The dual network theorem for static flow networks and its application for maximising the throughput flow

Michael Todorov Todinov

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

Correspondence: Michael Todorov Todinov. Address: Department of Mechanical Engineering and Mathematical Sciences, Oxford Brookes University, Oxford, Wheatley, OX33 1HX, UK. Telephone: 44-186-548-3546. Email: mtodinov@brookes.ac.uk.

Received: July 10, 2012 **Accepted:** August 5, 2012 **Online Published:**
DOI: **URL:**

Abstract

The paper discuses a new fundamental result in the theory of flow networks referred to as the 'dual network theorem for static flow networks'. The theorem states that the maximum throughput flow in any static network is equal to the sum of the capacities of the edges coming out of the source, minus the total excess flow at all excess nodes, plus the maximum throughput flow in the dual network. For very few imbalanced nodes in a flow network, determining the throughput flow in the dual network is a task significantly easier than determining the throughput flow in the original network. This creates the basis of a very efficient algorithm for maximising the throughput flow in a network, by maximising the throughput flow in its dual network.

Consequently, a new algorithm for maximising the throughput flow in a network has been proposed. For networks with very few imbalanced nodes, in the case where only the maximum throughput flow is of interest, the proposed algorithm will outperform any classical method for determining the maximum throughput flow.

In this paper we also raise awareness of a fundamental flaw in classical algorithms for maximising the throughput flow in static networks with directed edges. Despite the years of intensive research on static flow networks, *the classical algorithms leave undesirable directed loops of flow in the optimised networks*. These directed flow loops are associated with wastage of energy and resources and increased levels of congestion in the optimised networks. Consequently, an algorithm is also proposed for discovering and removing directed loops of flow in networks.

Key words

Maximising throughput flow, Static flow networks, Draining algorithms, Maximum flow after edge failures

1 Analysis of draining algorithms for maximising the throughput flow in a static network

A large number of algorithms have been proposed for solving the maximum flow problem by using a graph representation of the flow network. Most of this research has been focused mainly on static flow networks, where no failure of components is considered. Research related to static flow networks has already been comprehensively reviewed[1,2,3,11,16]. This research has been focused on calculating the maximum flow transmitted from sources to sinks, for static flow networks. Various distinct approaches for solving this problem have been proposed.

The first big category of methods for maximising the throughput flow in networks includes methods which start from an empty network and gradually saturate the edges with flow, until the whole network is fully saturated with flow[4,5,7,8,9,10,13,15]. Within this big category, two major sub-categories of flow maximisation algorithms can be distinguished. The first sub-category includes path augmentation algorithms which, at all steps, preserve the feasibility of the network flow until the maximum flow is attained[4,7,8]. Thus, the Ford-Fulkerson algorithm[8] is based on finding available flow paths and augmenting them with flow until no more augmentable flow paths can be found. An improvement compared to the Ford-Fulkerson algorithm is the Edmonds and Karp algorithm[7], based on finding the shortest augmentable source-to-sink path.

The second sub-category of methods for maximising the flow in a network with empty edges, includes the preflow-push algorithms, which preserve the maximum flow, until a feasible flow is attained[9,13,15]. The preflow-push algorithms maintain a preflow, for which the capacity constraints on the edges is honoured but the flow conservation law at the nodes may be violated in the followings sense: each node other than the source may contain excess flow. In other words, the sum of the flows that go into the node may be greater than the sum of the flows leaving the node. The algorithm works to convert the preflow into a feasible flow. This is done by including labels on the nodes indicating their 'height' with respect to the source node. Flow is pushed from a node with a higher height to a node with a lower height. If this cannot be done for a particular node, a relabeling operation is included, which increases the height of the node. Continuing this process, until no internal node (other than the source and the sink) has an excess flow, guarantees that the maximum throughput flow has been reached[14].

Most of the algorithms for maximizing the flow in static flow networks have polynomial time complexity. The running time of the algorithms for maximizing the flow in static networks however, can often be improved significantly, if the particular network topology is exploited directly. For planar networks for example, very efficient throughput flow maximisation algorithms have been designed[12]. For networks with tree topology, a very efficient algorithm for maximising the throughput flow can be designed, with linear worst-case running time, in the number of edges in the network (Todinov[21]). These algorithms take advantage of the planar topology and tree topology and their running time is superior to the running time of algorithms handling networks with arbitrary topology.

Recently, a principally different approach for maximising the flow in static flow networks was proposed by Dong et al.[6], based on the concept of starting from a fully saturated with flow network. The draining algorithm proposed in Dong et al.[6] starts from a network with fully saturated edges, which also includes a backward edge connecting the sink with the source. The algorithm drains flow from paths connecting deficit nodes and excess nodes, until all nodes become balanced and a maximum throughput flow is attained. In effect, the draining algorithm conducts balancing of excess and deficit nodes until the excess and deficit flow at the nodes disappears. The node balancing is conducted along the shortest augmentable paths between excess and deficit nodes.

The definition of excess and deficit nodes given by Dong et al.[6] is as follows. For a network with edges fully saturated with flow, if the sum of capacities of all edges going into a node *e* (different from the source and the sink) is greater than the sum of capacities of all outgoing edges, the node is referred to as excess node. The amount of excess flow ef at an excess node *e* is given by:

$$ef = \sum_{i \in \delta+} c(i,e) - \sum_{j \in \delta-} c(e,j) > 0 \tag{1}$$

Conversely, if the sum of capacities of all edges going into a node *d* (different from the source and the sink) is smaller than the sum of capacities of all outgoing edges, the node is referred to as deficit node. The amount of deficit flow *df* in the deficit node is:

$$df = \sum_{i \in \delta+} c(i,d) - \sum_{j \in \delta-} c(d,j) < 0 \tag{2}$$

Finally, if the sum of capacities of all edges going into a particular node is equal to the sum of capacities of all edges going out of the node, the node is referred to as balanced node. The amount of excess/deficit flow at a balanced node is zero.

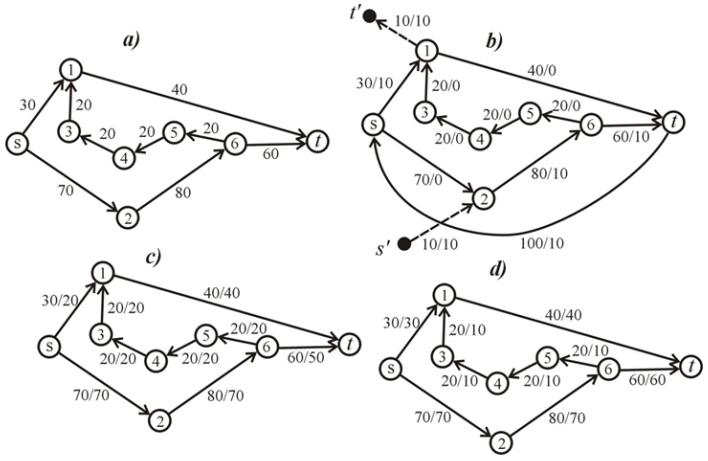


Figure 1. A simple counterexample network, for which the draining algorithm in Dong et al.[6] produces an incorrect result.

Unfortunately, the algorithm proposed in Dong et al.[6] is fundamentally flawed and, as a result, it yields suboptimal solutions. This can be demonstrated by the simple counterexample network from Fig.1a, which contains one deficit node (node 2) and one excess node (node 1). The labels on the edges of the network in Fig.1a are their flow capacities. According to (Dong et al.[6], the flooding network for the network in Fig.1a is the one in Fig.1b. Following the draining algorithm, after augmenting the flooding network in Fig.1b with 10 units flow, along the shortest path (s',2,6,t,s,1,t'), and subtracting the network flows from the flow capacities of the edges in Fig.1a, an incorrect value for the maximum throughput flow (equal to 90 flow units) is obtained (Fig.1c). However, the maximum throughput flow in the network from Fig.1a is 100 units, and the correct edge flows yielding this throughput flow are shown in Fig.1d.

This simple counterexample shows that there are networks for which the draining algorithm proposed in Dong et al.[6], fails to determine the correct maximum throughput flow. In what follows, we show that a correct algorithm for obtaining the maximum throughput flow in a network whose edges are initially saturated with flow must initially include node balancing in the network which does not include the circulation edge (*t,s*). The purpose is to make the maximum possible redistribution of excess and deficit flow between the excess and deficit nodes, before draining the excess flow from the network. However, the draining algorithm proposed in Dong et al.[6], always does node balancing on a network with a circulation edge (*t*,*s*). This is the reason why for some networks it yields sub-optimal solutions.

Very recently, similar work related to maximising the throughput flow in static flow networks appeared in Ursani[22], which exploits the concept proposed by Dong et al.[6] – maximising the throughput flow in a static flow network by starting from fully saturated with flow edges and performing node balancing. Despite the terms 'dissipative flow' and 'dissipative paths' used in Ursani[22], the described method is a modification of the draining method proposed earlier by Dong et al. (2009).

In Ursani[22], an attempt has been made to provide a proof that node-balancing will always yield the maximum throughput flow in a static network. The presented proof hinged on proving that the minimal cut remains saturated during all operations involved in node balancing. Unfortunately, the proof given in Ursani[22] contains several fundamental errors which render it invalid. The proof was based on an equation stating that the maximum throughput flow in a network (the capacity of the minimal cut) is equal to the sum of the capacities of the edges going out of the source minus the sums of the

excess and deficit flows at the internal nodes belonging to the s-part of the minimal cut (part A in Fig.2, where the source *s* belongs). The simple counterexample network in Fig.2 shows that this statement is incorrect.

