertical paths.

4. DISCOVERING AND REMOVING PARASITIC FLOW LOOPS IN NETWORKS

The algorithm for discovering and removing a parasitic flow loop is based on calling a depth-first-search (dfs) procedure from an initial node and subsequently calling the dfs-procedure from the descendents of this node, etc., until an already traversed node is discovered again. Initially, all nodes are marked as 'not visited' (coloured' white). As the nodes are visited by the dfs-procedure, they are marked as "visited" (coloured gray, Fig.12). We must point out that the dfs-procedure does not scan all successors of the current node. It scans all eligible successors only. A successor node *i* of the current node 'r_node' is eligible if: (i) there is an edge directed from node r_node to node *i* and the edge (r_node,i) carries nonzero flow. If edge (r_node, *i*) is empty, the successor *i is not considered by the dfs-procedure*. Suppose that the call of the dfs- procedure from node 'r_node1' does not discover any already traversed node (Fig.12a). In this case, after the return from the dfs-call, the node r_node1 is marked as 'completed' (coloured black) (the entry of the r_node1 in the array cmpl[] is set to '1', cmpl[r_node1]=1). Under these conditions, the following theorem holds.

Theorem 3. *In a network with feasible flow, a parasitic flow loop is present only if during a recursive call of the depth-first search procedure, an already traversed node has been discovered again and it has not been marked as 'completed'.*

Proof. Suppose that the node '*e*', which has been visited again, has been marked as 'completed' (Fig.12a). Because the node has been marked as 'completed' (cmpl[*e*]=1), an earlier depth-first search must have started from this node and no directed flow loop must have been discovered starting with node '*e*'. Therefore, node '*e*' discovered twice and marked as 'completed', cannot possibly belong to a parasitic flow loop.

Now suppose that the dfs-procedure has been called and during the subsequent calls of the dfs-procedure from subsequent descendent nodes, an already traversed node 'r_node' not marked as 'completed' (cmpl[r_node]=0), has been

visited again (Fig.12b). Because node 'r_node' has been visited for a second time and it has not been marked as 'completed', no return from the earlier dfs-call initiated from this node has occurred (otherwise the node would have been marked as 'completed'). As a result, the 'r_node' has been visited again, after traversing a chain of descendent nodes starting from node 'r_node' and ending at r_node (Fig.12b). This essentially means that a parasitic flow loop has been discovered. □

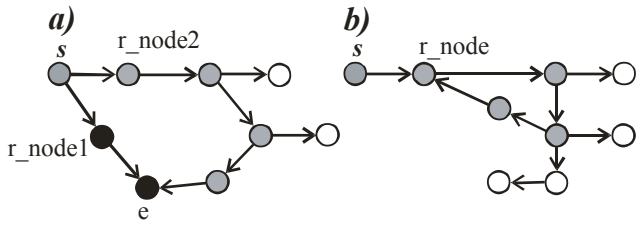

Figure 12. Traversing the nodes of the network by recursive calls to the depth-first-search procedure.

Here is the algorithm in pseudo-code:

Direct all edges of the network to match the directions of the edge flows.
//As a result, the network is transformed from a network with undirected edges to a network with directed edges.

```
procedure retrieve_directed_flow_loop(cur_node)
{// retrieves and eliminates the identified directed loop of flow}

procedure dfs(r_node)
{
 marked[r_node] = 1;
 for i= 1 to all eligible successors of r_node do
{
 cur_node = current eligible successor;
 if (marked[cur_node] = 0) then {
                          pred[cur_node] = r_node;
                          dfs(cur_node);
                                        }
else { if (cmpl[cur_node] = 0) then
          {
            pred[cur_node] = r_node;
            retrieve_directed_flow_loop(cur_node);
            break;
          }
       }
}
 cmpl[r_node] = 1;
}

Statements before the call of the dfs-procedure:
for i=1 to n do {marked[i] = 0; cmpl[i] = 0; pred[i] = 0;}
dfs(1).
```

A parasitic flow loop can only be discovered if both conditions are fulfilled: (i) a node cur_node, already marked as 'visited' has been encountered during the search and (ii) the call dfs(cur_node) is still active, in other words, its activation record is still in the stack. After the end of the dfs (cur_node) call, node r_node is marked as 'completed' by the statement 'cmpl[r_node] = 1'. This is why, only when both marked[cur_node] = 1 and cmpl[cur_node] = 0 are encountered during the search, a directed loop of nonzero flow has been discovered. The parasitic flow loop is subsequently retrieved and eliminated by the procedure ***retrieve_directed_flow_loop***(cur_node). The array pred[]

records the predecessors of the visited nodes and helps retrieve the identified directed flow loop. The procedure ***retrieve_directed_flow_loop***(cur_node) retrieves the discovered loop of flow by starting with the statement 'k = cur_node' followed by a loop, where the statement k = pred[k] is repeatedly executed and followed by a check whether an already encountered node has been encountered again and whether node *k* is a descendent of the 'cur_node'. These checks are used for identifying the node which closes the identified directed loop carrying nonzero flow. After discovering the parasitic flow loop, the procedure determines the edge from the loop carrying the smallest amount of flow and subtracts this flow from the flows of all edges belonging to the loop. As a result, at least one edge in the directed loop becomes empty. Once an edge becomes empty, it remains empty until the end of the procedure. This flow loop will not be discovered again during subsequent searches.

The proposed algorithm has been tested on a benchmark network which has the shape of a lattice (Fig.13). The lattice-type network has been selected because it is a natural network, often encountered in real applications. Because the lattice network is easily scalable, it provides an opportunity to isolate the impact of the size of the network on the algorithm's performance. To increase the number of parasitic flow loops, alternating directions of the flows from the sources s_i to the destinations d_i have been selected (Fig.13).

The same flow of 10 units per unit time has been assumed along each source-destination path.

Six different sizes of lattice-type network have been constructed and the network loops have been removed by the proposed algorithm, implemented in *C* and run on a computer with a processor *Intel(R) Core(TM) 2 Duo CPU T9900 @ 3.06 GHz*. According to the number of intersecting horizontal and vertical paths, the following network sizes were tested: 2x2, 3x3, 4x4, 5x5, 6x6 and 7x7. The number of nodes corresponding to the six test-networks were: 12, 22, 32, 45, 60 and 77, correspondingly.

The running time of the algorithm versus the size (the number of nodes) of the lattice network are shown in Fig.14.

It needs to be pointed out, that identifying all directed cycles in the network, before removing the bottleneck flow from any of them, is not a feasible approach. To show why this is the case, consider a complete network where any two nodes (*i*,*j*) are connected with directed edges. The number of directed cycles in this network is equal to the number all possible subsets of 2 nodes, 3 nodes,...,*n* nodes. Consequently, the number of directed cycles is $2^N - N - 1$ and determining all possible cycles is a task of exponential complexity - a task which is practically impossible even for moderately large *n*.

In the proposed approach, identifying a directed flow loop with the dfs-procedure and subtracting the bottleneck flow from the edges, has a worst-case complexity O(*m*), where *m* is the number of edges in the network. After each flow subtraction, at least a single edge remains empty. Therefore, after at most *m* steps, all parasitic loops of flow will be discovered and removed. As a result, the procedure for removing all parasitic loops of flow in a network has a worst-case running time $O(m^2)$.

3.1 Removing parasitic flow loops by solving a minimum-cost problem

The parasitic loops of flow can also be eliminated if the following procedure is used. Each edge is assigned a cost of transportation proportional to the length of the edge. Then, a problem of guaranteeing a throughput flow of a given magnitude, at a minimum cost is solved. In this case, the next theorem holds.

Theorem 4 *If a cost of transportation proportional to the length of the edges is assigned to each edge, guaranteeing the specified throughput flow at a minimum cost leaves no parasitic flow loops in the network.*

Proof. Suppose that the specified throughput flow has been guaranteed and the sum of the costs of transportation along all edges is the smallest possible. Suppose that there exists a directed loop of flow (k, k + 1,..., k + c, k) in the network. The smallest edge flow magnitude Δmin along this directed loop is then determined. This is the bottleneck flow characterising the cyclic path. The parasitic flow loop (k, k + 1,..., k + c, k) can then be drained by removing the bottleneck flow Δmin from each edge of the loop. The draining of the flow loop does not affect the throughput flow in the network. The throughput flow will remain unchanged after the draining.