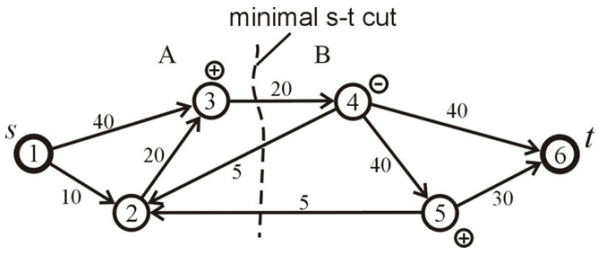


Figure 2. A counterexample network showing that the maximum throughput flow (the capacity of the minimal cut) is not necessarily equal to the sum of the capacities of the edges going out of the source minus the sums of the excess and deficit flow at the excess and deficit nodes belonging to the *s*-part of the minimal cut (part A).

The only imbalanced node in the *s*-part of the s-t cut is node 3, with 40 units excess flow. According to the classical Ford-Fulkerson result[8], the maximum throughput flow is equal to the capacity $C_0 = 20$ units of the minimal cut, which consists of the forward edge (3,4) only.

The sum of the capacities of the edges coming out of the source *s* is 50 units and the excess flow at the excess node 3 is 40+20-20=40 units. Contrary to the statement made in Ursani[22], $C_0 = 20 \neq 50 - 40$.

The second, fundamental error in the proof presented in Ursani[22] is that the minimal cut has not essentially been used in the proof. Ursani[22] stated that for the minimal *s-t* cut, the sum of capacities of the edges going out of the source $\sum_p c(s,p)$ minus the sum of the excess (ef_i) and deficit (df_i) flows at the internal nodes belonging to the *s*-part (part A) of the minimal cut $\sum_{i\in A} ef_i + \sum_{j\in A} df_j$ is always equal to the sum of capacities of the edges going into the sink $\sum_k c(k,t)$ plus the sums of the excess and deficit flows $\sum_{i\in B} ef_i + \sum_{j\in B} df_j$ in the *t*-part (part B) of the minimal cut.

$$\sum_p c(s,p) - \left(\sum_{i\in A} ef_i + \sum_{j\in A} df_i \right) = \sum_k c(k,t) + \left(\sum_{i\in B} ef_i + \sum_{j\in B} df_i \right) \tag{3}$$

Equation (3) is correct, but it is valid not only for the minimal cut; equation (3) is valid for *any s-t* cut in the network! An *s-t* cut, is a partition ($A, \overline{A}$) of the set *V* of all nodes in the flow network into two sets A and $\overline{A}$ ($A \cap \overline{A} = \varnothing$; $A \cup \overline{A} = V$), in such a way, that the source *s* belongs to the first set and the sink *t* belongs to the second set ($s \in A;\ t \in \overline{A}$)[1].

Indeed, for any fully saturated flow network the flow conservation law necessarily holds:

$$\sum_p c(s,p) - \sum_k c(k,t) = \sum_{i\in V} ef_i - \sum_{i\in V} |\, df_i \,| \tag{4}$$

Equation (4) simply states that the difference between the flow coming out of the source *s* and the flow going into the sink *t* is equal to the difference of the total excess flow at the excess nodes and the total deficit flow at the deficit nodes. Note that the deficit flows df_i at the deficit nodes are with negative sign; this is why they have been taken with their absolute values. Consider now an arbitrary *s-t* cut ($A, \overline{A}$). Equation (4) can be re-written as

$$\sum_{p} c(s,p) - \sum_{k} c(k,t) = \sum_{i \in A} ef_i + \sum_{j \in \bar{A}} ef_i - \left(\sum_{i \in A} |\, df_i \,| + \sum_{j \in \bar{A}} |\, df_j \,| \right) \tag{5}$$

or

$$\sum_{p} c(s,p) - \left(\sum_{i \in A} ef_i - \sum_{j \in A} |\, df_j \,| \right) = \sum_{k} c(k,t) + \sum_{j \in \bar{A}} ef_i - \sum_{j \in \bar{A}} |\, df_j \,| \tag{6}$$

When the absolute values in equation (6) are removed, equation (3) is obtained. In words, equation (3) is valid not only for the minimal cut but also for any *s*-*t* cut.

Therefore, in Ursani[22], an attempt was essentially made *to prove the impossible*: that any *s*-*t* cut remains fully saturated after the node balancing operations. However, when the maximum throughput flow is set in the network, only the forward edges of the minimal *s*-*t* cut are fully saturated[8]. The forward edges of other *s*-*t* cuts may not necessarily be fully saturated. Therefore, the "proof" presented in Ursani[22] is trying to establish that at the end of the node balancing operations the forward edges of any *s*-*t* cut will be fully saturated which is incorrect. Hence, the proof presented in Ursani[22] is invalid.

Here, we need to point out that the node-balancing theorem as a way of maximising the throughput flow in repairable flow networks, has already been stated explicitly in Todinov[18,19] where the redistribution of flow along paths connecting excess and deficit nodes has also been used. The node-balancing theorem has also been rigorously proved in Todinov[17,21] and this proof is valid for both, repairable and static flow networks.

Furthermore, in Ursani[22], a worst-case running time $O(m^2 - m)$ has been claimed, where *m* is the number of edges in the network). Unfortunately, the complexity analysis presented in Ursani[22] is incorrect. The reason for this error was routed in the circumstance that the worst-case analysis in Ursani[22] has been based on a special-case simple network, with a linear topology. A general flow network however, rarely has a linear topology. In fact, the worst-case running time of the modified Dong et al[6]. algorithm, presented in Ursani[22], is no better the worst-case running time $O(m \,|\, f^* \,|)$ of the classical Ford-Fulkerson augmentation algorithm[8], where $|\, f^* \,|$ is the magnitude of the maximum throughput flow. To see this, take a single-source single-sink flow network with directed edges, complex topology, and no deficit and excess internal nodes (Fig.3a). In other words, for any internal node, the sum of capacities of the ingoing edges is equal to the sum of capacities of the outgoing edges. Assume that the maximum throughput flow, from the source *s* to the sink *t*, in the empty network in Fig.3a, is the big value $C_0 >> 1$. Let the sum $\sum_i c(s,i)$ of capacities $c(s,i)$ of the edges coming out of the source *s* be equal to the maximum throughput flow C_0 and the sum $\sum_k c(k,t)$ of capacities $c(k,t)$ of edges going into the sink *t*, be also equal to C_0, $\sum_i c(s,i) = C_0 = \sum_k c(k,t)$.

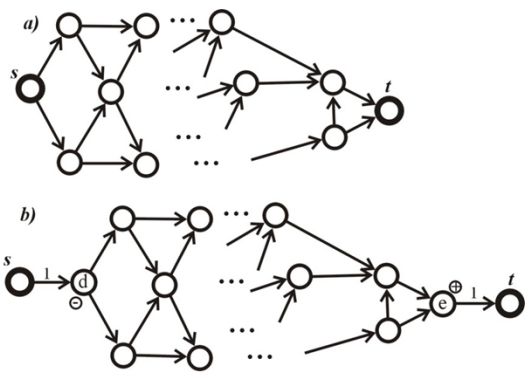


Figure 3. a) Initial network; b) Initial network, transformed into a worst-case network.

Now, let us transform the network in Fig.3a by detaching the source *s* and the sink *t* and connecting them to the rest of the network through edges with unit capacity (Fig.3b). After this operation, in the resultant network (Fig.3b), exactly two imbalanced internal nodes will appear: the deficit node 'd' with deficit flow $-(C_0 - 1)$ and the excess node 'e', with excess flow $C_0 - 1$. Now, according to the modified algorithm described in Ursani[22], all edges are initially fully saturated with flow and the deficit and excess flow at nodes 'd' and 'e' is decreased by draining flow from feasible paths, starting at the deficit node d and ending at the excess node e. However, the process of draining flow from feasible paths connecting the deficit and excess node and draining the edges of the network in Fig.3b is *the reverse* of the process of maximising the throughput flow in the empty network in Fig.3a, by the classical Ford-Fulkerson augmentation algorithm[8].

The worst case running time of the Ford-Fulkerson algorithm is $O(mC_0)$ where C_0 is the maximum throughput flow in the network and *m* is the number of edges in the network[1]. Consequently, the worst-case running time of eliminating the excess and deficit flow at nodes 'e' and 'd' in the network in Fig.3b cannot possibly be better than $O(mC_0)$ This worst-case running time is significantly worse than the claimed worst-case running time of $O(m^2 - m)$ by Ursani[22]. With increasing the maximum throughput flow C_0, the worst-case running time $O(mC)$ increases significantly.

Even if the shortest-path Edmonds and Karp algorithm were used for draining paths between excess and deficit nodes, the worst case running time of the modified algorithm described in Ursani[22] would not be greater than $O(m^2 n)$ where *n* is the number of nodes in the network. This is because $O(m^2 n)$ is the worst-case running time of the Edmonds & Karp algorithm[7]. Again, this worst-case running time is significantly worse than the claimed worst-case running time $O(m^2 - m)$ by Ursani[22].

Another drawback of the algorithms described in Ursani[22] and Dong et al[6] for maximising the throughput flow in a static network is that both algorithms have an increased tendency of leaving directed loops with parasitic flow in the optimised networks. These parasitic loops of flow are *highly undesirable* because: (i) they increase transportation costs by circulating commodity along closed loops unnecessarily (ii) they consume unnecessarily residual capacity from the edges of the network and (iii) energy is unnecessarily wasted for maintaining the directed loops of flow. For electric networks, directed loops of flow mean significant electrical losses and unnecessary consumption of power line capacity which leads to congestion and overloading of the power lines and diminishing the reliability of the grid. In computer networks, parasitic loops of flow consume bandwidth from the communication lines, increase unnecessarily data traffic. These lead ultimately to congestion and delayed data transmission which affects negatively the quality of service of the network. For supply networks, the existence of parasitic loops of flow means that the transportation costs are unnecessarily increased and energy is wasted on circulating commodity unnecessarily.