Because, after draining of the bottleneck flow, the actual flows along the edges of the cyclic path are reduced (the bottleneck flow is prevented from being circulated), the overall costs of transportation along the edges will be reduced. However, this contradicts the assumption that the transportation cost associated with the initial edge flows was the smallest possible. Consequently, there can be no parasitic loops of flow in a network where the specified throughput flow has been achieved at a minimum transportation cost.□

The directed closed flow loops are not the only flow configuration associated with losses. *Dominated parasitic flow loops* are also associated with losses, congestion and unnecessary environmental pollution. A dominated parasitic loop can be

defined as a closed contour of edges with a defined direction of traversing, along which the sum of the lengths of the edges with forward flow dominates the sum of the lengths of the rest of the edges (which can be empty or with backward flow). In addition, all edges along the defined closed contour must be augmentable in a direction opposite to the direction of traversing the contour. Similar to the parasitic flow loops, a bottleneck capacity can be defined for dominated parasitic flow loops. A bottleneck residual capacity of a dominated parasitic flow loop is the maximum flow magnitude with which it can be augmented in a direction opposite to the direction of the flow in the dominating edges (in a direction opposite to the direction of traversing of the closed contour). Dominated flow loops can always be drained by augmenting them with flow in the opposite direction of the direction of traversing the closed contour. This operation results in an overall decrease of the transportation cost without affecting the quantity of the delivered commodity from the sources to destinations.

If a cost of transportation proportional to the length of the edges is assigned to each edge, it can be shown (the details are to be published elsewhere) that the network flow which minimises the transportation cost not only removes all closed parasitic flow loops but it also removes all dominated parasitic flow loops which are also associated with big transportation losses.

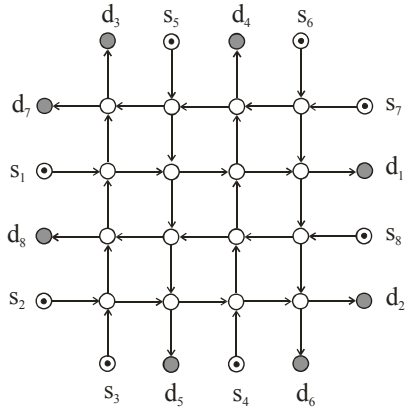

Figure 13. Lattice networks used for testing the proposed algorithm.

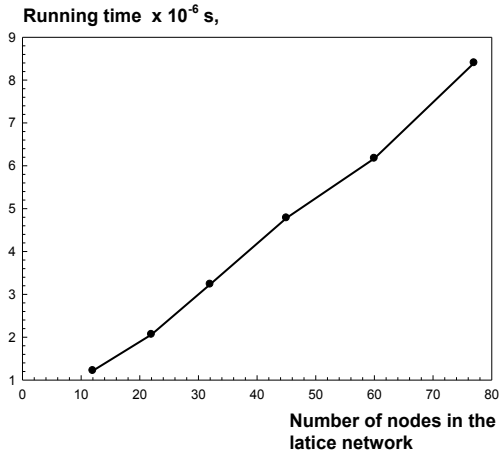

Figure 14. Performance of the proposed algorithm for a different size of the lattice network.

5. CONCLUSIONS

1. Parasitic flow loops occur naturally in real networks dispatching the same type of interchangeable commodity - fuel, electricity, gas, chemicals, particular goods, etc. by following directed straight-line paths from sources of flow to destinations. They are associated with transportation losses, congestion and increased pollution of the environment. Parasitic loops of flow may appear in networks even if the transported commodity physically does not travel along a closed contour.

2. The algorithms for maximising the throughput flow published since 1956 are fundamentally flawed: they all leave highly undesirable parasitic loops of flow in the optimised networks and are unsuitable for optimisation without an additional stage aimed at removing the parasitic flow loops.

3. For *n* intersecting, randomly oriented flow paths in a plane, fully saturated with flow, the probability of a directed flow loop is equal to $1-n/2^{n-1}$. Even for a relatively small number of intersecting randomly oriented flow paths, the probability of a directed flow loop is very large, which shows that the existence of directed flow loops in large and complex networks is practically inevitable.

4. For any possible number and for any possible orientation of straight-line flow paths on a plane, it is always possible to choose the directions of the flows along the paths in such a way, that no parasitic flow loops appear between the points of intersection.