This drawback of the methods proposed in Dong et al.[6] and Ursani[22] can be illustrated with the network in Fig.4a, where the labels stand for edge capacities.

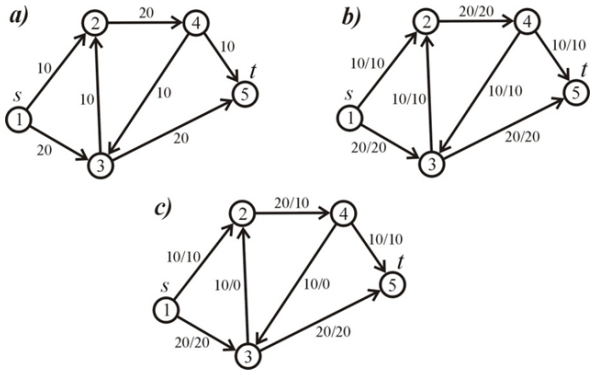


Figure 4. a) Initial network; b) A network with a parasitic loop (3,2,4,3) of 10 units flow, after the throughput flow maximisation by the method proposed in Ursani[22] and Dong et al.[6]; c) The same network, optimised by the proposed in this paper method has the same maximum throughput flow of 30 units, but no parasitic loops of flow are present.

According to the method proposed in Dong et al.[6] and its modification proposed in Ursani[22], in the network from Fig.4a, there are no excess and deficit nodes. As a result, no flow is removed from any edge in the fully saturated with flow network and the resultant edge flows maximising the throughput flow are according to Fig.4b. The first number 'c' of the labels 'c/f' on the edges stands for the edge capacity and the second number 'f', stands for the actual flow along the edge. The maximum throughput flow of 30 units (Fig.4b) has indeed been obtained correctly, but a directed cyclic path (3,2,4,3) with parasitic flow of 10 units still remains in the network!

The optimal solution has been shown in Fig.4c. where the maximum throughput flow is still 30 units but no parasitic loops of flow exist. Because the algorithms discussed in Ursani[22] and Dong et al.[6], have an increased tendency of leaving parasitic loops of flow, which are highly undesirable in any real flow network, these algorithms, in their present form, *should not be used* for optimising edge flows in static flow networks.

If the classical Edmonds and Karp shortest-path algorithm[7] was used for maximising the throughput flow, by starting from a network with empty edges (Fig.4a), the shortest path (1,3,5) will be augmented first, with 20 units of flow. This path augmentation will be followed by augmenting the path (1,2,4,5) with 10 units of flow. The result is the network in Fig.4c. with a maximum throughput flow of 30 units. No parasitic loops of flow exist in the optimised network from Fig.4c.

The counter-example network in Fig.5 however, reveals the same fundamental flaw also in well-established classical algorithms for maximising the throughput flow in networks.

For the counter-example network in Fig.5, the Edmonds and Karp shortest-path algorithm[7] proceeds with saturating the shortest-path (1,2,3,13) with 10 units of flow, followed by saturating the next shortest path (1,5,6,3,4,13) with 10 units of flow and finally, saturating the path (1,8,9,10,11,4,2,7,12,13) with 10 units of flow. As a result, a directed flow loop (2,3,4,2) appears, carrying 10 units of flow!

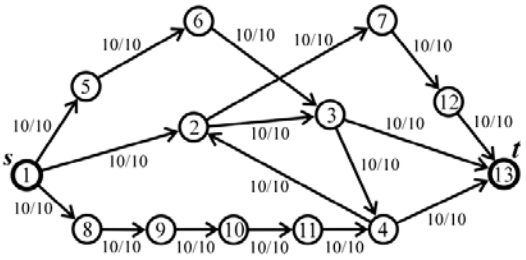


Figure 5. A counterexample network, demonstrating that the classical Edmonds and Karp[7] shortest-path algorithm leaves a directed loop of flow (2,3,4,2) in the optimised network with directed edges. All edges have a flow capacity of 10 units.

In Todinov[21] it has been shown that that the push-preflow algorithms[9] for maximising the throughput flow in static networks also leave directed flow loops in th optimised networks. Consequently, with this paper, we raise awareness of the fact that classical algorithms for maximising the throughput flow in static networks with directed edges, leave highly undesirable directed loops of flow in the optimised networks and should not be used for optimising the edge flows without an extra stage aimed at identifying and removing directed loops of flow.

2 The dual network theorem for static flow networks

The analysis in Section 1 poses the important question: 'what is the correct algorithm which exploits the concept proposed by Dong et al.[6] for maximising the throughput flow by starting from a network with edges fully saturated with flow.

Here, by using 'the maximum flow after edge failures theorem' suggested in Todinov[17,18,19], a correct algorithm can be developed for determining the maximum throughput flow in a static network, whose edges are initially fully saturated with flow. The maximum flow after edge failures theorem applies to a network where the maximum throughput flow $Q_{\max}$ has been established (for example by any of the classical algorithms for maximising the flow in a network) and in which several edges (i=1,2,...,*K*) fail simultaneously. The failed edge *i* is directed from node e_i to node d_i. (Fig.6a). The flow through the edges before their failures are $f(e_i,d_i)$, *i*=1,2,...,*K*.

If the failed edge (e_i, d_i) is not empty, then after the edge failure, a momentary excess flow will exist at node e_i, equal to the flow $f(e_i,d_i)$ through the edge (e_i, d_i), (Fig.6b). In other words, the sum of the flows going into node e_i will be greater than the sum of the flows leaving the node. This difference has been referred to as 'excess flow' ef_i at node e_i

$$ef_i = \sum_{k\in\delta+} f(k,e_i) - \sum_{k\in\delta-} f(e_i,k) > 0 \tag{7}$$

and the node e_i has been referred to as 'excess node'.

Alternatively, at the other end of the failed edge (e_i, d_i), at node d_i, deficit flow will be created, equal to the flow $f(e_i,d_i)$ through edge (e_i, d_i). The sum of the flows going into node d_i is smaller than the sum of the flows leaving node d_i. The difference between the sum of the ingoing flows and the sum of the outgoing flows is negative, and is referred to as 'deficit flow' at node d_i:

$$df_i = \sum_{k \in \delta+} f(k, d_i) - \sum_{k \in \delta-} f(d_i, k) < 0 \tag{8}$$

Accordingly, node d_i is referred to as a 'deficit node'.

It is important to note, that the definition of excess nodes and deficit nodes given in Todinov[18] (equations (7) and (8)) is different from the definition of excess and deficit nodes given in Dong et al.[6] (equations (1) and (2)). The definition given by (1) and (2) is based on the capacities of the ingoing and outgoing edges while the definition given by (7) and (8) is based on the actual ingoing and outgoing flows of a node. The definition given by (7) and (8) is related to networks with a feasible flow, for which the edge capacity constraints are fulfilled for every single edge and the flow conservation law is fulfilled for every single internal node. According to the definition in Todinov[18], specified by (7) and (8), excess and deficit flows do not exist anywhere in the network before an edge fails. Excess and deficit nodes are created only after edge failures.

Let us connect all excess nodes e_i with a new source s_d by edges with capacity equal to the excess at the excess nodes. Similarly, let us connect all deficit nodes d_i to a new sink t_d by edges with capacities equal to the deficit flows at the deficit nodes. In this way, from the original network, a dual network is obtained, with a different source and sink (Fig. 6b). The maximum throughput flow in the original flow network can be obtained after augmenting $s_d - t_d$ paths in the dual network. The magnitude of the maximum throughput flow in the original network, after the edge failures, is specified by the next theorem.

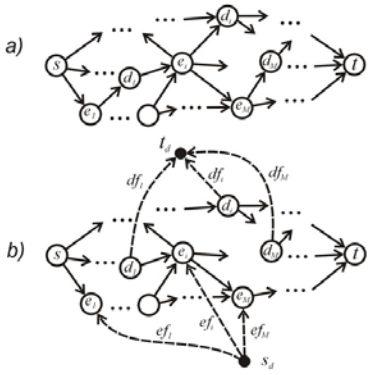


Figure 6. Re-optimising the flow in a network after failure of several edges.

Theorem 1 (Maximum flow after edge failures, Todinov[17,18,19]. If $q_d^{\max}$ is the maximum throughput flow in the dual network, the maximum possible throughput flow in the original network is $Q_{\max} - \sum_i f(e_i, d_i) + q_d^{\max}$.

Consider now the static network in Fig.7, whose edges are fully saturated with flow. Because the edges of the network are fully saturated with flow, excess and deficit nodes will exist. Suppose that nodes e_i (i=1,...,M) are excess nodes and nodes d_i (i=1,...,N) are deficit nodes (the rest of the internal nodes are balanced nodes). The deficit and excess nodes in the network are defined in the sense of Dong et al.[6] (equations (1) and (2)). If imbalanced nodes are present, the network flow is not feasible at the start, and the purpose is to make it feasible and maximise the throughput flow, by an appropriate flow redistribution between excess and deficit nodes and by draining flow from the network.

Now let us connect all excess nodes with the sink *t*, by *ghost edges* directed to the sink, with flow capacities equal to the amount of excess at the excess nodes (Fig.7). Simultaneously, let us also connect all deficit nodes with the source *s*, by ghost edges directed towards the deficit nodes, with flow capacities equal to the deficit flow at the deficit nodes. This operation transforms the original network into a network where all internal nodes are balanced (Fig.7). The ghost edges are drawn by dashed lines.

In the network in Fig.1a, there is a single excess node (node 1) with excess flow equal to 10 units and a single deficit node (node 2), with a deficit flow equal to 10 units (Fig.8a). Node 1 can be connected to the sink t by a ghost edge directed towards the sink, with a flow capacity of 10 flow units. The source *s* can also be connected to the deficit node 2, by a directed ghost edge with a flow capacity of 10 units. These operations transform the network in Fig.1a into the network in Fig.8a, where all internal nodes are balanced. The ghost edges in Fig.8a have been marked by dashed lines.

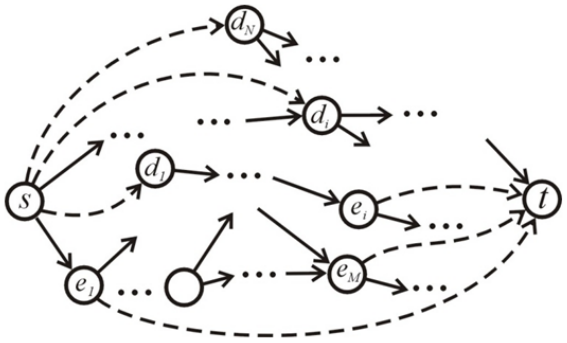


Figure 7. By using ghost edges (the dashed lines), the excess and deficit node in any static flow network with fully saturated edges can be transformed into a network where all internal nodes are balanced.

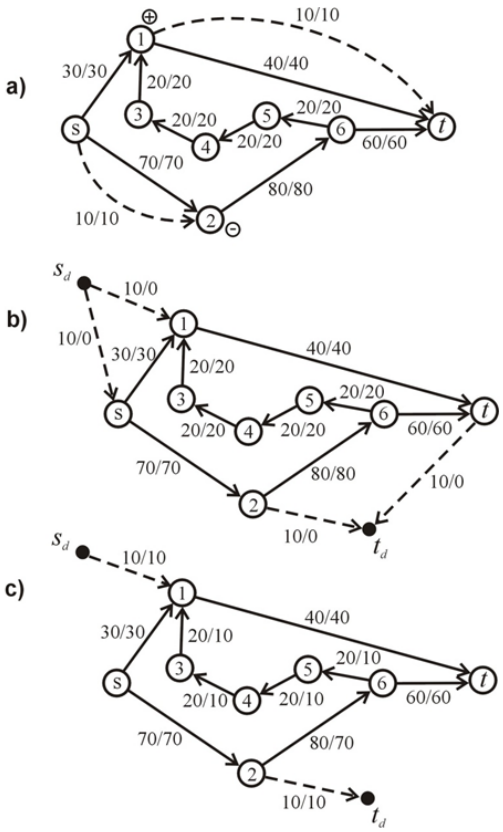


Figure 8. a) The network from Fig.1a, transformed by introducing ghost edges, so that all nodes become balanced; b) A dual network of the network in Fig.8a; c) A dual network of the network in Fig.8a, after removing (cancelling) edges (s_d, s) and (t, t_d), characterised by the same capacity.

After the introduction of ghost edges, the network flow is now feasible everywhere. In other words, the flow conservation at the nodes and the capacity constraints of the edges are honored in the network. The maximum throughput flow $Q_{\max,g}$ in the network, which includes all ghost edges, can be determined immediately – it is equal to the sum of capacities of all edges going into the sink.

$$Q_{\max,g} = \sum_i c(i,t) + \sum_k c_g(k,t) \tag{9}$$

In equation (9), $\sum_i c(i,t)$ is the sum of capacities of all real edges going into the sink *t* and $\sum_k c_g(k,t)$ is the sum of capacities of all ghost edges going into the sink *t*. For the network in Fig.8a for example, $Q_{\max,g} = 110$ units of flow.

Considering the flow conservation in the network, the maximum throughput flow can also be presented as a sum of the capacities of all edges coming out of the source *s*,

$$Q_{\max,g} = \sum_i c(s,i) + \sum_m c_g(s,m) \tag{10}$$

where $\sum_i c(s,i)$ is the sum of capacities of all real edges coming out of the source *s* and $\sum_m c_g(s,m)$ is the sum of capacities of all ghost edges coming out of the source *s*.

Now, suppose that all ghost edges in the network from Fig.8a 'fail' simultaneously. Because, by saturating all edges with flow, the throughput flow had essentially been maximised, the problem is now reduced to the problem handled by Theorem 1, for a network with a maximised throughput flow, in which several edges (the ghost edges) fail. Because the conditions of Theorem 1 are fulfilled, it can be applied to determine the maximum throughput flow in the network, after the 'failure' of all ghost edges. Because, removing ghost edges essentially transforms the network in Fig.8a into the original network from Fig.1a, applying Theorem 1 essentially determines the maximum throughput flow in the original network from Fig.1a.

Following Theorem 1, a dual network is formed by connecting the excess nodes to a new source s_d, and the deficit nodes to a new sink t_d (Fig.8b). Because the original source *s* is connected to all deficit nodes, there is a single edge (s_d, s), connecting the new source s_d with the original source *s*, whose flow capacity is equal to the sum of the deficit flows at all deficit nodes. Because, the original sink *t* is connected to all excess nodes, there is also a single edge (t, t_d), connecting the original sink *t* with the new sink t_d, whose flow capacity is equal to the sum of the excess flows at all excess nodes.

Paths connecting the new source s_d with the new sink t_d in the dual network are augmented, until no more augmentable paths can be found. Suppose that $q_d^{\max}$ is the sum of the flows along the edges connecting the excess nodes and the new source s_d. The quantity $q_d^{\max}$ is essentially the maximum throughput flow in the dual network.

By making use of equation (9) and Theorem 1, the maximum throughput flow in the original network (without ghost edges) becomes

$$Q_{\max} = \sum_i c(i,t) + \sum_k c_g(k,t) - [\sum_i c_{gi} - q_d^{\max}] \tag{11}$$

where $\sum_i c_{gi}$ is the sum of capacities of all ghost edges. The term $\sum_i c_{gi}$ however, can also be presented as:

$$\sum_i c_{gi} = \sum_k c_g(k,t) + \sum_m c_g(s,m) \tag{12}$$

where $\sum_k c_g(k,t)$ is the sum of capacities of all ghost edges going to the sink *t* and $\sum_m c_g(s,m)$ is the sum of capacities of all ghost edges coming out of the source *s*. Expression (12) holds because each ghost edge is either coming out of the source *s* or is going into the sink *t* and no ghost edge connects directly the source *s* and the sink *t*. Substituting equation (12) in equation (11) yields

$$Q_{\max} = \sum_i c(i,t) - \sum_m c_g(s,m) + q_d^{\max} \tag{13}$$

for the maximum throughput flow.

Equation 13 has been derived from the 'maximum flow after component failures theorem' (Theorem 1) and is, in effect a new fundamental result for the maximum throughput flow in a static flow network.

If equation (10) was used as an expression for the maximum throughput flow in the network with ghost edges, the equation

$$Q_{\max} = \sum_i c(s,i) + \sum_m c_g(s,m) - [\sum_i c_{gi} - q_d^{\max}] \tag{14}$$

would be obtained from Theorem 1, which, after the substitution of equation (12), yields

$$Q_{\max} = \sum_i c(s,i) - \sum_k c_g(k,t) + q_d^{\max} \tag{15}$$

for the maximum throughput flow $Q_{\max}$.

Equation (15) is an alternative equation for the maximum throughput flow. In equation (15), $\sum_k c_g(k,t)$ is the sum of the excess flow at all excess nodes. In equation (13), $\sum_m c_g(s,m)$ is the sum of the deficit flow at all deficit nodes.

The results (15) and (13) can be summarized in the next two theorems:

Theorem 2 (dual network theorem for static flow networks) The maximum throughput flow in any static flow network is equal to the sum of capacities of all edges coming out of the source, minus the total excess flow at the excess nodes, plus the maximum throughput flow in the dual network.

Theorem 3 (alternative dual network theorem for static flow networks) The maximum throughput flow in any static flow network is equal to the sum of capacities of all edges going into the sink, minus the total deficit flow at the deficit nodes, plus the maximum throughput flow in the dual network.

The dual network theorem establishes a very important link between the maximum throughput flow in a static flow network and the maximum throughput flow in its dual network. The dual network theorem replaces the task of determining the maximum throughput flow in the original network with the task of determining the maximum throughput flow in the dual network. In the case where there are few unbalanced nodes in the original network, determining the maximum throughput flow in the dual network is a task which is significantly easier than the task of determining the maximum throughput flow in the original network. The main reason for this important tradeoff is that the dual network is already saturated with flow. As a result, in case of fewer imbalanced nodes, fewer path augmentations are normally needed for maximising the throughput flow in the dual network. Equations 13 and 15 can be checked easily. Excluding $q_d^{\max}$ from the system of equations 13 and 15 results in

$$\sum_i c(s,i) - \sum_k c_g(k,t) = \sum_i c(i,t) - \sum_m c_g(s,m) \tag{16}$$

which is the flow conservation law in static networks: 'the sum of capacities of all edges coming out of the source minus the total excess flow at the excess nodes is equal to the sum of capacities of all edges going into the sink minus the absolute value of the total deficit flow at all deficit nodes'.

Applying Theorem 2 and equation (15) to the network in Fig.8a, yields $Q_{\max} = (70+30) - 10 + 10 = 100$ flow units because (the maximum throughput flow in the dual network is $q_d^{\max} = 10$). Applying Theorem 3 and equation (13) to the network in Fig.8a also yields $Q_{\max} = (60+40) - 10 + 10 = 100$ for the maximum throughput flow.

For static flow networks, Theorem 2 should be used for maximising the throughput flow in cases where the maximum throughput flow is of interest and not the actual edge flows. Such an application exists in building fast discrete-event simulators for determining the performance of repairable flow networks where the maximum throughput flow after a contingency event (e.g. edge failure) and not the edge flows are important. In future research, Theorems 2 and 3 could also be instrumental in proving new fundamental results about static flow networks.