5. A closed-form relationship has been derived to reveal the probability of a directed flow loop in a fixed lattice with flows with random directions. The analysis of the derived relationship shows that even for a relatively small number of intersecting flow paths, the probability of a directed flow loop is very large.

6. A theorem has been stated and proved related to discovering a directed flow loop in a network with feasible flows. On the basis of this theorem, a simple and efficient recursive algorithm has been proposed for discovering and removing parasitic flow loops in networks. The algorithm has a linear running time in the size of the network.

7. If a cost of transportation proportional to the length of the edges is assigned to each edge, guaranteeing the required throughput flow from sources to destinations at a minimum transportation cost leaves no parasitic flow loops in the network.

6. REFERENCES

1. Ahuja R.K., T.L.Magnanti, J.B.Orlin, (1993). Network flows: Theory, Algorithms and Applications, Prentice Hall.
2. Ahuja R.K. and J.B.Orlin, (1991). Distance-directed augmenting path algorithms for maximum flow and parametric maximum flow problems, Naval Research Logistics*,* 38: 413-430.
3. Asano T., Y.Asano, (2000). Journal of the Operations Research Society of Japan, 43(1): 2-31.
4. Cormen T.H., T.C.E.Leiserson, R.L.Rivest, and C.Stein, (2001). Introduction to Algorithms, 2nd ed., MIT Press and McGraw-Hill.
5. Dinic E.A., (1970). Algorithm for solution of a problem of maximum flow in a network with power estimation, Soviet Math. Doklady, 11(8): 1277-1280.
6. Edmonds J. and R.M.Karp, (1972). Theoretical improvements in algorithmic efficiency for network flow problems, Journal of the ACM, 19(2): 248-264.
7. Elias P., A.Feinstein and C.E.Shannon, (1956). Note on maximum flow through a network, IRE Transactions on Information Theory, IT2: 117-119.
8. Ford L.R. and D.R. Fulkerson, Maximal flow through a network, (1956). Canadian Journal of Mathematics, 8(5): 399-404.
9. Ford L.R. and D.R.Fulkerson, (1962). Flows in Networks, Princeton University press, Princeton, N.J.
10. Goldberg A.V. and R. E. Tarjan, (1988). A new approach to the maximum flow problem, Journal of ACM*,* 35: 921-940.
11. Goldberg A.V., E.Tardos and R.E.Tarjan, (1990). Network flow algorithms, in Algorithms and Combinatorics, v.9. Paths, Rows and VLSI-Layout, Sprmger-Verlag Berlin, Heidelberg.
12. Goodrich M.T. and R.Tamassia, (2002). Algorithm design, John Wiley & Sons.
13. Hengartner, U., S. Moon, R. Mortier, and C. Diot. 2002. Detection and analysis of routing loops in packet traces. Sprint ATL Technical Report TR02-ATL-051001.
14. Hu T.C., Integer programming and network flows, (1969). Addison-Wesley Publ.Company, Reading, Mass.
15. Karzanov A., (1974). Determining the maximal flow in a network by the method of preflows., Sov.Math.Dokl.
16. Kleinberg J. and E.Tardos, (2006). Algorithm design, Addison Wesley.
17. Kordalewski D. and R.Robere, A Dynamic Algorithm for Loop Detection in Software Defined Networks, University of Thoronto, 2012.
18. Lawler E., Combinatorial Optimisation, (1976). Dover publications, Inc, New York.
19. Papadimitriou C.H., K.Steiglitz, (1998). Combinatorial Optimisation: Algorithms and complexity, Dover publications, Inc., New York.

20. Paxson V. (1997). End-to-end routing behavior in the Internet. IEEE/ACM Transactions on Networking, 5(5): 610–615.
21. Sharir M., (1981). A strong-connectivit;y algorithm and its applications in data flow analysis, Computers and Mathematics with Applications, 7: 67-72.
22. Sleator D.D. and R.E.Tarjan (1980). An O(*nm* log *n*) algorithm for maximum network flow. Technical report STAN-CS-80-831, Department of Computer Science, Stanford University, Stanford, CA.
23. Tarjan R.E., (1983). Data structures and network algorithms, SIAM, Philadelphia.