The edge flows corresponding to the maximum throughput flow in a static flow network can be determined by following the procedure outlined in Todinov[17]. The first stage consists of maximising the throughput flow in the dual network with a new source s_d and new sink t_d, by augmenting the shortest feasible paths, until no more augmentable paths can be found.

The second stage follows the first stage and consists of path augmentation in the *dual circulation network*. This is obtained by including a circulation (*t,s*) edge in the dual network, with capacity $c(t,s)$, equal to the sum of capacities of the edges going out of the original source *s* or the sum of capacities of the edges going into the sink *t*, whichever is smaller:

$$c(t,s) = \min\{\sum_i c(s,i);\ \sum_k c(k,t)\}$$

The initial edge flows with which the augmentation of the dual circulation network starts, are the edge flows obtained after maximising the throughput flow in the dual network.

The throughput flow in the as-defined dual circulation network is maximised by augmenting the shortest s_d - t_d paths, until all outgoing edges from the new source s_d are fully saturated with flow. This is always guaranteed, and a proof of this fact has been given in Todinov[17,21].

Going back to Fig.8c, because, the deficit and excess flow have been eliminated during the flow redistribution stage, all edges coming out of the new source s_d are fully saturated with flow and no flow augmentation in the dual circulation network is needed. The edge flows, corresponding to the maximum throughput flow are shown in Fig.8c.

3 Improving the average running time of maximising the throughput flow in the dual network

Theorems 2 and 3 suggest an algorithm for maximising the throughput flow in a static flow network. The average running time of maximising the flow in the dual network, can be improved significantly, if the two-stage algorithm proposed in Todinov[20] is used for this purpose. The essence of this algorithm is to augment the shortest s_d - t_d paths in the dual network which contain only backward edges, except the first edge which comes out of the source s_d and the last edge which goes to the new sink t_d. The path augmentation continues until no more augmentable paths with backward edges belonging to the original network can be found. The second stage of the algorithm includes augmenting the shortest s_d - t_d paths that may include both, forward and backward edges belonging to the original network.

The augmentation of the shortest s_d - t_d paths is a Ford-Fulkerson type of augmentation. A forward edge (i,j) is admissible for augmentation if $f(i,j) < c(i,j)$ holds, where $f(i,j)$ is the flow through edge (i, j) and $c(i,j)$ is the capacity of edge (*i,j*). A backward edge (i,j) is admissible for augmentation if $f(i,j) > 0$ is fulfilled. The slacks $\Delta = c(i,j) - f(i,j) \geq 0$ characterizing the forward edges (i, j) and the absolute values of the flows $\Delta = \mid f(i,j) \mid \geq 0$ through the backward edges (i, j) are compared and the smallest value $\Delta_{\min}$ is selected. The value $\Delta_{\min}$ is the bottleneck residual capacity. If $\Delta_{\min} > 0$, the flow path has a nonzero residual capacity and is admissible for augmentation.

Saturating the flow path with flow $\Delta_{\min}$, proceeds by increasing the flow on forward edges by $\Delta_{\min}$

$$f'(i,j) = f(i,j) + \Delta_{\min}$$

and decreasing the flow on backward edges by $\Delta_{\min}$

$$f'(i,j) = f(i,j) - \Delta_{\min}$$

This process creates a bottleneck in the flow path: a fully saturated forward edge or an empty backward edge. The flow through the path can be increased by $\Delta_{\min}$ without violating the capacity constraints on the edges and the flow conservation law at the nodes belonging to the path.

Very often, the throughput flow in the dual network is found during the first stage of s_d - t_d paths augmentation, which includes only s_d - t_d paths with backward common edges with the original network. Often, there is either no need for a second stage or there is a need of very few augmentations of s_d - t_d paths which include both backward and forward common edges with the original network.

Suppose that the maximum throughput flow in the dual network is found during the first stage only (augmenting $s_d - t_d$ paths with backward edges only), or during the first stage and after augmenting few s_d - t_d paths which have both backward and forward common edges with the original network. In this case, it can be shown that the worst-case running time of the algorithm is $O(m^2)$, where *m* is the number of edges in the original network.

Indeed, the time complexity of the algorithm determining an augmentable $s_d - t_d$ path with backward common edges with the original network is $O(m)$ (Cormen et al.[3]). Finding the bottleneck flow (the edge with the smallest flow) and augmenting the flow along the path is also of time complexity $O(m)$. After each augmentation of a path with backward edges only, at least a single bottleneck is created in the dual network – a fully saturated forward edge coming out of the new source s_d, a fully saturated forward edge going into the new sink t_d or an empty backward edge, belonging to both the original network and the augmented $s_d - t_d$ path. As a result, after each augmentation of an $s_d - t_d$ path, at least a single bottleneck edge is essentially excluded from the dual network. Once a bottleneck edge has been excluded from the dual network, it is never augmented again, because there are no flow reversals along the edges, during the stage which involves only $s_d - t_d$ paths with backward common edges with the original network.

As a result, after at most m+ n_{e+d} augmentations, where *m* is the number of edges in the original network and n_{e+d} is the maximum possible number of deficit and excess nodes, all $s_d - t_d$ paths with backward common edges with the original network, will contain a bottleneck and will no longer be augmentable. Therefore, the worst-case running time of the first augmentation stage is $O(m^2)$. The average running time will also be $O(m^2)$ if a second augmentation stage is necessary, which involves augmentation *of* $s_d - t_d$ paths, with both backward and forward common edges with the original network, and the number of augmented $s_d - t_d$ paths does not significantly exceed the number of augmented $s_d - t_d$ paths in the first stage.

Here we must point out that an essential part of the proposed algorithm is that *the shortest* $s_d - t_d$ paths in the dual network must be augmented. Otherwise, for some networks the algorithm could have unacceptably low running time. This can be readily demonstrated with the network shown in Fig.9, where there are four internal imbalanced nodes: the excess node 5, with excess flow equal to $2\times10^9 - 1$, the excess node 4, with excess flow equal to 1, the deficit node 2, with a deficit flow equal to –($2\times10^9 - 1$) and the deficit node 3 with a deficit flow equal to -1.

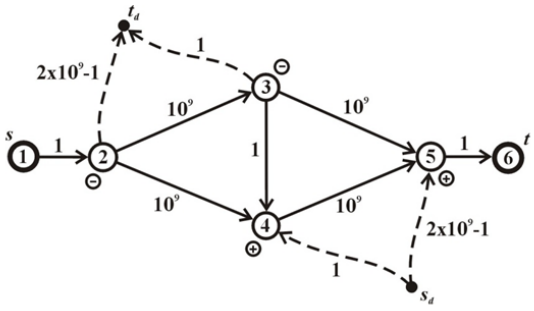


Figure 9. A slow convergence of the solution could be produced if arbitrary $s_d - t_d$ paths and not the shortest $s_d - t_d$ paths are augmented in the dual network.

Initially, all edges of the network shown in Fig.9 are fully saturated with flow. Suppose that the first identified augmentable path in the dual network is path ($s_d, 4, 3, t_d$), which can be augmented with 1 unit of flow. As a result, edge (3,4) will become empty. Suppose that the next selected augmentable path in the dual network is the augmentable path ($s_d, 5, 3, 4, 2, t_d$). This path can be augmented with the maximum possible flow of one unit. As a result, edge (3,4) will be fully saturated again. Now suppose that the next selected augmentable path is ($s_d, 5, 4, 3, 2, t_d$), which is augmented with the maximum possible flow of one unit. If the sequence of augmentable paths ($s_d, 5, 3, 4, 2, t_d$) and ($s_d, 5, 4, 3, 2, t_d$) is repeated, there will be 2×10^9 path augmentations before the dual network is fully saturated! However, if the shortest augmentable paths were used, after the initial augmentation of path ($s_d, 4, 3, t_d$), the maximum throughput flow could be found only after 2 path augmentations. The shortest path ($s_d, 5, 3, 2, t_d$) is augmented with the maximum possible 10^9 units flow, followed by augmenting the shortest path ($s_d, 5, 4, 2, t_d$) with the maximum possible flow of $10^9 - 1$ units. After the saturation of the dual network, the maximum throughput flow equal to 1 will be set in the original network.

4 Eliminating parasitic directed loops of flow

It has been stressed in Section 1 that a correct algorithm for maximising the flow in static networks by starting from a fully saturated network, needs to identify and eliminate parasitic loops of flow. The procedure described in this section provides an algorithm for identifying and eliminating all directed loops of flow in a network with feasible flow, where the capacity constraints on the edges and the flow conservation law at the nodes are honoured.

The procedure identifies directed loops in the network by using a modified depth-first search (mdfs-procedure), whose algorithm in pseudo-code is given below. Nodes encountered by the mdfs-procedure are marked as visited in the array marked[], by the statement 'marked[r_node]=1'. Initially all entries in the array marked[] are set to zero. Apart from the array marked[], another array called done[] is also maintained. Initially, all entries of the done[] array are also set zero. The mdfs() procedure is called with the index of the source node *s* as a parameter. We need to point out here that the modified mdfs() procedure does not scan all successors of the current node. It scans all *eligible* successors. 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 (there is some flow $f(r\_node, i) > 0$ along the edge). If the edge (r_node,i) is empty, the successor *i* is not considered by the mdfs() procedure.

Algorithm 1. Algorithm of the modified mdfs-procedure for discovering a directed loop in a network with directed edges

```
procedure retrieve_directed_flow_loop(cur_node)
{  // retrieves the identified directed loop}

procedure mdfs(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;
                                    mdfs(cur_node);
                                  }
            else {
                    if (done[cur_node]=0) then {    pred[cur_node]=r_node;
                                                    retrieve_directed_flow_loop(cur_node);
                                                    break;
                                                }
                 }
    }

    done[r_node]=1;
 }

Statements before the call of the mdfs –procedure:

for i=1 to n do {marked[i]=0; done[i]=0; pred[i]=0;}
mdfs(1).
```

A directed 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 mdfs(cur_node) is still active, in other words, its activation record is still in the stack. After the end of the mdfs(cur_node) call, node 'r_node' is marked as 'done' by the statement 'done[r_node]=1'. This is why, when both 'marked[cur_node]=1' and 'done[cur_node]=0' are encountered during the search, a directed loop of non-zero flow has been discovered. This is subsequently retrieved 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 flow loop 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 *k* is equal to a descendent of the node 'cur_node'. This check is used for identifying the node which closes the identified directed flow loop.

After discovering the directed flow loop, the algorithm determines the edge carrying the smallest amount of flow and subtracts this flow from the flows of all edges belonging to the directed loop. As a result, at least one edge from the loop becomes empty. Once an edge becomes empty, it remains empty until the end of the procedure for removing directed flow loops. This loop will not be discovered again during subsequent searches with the mdfs() procedure because one of its edges is empty and therefore not eligible as a 'descendent'. The empty edge essentially breaks the directed flow loop.

Subtracting the flow from a directed flow loop eliminates the parasitic loop without affecting the throughput flow in the network.

Identifying a directed loop with the mdfs() procedure and subtracting the bottleneck flow from all of its edges, has a worst-case complexity O(m), where m is the number of edges in the network. Because after each flow subtraction, at least a single edge remains empty, after at most m steps (equal to the number of edges in the network) all flow loops will be removed. Consequently, the described procedure for removing all parasitic loops of flow has a worst-case running time $O(m^2)$.

Applying the procedure to the network in Fig.5 discovers the directed loop (2,3,4,2) carrying 10 units bottleneck flow. Subtracting this bottleneck flow from the flows of all edges belonging to the loop, eliminates the flow loop. The maximum throughput flow is still 30 units but no directed flow loops are present.

5 Application of the dual network theorem for determining the maximum throughput in a static flow network

The algorithm for determining the maximum flow in a static network by 'failure' of ghost edges can be simplified significantly by noticing that during the flow redistribution stage (maximising the throughput flow in the dual network), the source s or sink t in the original network cannot be visited by the augmented path (Fig.10a).

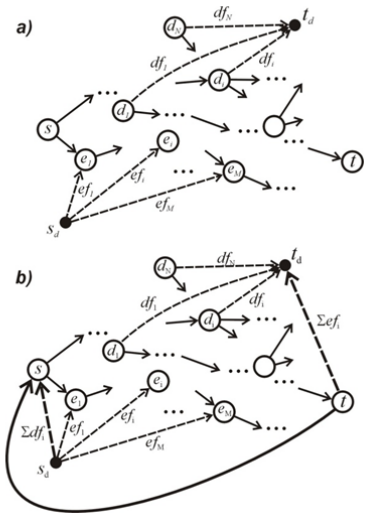


Figure 10. a) Flow distribution stage in the dual network and b) Flow draining stage in the dual circulation network for a static network with imbalanced nodes.

Indeed, suppose, that such a possibility exists. Without loss of generality, let us examine the first augmented path that starts from the new source s_d and at some point visits the original source s.

In order to reach the new sink t_d, the augmented path must get out of the original source s. Because, according to our hypotheses, this is *the first* augmentation path that visits the source s, all outgoing edges from the source s, except the one from which the source s has been visited by the augmented path, must be fully saturated with flow. There is no possibility of augmenting a path passing through the original source s, because no flow can be increased along any of the edges coming out of the source s. Similar considerations apply to the first augmented path visiting the original sink t.

Consequently, during the flow redistribution stage in the dual network (Fig.10a), no augmentable path can visit the source *s* or the sink *t*. As a result, at the end of the first stage of the algorithm, the edge connecting the new source s_d and the original source *s* and the edge connecting the original sink *t* and the new sink t_d will always remain empty. Therefore, the dual network can be simplified significantly, by ignoring the edge connecting the new source s_d and the original source *s* and the edge connecting the original sink *t* and the new sink t_d (Fig.10a). These are included only during the second stage – the draining of the dual circulation network (Fig.10b).

During the draining stage, a circulation edge (*t,s*) is introduced (Fig.10b), with capacity equal to either the sum of the capacities of the edges coming out of the source *s* or the sum of capacities of all edges going into the sink *t*, whichever is smaller. By using the circulation edge (*t,s*), the flow can be augmented directly through the path (s_d,*s*,*t*,t_d). If there are two edges (s_d,*s*) and (t,t_d) connecting the old source with the new source and the old sink with the new sink, the edge with the smaller flow capacity, either edge (s_d,*s*) or edge (t,t_d), can then be removed from the network without affecting the procedure. The capacity of the other edge is corrected by subtracting from it the smaller capacity of the removed edge. If the sum of the excess flow at the excess nodes is equal to the absolute value of the sum of the deficit flow at the deficit nodes, both edges (s_d,*s*) and (t,t_d), can be removed from the network. This is exactly the case for the network in Fig.8b. After removing edges (s_d,*s*) and edge (t,t_d), the resultant network is shown in Fig.8c.

The algorithm for maximising the throughput flow in any static network has several distinct stages:

1. Flow redistribution stage (maximises the throughput flow in the dual network, Fig.10a). If all edges coming out of the new source s_d are fully saturated with flow, the maximum throughput flow in the original network has been found. If there is at least a single edge coming out of the new source s_d, which has not been fully saturated with flow, the algorithm continues with the next, draining stage. If only the maximum throughput flow is needed and not the edge flows, the algorithm ends with this stage.

2. Draining stage. Maximises the throughput flow in the dual circulation network, until all edges coming out of the new source s_d are fully saturated with flow.

3. Eliminating parasitic loops of flow. Identifies and removes parasitic loops of flow in the network with maximum throughput flow, by applying Algorithm 1. During this operation, the maximum throughput flow remains unaffected.

The algorithm can be detailed by the following basic steps.

Algorithm 2

Initial state of the network – all edges are fully saturated with flow, to their full capacity.

All deficit nodes and excess nodes in the network are known before the start of the algorithm.

1. Flow redistribution stage. (maximising the throughput flow in the dual network)

1.1. Create a dual network by linking all excess nodes with the new source s_d through edges with capacities equal to the amount of excess flows at the excess nodes. Similarly, all deficit nodes are linked with the new sink t_d through edges with capacities equal to the absolute value of the deficit flows at the deficit nodes.

1.2 If (there are no deficit and excess nodes) **then**

The maximum throughput flow in the network is equal to the sum of the

capacities of the edges coming out of the source *s*. **Go to stage 3**.

1.3 If (there are both deficit and excess nodes) **then**

Augment the shortest s_d - t_d paths in the dual network, until no more augmentable s_d - t_d paths can be found.

Record the total amount of throughput flow $q_d^{\max}$ in the dual network.

1.4 The maximum throughput flow in the original network is equal to the sum of capacities of all edges coming out of the source *s*, minus the total amount of excess flow at all excess nodes, plus the maximum throughput flow $q_d^{\max}$ in the dual network.

If (only the maximum throughput flow is required and not the edge flows) **then Stop**.

If (all edges coming out of the new source are fully saturated) **then go to stage 3**. //*there is no need of a draining stage*

2. Draining stage. (maximises the throughput flow in the dual circulation network)

2.1 Introduce a *t*-*s* circulation edge, with capacity equal to either the sum of the capacities of the edges going into the sink *t* or the sum of the capacities of the edges coming out of the source s, whichever is smaller (Fig.10b).

Find the sum Δ of the excess flow at the excess nodes and the deficit flow (taken with minus sign) at the deficit nodes.

If the sum Δ is positive ($\Delta > 0$), connect the old sink *t* with the new sink t_d, by a directed edge with capacity Δ. If the sum is negative ($\Delta < 0$), connect the new source s_d with the original source *s* by a directed edge with capacity $|\Delta|$.

2.2 Augment the shortest s_d - t_d paths in the dual circulation network, until all edges coming out of the new source are fully saturated with flow and no more augmentable s_d - t_d paths can be found. This is guaranteed, according to Todinov[17].

3. Eliminating the parasitic loops of flow.

This stage proceeds according to Algorithm 1 described in Section 4.

Note that if no deficit and excess nodes exist simultaneously, the flow redistribution stage, consisting of augmenting the shortest s_d - t_d path in the dual network is skipped and the algorithm continues with the draining stage.

The proof of the maximum flow after edge failure theorem in Todinov[17] is essentially a proof of optimality for the proposed algorithm. This proof guarantees that the algorithm will always determine the maximum throughput flow and during the draining stage, a state will be reached where all edges coming out of the new source will be fully saturated with flow and the algorithm will terminate.

5 Solved examples

The work of the proposed algorithm will be illustrated by the example in Fig.11a. In the network, there is a single excess node (node '5') and a single deficit node (node '8'). The excess flow at the excess node is $ef_5 = 12$. The deficit flow at the deficit node is $df_8 = -9$. In the dual network (Fig.11b), edge $(s_d, 5)$ connects the new source s_d with the excess node '5' and edge $(8, t_d)$ connects the deficit node '8' with the new sink t_d. The capacity of connecting edge $(s_d, 5)$ is 12 flow units – equal to the excess flow at node '5'. The capacity of connecting edge $(8, t_d)$ is 9 units – equal to the absolute value of the deficit flow at node '8'.

The maximum throughput flow in the dual network is attained by augmenting the shortest path $(s_d, 5, 6, 8, t_d)$ with 4 units of flow. After this operation, there are no more augmentable paths in the dual network and the maximum throughput flow in the dual network is $q_d^{\max} = 4$.

The total excess flow at all excess nodes is 12 units. The sum of capacities of all edges coming out of the source s is 32 units. According to Theorem 2, the maximum flow in network 7a is equal to 32-12+4=24 units.

The edge flows corresponding to this maximum flow are obtained by following the steps of Algorithm 2. The dual circulation network is shown in Fig.11c. The shortest path $(s_d, 5, 1, 12, t_d)$ is augmented with 3 units flow, while the next shortest path $(s_d, 5, 1, 12, 8, t_d)$ is augmented with 5 units flow. As a result, after removing the connecting edges, the excess and deficit flow disappear and the final state of the edge flows is given in Fig.11d. The maximum throughput flow is indeed 24 units.

Finally, the stage 3 of the algorithm identifies and removes the directed flow loop (4,6,5,4) with flow of 2 units. After removing the directed loop of flow from the network, the final edge flows are according to Fig.11e. In this network, there are no more directed loops of flow.

In the network in Fig.12a, there are two excess nodes '7' and '10' and a single deficit node '6'. The excess flow at the excess nodes is $ef_7 = 11$ and $ef_{10} = 6$. The total excess flow is $ef_7 + ef_{10} = 17$ units. The deficit flow at the deficit node is $df_6 = -10$. In the dual network (Fig.12b), two edges $(s_d, 7)$ and $(s_d, 10)$ connect the new source s_d with the excess nodes and edge $(6, t_d)$ connects the deficit node '6' with the new sink t_d. The capacities of the connecting edges are equal to the excess at the excess nodes or the absolute value of the deficit flow, at the deficit node. The dual network can be augmented with $q_d^{\max} = 10$ units of flow, which is the maximum throughput flow in the dual network. The shortest path $(s_d, 7, 6, t_d)$ can be augmented with 8 units flow and the next shortest path $(s_d, 10, 9, 6, t_d)$ can be augmented with 2 units of flow. As a result, node 6 becomes a balanced node. The sum of capacities of the edges coming out of the source *s* is 36 units. According to Theorem 2, the maximum flow in the network in Fig.12a is equal to 36-17+10=29 units.

The edge flows corresponding to the maximum flow of 29 units are obtained by following the steps of the algorithm.

The dual circulation network is created by: (i) introducing a *t*-*s* circulation edge with capacity equal to 29 units (the sum of the capacities of the edges going into the sink is 29 and it is smaller than the sum of the capacities (36) of the edges going out of the source *s*), and (ii) introducing edge (t, t_d) with capacity 7 units, equal to the sum of the total excess flow at the excess nodes and the total deficit flow at the deficit nodes in the original network ($17 - 10 = 7$).

After augmenting path (s_d , 10,5,1,11, t_d) with 4 units of flow and path (s_d , 7,2,1,11, t_d) with 3 units of flow, the excess flow at the excess nodes disappears and the final state of the edge flows is given in Fig.12d. The maximum throughput flow is indeed 29 units. There are no directed loops of flow in the network.

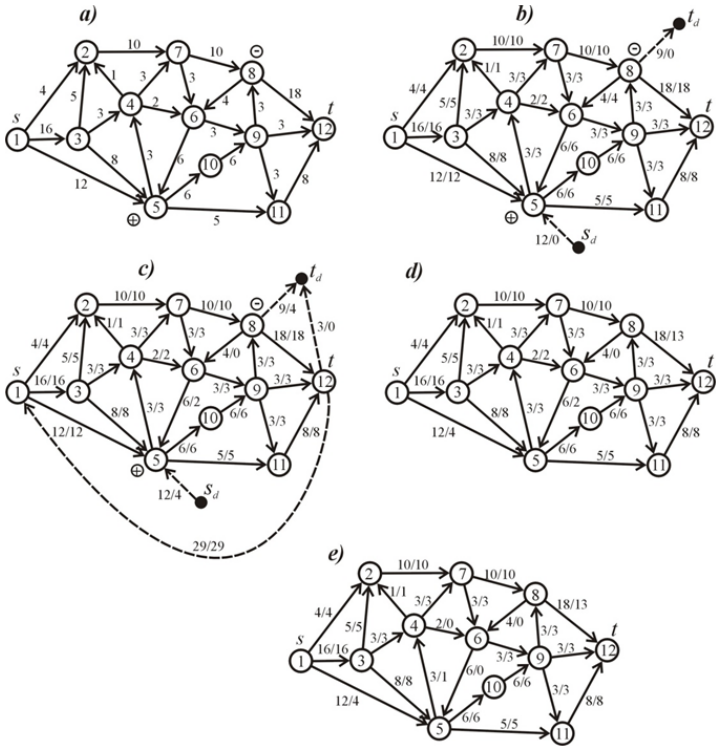
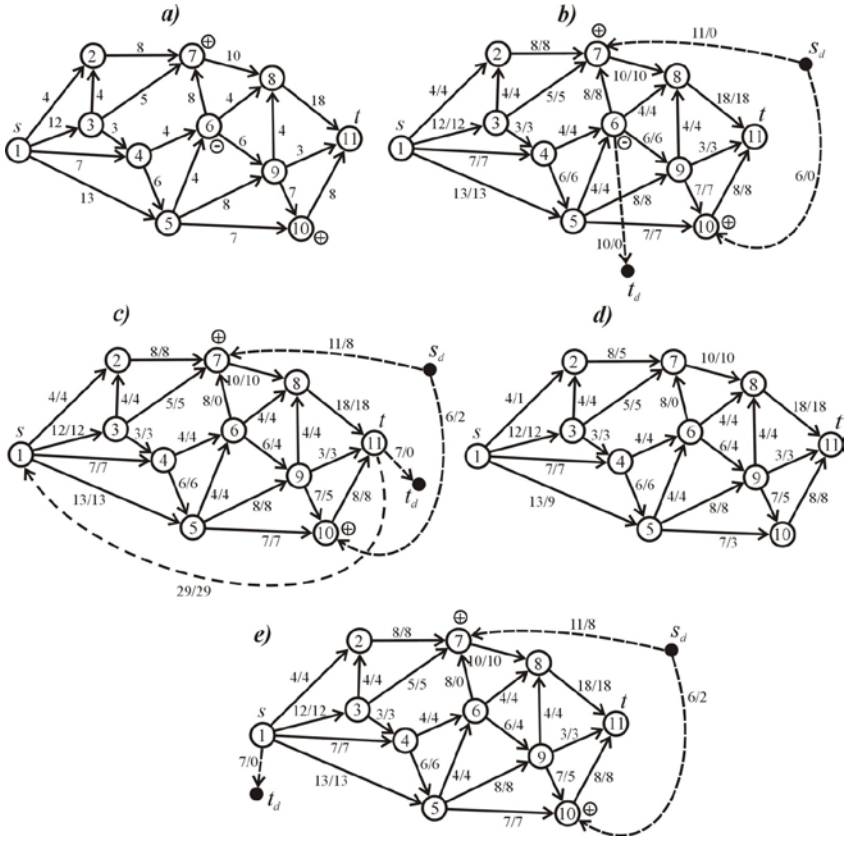


Figure 11. A network with a single excess node and a single deficit node.

Figure 12. A network with two excess nodes and a single deficit node



Because, there are only excess nodes after the augmentation of the dual network, the algorithm can further be simplified by removing the *t-s* circulation edge and connecting the new sink t_d directly to the old source s with an edge $(1, t_d)$ with capacity equal to the sum of the total excess flow at the excess nodes and the total deficit flow at the deficit nodes (taken with minus sign) in (Fig.12e). Paths (s_d ,10,5,1, t_d) and (s_d ,7,2,1, t_d) are then augmented in the dual circulation network, until no more augmentable paths can be found.

A similar simplification can be made if, at the end of the flow augmentation in the dual network, only deficit nodes remain. In this case, the circulation edge can also be removed and the new source s_d can be directly connected to the original sink *t*, through an edge with capacity equal to the absolute value of the sum of the total excess flow at the excess nodes and the total deficit flow at the deficit nodes in the original network.

6 Advantages and limitations of the proposed throughput flow maximisation algorithm

The proposed algorithm, based on maximising the throughput flow in the dual network, will be very efficient in networks where most of the nodes are well-balanced nodes, and very few excess and deficit nodes exist. In most of these cases, the classical algorithms for maximising the throughput flow by filling gradually an empty network, will be outperformed by the proposed algorithm.

The condition of very few excess and deficit nodes is often closely matched by real production networks, for example oil and gas production network. Commonly, for well-designed production networks, the sum of the capacities of edges going

into a node is equal to the sum of capacities of the outgoing edges. As a result, most of the nodes in these networks are balanced nodes.

If only excess nodes are present in the fully saturated network, the maximum throughput flow is determined immediately. It is equal to the sum of capacities of all edges coming out of the source *s* minus the total excess flow at all excess nodes. If only deficit nodes are present in the fully saturated network, the maximum throughput flow is also obtained immediately. It is equal to the sum of capacities of all edges going into the sink, minus the total deficit flow at all deficit nodes. Finally, if both excess and deficit nodes are present, but there is no augmentable $s_d - t_d$ path in the dual network, the maximum throughput flow is equal to the sum of capacities of all edges coming out of the source minus the total excess flow at all excess nodes. According to Theorem 3, the maximum throughput flow is also equal to the sum of capacities of all edges going into the sink, minus the absolute value of the total deficit flow at all deficit nodes.

The running time of the algorithm varies significantly. In the case where the maximum throughput flow in the dual network is determined by augmenting a single $s_d - t_d$ path or a few $s_d - t_d$ paths, the average running time depends linearly in the size *m* of the network. Such an example will be discussed in Section 7.

In the case where most of the nodes in the network are imbalanced nodes, the proposed algorithm will have to perform a lot of work to eliminate the excess and deficit flow from the imbalanced nodes. This point is illustrated by the extreme–case network in Fig.13, where there are many excess and deficit nodes and, in addition, the capacities of the edges connecting the source and the sink to the network are small.

In this extreme case, instead of augmenting a single *s*-*t* path and terminating, the algorithm will shift unnecessarily all of the excess flow in the network, until it determines the throughput flow of 1 unit.

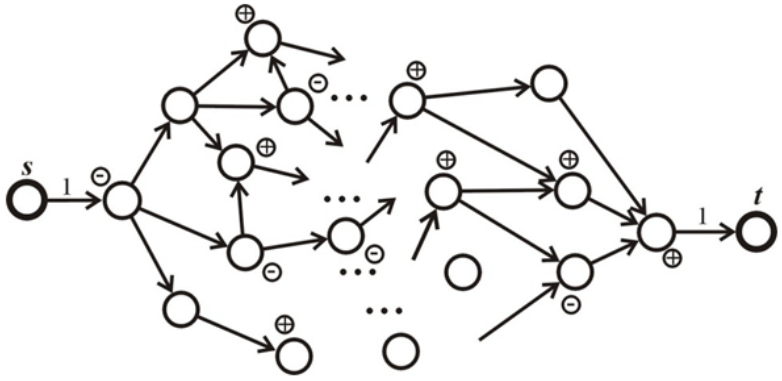


Figure 13. An extreme-case network with many imbalanced nodes. Edges with unit capacities connect the source s and the sink t with the rest of the network.

The ideal application of the proposed method is for reoptimising the throughput flow in repairable flow networks, where, upon failure of an edge, only two imbalanced nodes are formed, and in close proximity (Todinov[18]). Eliminating the momentary excess and deficit flow from the pair of imbalanced nodes resulting from the edge failure is often performed in linear time or in constant time (Todinov[17,18,19,21]. In the case of a single edge failure, the high computational speed of the algorithm proposed in Todinov[18] is due to the circumstance that the reoptimisation after an edge failure is done by altering the flows along a relatively small number of edges. In contrast, the reoptimisation by using conventional algorithms starts from empty edges and will proceed by altering the flows along most of the edges of the network. Furthermore, in the case of a single edge failure, the re-optimisation method proposed in Todinov[17,18,19] is essentially a very efficient decentralised algorithm that allows a network locally optimised by independent distributed agents, to achieve a global maximum of the

throughput flow, after sequential edge failures. This important feature of the algorithm opens the powerful possibility of real-time management of the power flows in power networks with active control.

7 Comparing the performance of the proposed algorithm for maximising the throughput flow with the performance of a conventional shortest-path augmentation algorithm

To demonstrate the clear advantages of the proposed algorithm over classical algorithms, a comparison has been made. The comparison is only in terms of the value of the maximum throughput flow, not in terms of determining the edge flows leading to the maximum throughput flow. Consequently, the extra stage of the proposed algorithm, related to removing parasitic loops of flow, has been excluded.

The throughput flow in the network from Fig.14, was maximised by using the classical Edmonds and Karp algorithm, starting from an empty network. One million runs of the Edmonds and Karp algorithm on a computer with processor Intel(R) Core(TM) 2 Duo CPU T9900 @ 3.06 GHz, took 27.9 seconds.

The maximum throughput flow in the network from Fig.14 was also determined by running the proposed algorithm. There are only two imbalanced nodes in the network: the excess node 6, with 40 units excess flow and the deficit node 5, with 40 units deficit flow. The path (6,7,5) between the excess node 6 and the deficit node 5 was augmented with 40 units. After the augmentation, the deficit and excess flow at nodes 6 and 5 disappears and the maximum flow is 360 units.

Running the new optimisation algorithms one million times was done within 1.06 seconds, 26 times faster than the conventional algorithm.

Increasing the size of the network, by adding more edges between the source *s* and nodes 6,7,5 and between the sink *t* and nodes 6,7 and 5, makes the proposed algorithm even more efficient. The algorithm will run in constant time, independent of the size of the network!

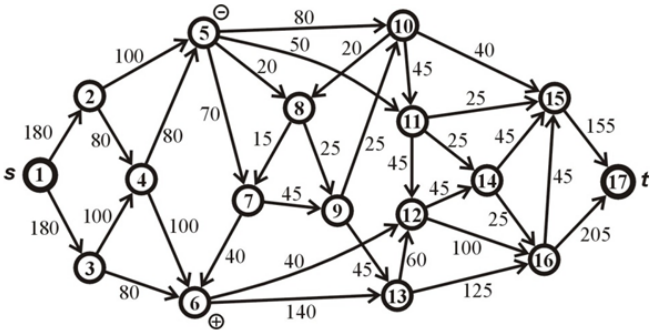


Figure 14. Test network demonstrating the high performance of the proposed algorithm in the case of a very few imbalanced nodes.

8 Conclusions

1) In any static flow network, the maximum throughput flow is also equal to the sum of the capacities of the edges coming out of the source minus the total excess flow at all excess nodes plus the maximum throughput flow in the dual network.

2) In any static flow network, the maximum throughput flow is equal to the sum of capacities of all edges going into the sink, minus the total deficit flow at the deficit nodes, plus the maximum throughput flow in the dual network.

3) For a network with a very few imbalanced nodes and fully saturated edges, instead of determining the maximum throughput flow, it is easier to determine the maximum throughput flow in the corresponding dual network. Consequently, an efficient algorithm for maximising the throughput flow in a network with edges fully saturated with flow has been proposed based on maximising the throughput flow in the dual network. For networks with very few imbalanced nodes, the proposed algorithm will outperform any classical method for maximising the throughput flow.

4) Classical shortest-path augmentation algorithms for maximising the throughput flow leave highly undesirable directed loops of flow and should not be used for optimising the edge flows in a network without an additional stage aimed at removing directed loops of flow.

5) A recursive algorithm has been proposed for identifying and removing directed loops of flow in flow networks. The algorithm can be used as an extra stage to guarantee that no directed loops of flow are left in the optimised networks.

6) The draining algorithm for maximising the flow in a network by starting from a network with edges fully saturated with flows, reported by Dong et al[6], is incorrect and leads to sub-optimal solutions.

7) The proof of the node balancing theorem presented in Ursani[22] is fundamentally flawed.

8) The worst-case running time of the draining algorithm reported in Ursani[22] is not $O(m^2 - m)$ as it has been incorrectly claimed. The worst case running time of the reported algorithm is no better than the worst case running time of the Ford-Fulkerson augmentation method for determining the throughput flow in a network.

9) The draining algorithm reported in Dong at al.[6] and its recent modification reported in Ursani[22], have an increased tendency to leave highly undesirable directed loops of flow in the optimised networks. Consequently, these algorithms should never be used for optimising network flows without an additional stage aimed at removing the parasitic loops of flow.

Acknowledgment

Gratefully acknowledged is financial help from The Leverhulme trust with the research grant F/00 382/J ‘High-speed algorithms for the output flow in repairable flow networks’.

References

[1] Ahuja R.K., T.L.Magnanti, J.B.Orlin. “Network flows: Theory, Algorithms and Applications”, Prentice Hall. 1993.

[2] Asano T., Y.Asano, Recent developments in maximum flow algorithms. Journal of the Operations Research Society of Japan. 2000; 43 (1): 2-31.

[3] Cormen T.H., T.C.E.Leiserson, R.L.Rivest, and C.Stein. “Introduction to Algorithms”. 2nd ed. MIT Press and McGraw-Hill. 2001.

[4] Elias P., A.Feinstein and C.E.Shannon. “Note on maximum flow through a network”. IRE Transactions on Information Theory. IT2, 1956; 117-119.

[5] Dinic E.A., “Algorithm for solution of a problem of maximum flow in a network with power estimation”. Soviet Math. Doklady. 1970; 11(8): 1277-1280.

[6] Dong J., W.Li, C.Cai, Z.Chen. “Draining algorithm for the maximum flow problem”. International conference on communications and mobile computing. 2009; 197-200.

[7] Edmonds J. and R.M.Karp. Theoretical improvements in algorithmic efficiency for network flow problems. Journal of the ACM. 1972; 19 (2): 248-264.
[8] Ford L.R. and D.R. Fulkerson. Maximal flow through a network. Canadian Journal of Mathematics. 1956; 8(5): 399-404.
[9] Goldberg A.V. and R. E. Tarjan. A new approach to the maximum flow problem. Journal of ACM. 1988; 35: 921-940.
[10] Hochbaum D.S.. The Pseudoflow Algorithm: A New Algorithm for the Maximum-Flow Problem. Operational Research. 2008; 56 (4): 992–1009.
[11] Hu T.C. "Integer programming and network flows". Addison-Wesley Publ.Company. 1969.
[12] Itai A. and Y.Shiloach. Maximum flows in planar networks. SIAM J. Comput. 1979; 8: 135-150.
[13] Karzanov A. Determining the maximal flow in a network by the method of preflows. Sov.Math.Dokl. 1974; 15: 434-437.
[14] Kleinberg J. and E.Tardos. "Algorithm design" Addison Wesley. 2006.
[15] Sleator D.D. and R.E.Tarjan. An O(nm log n) algorithm for maximum network flow. Technical report STAN-CS-80-831. Department of Computer Science, Stanford University, Stanford, CA. 1980.
[16] Tarjan R.E. "Data structures and network algorithms". SIAM, Philadelphia. 1983.
[17] Todinov M.T. Topology optimisation of repairable flow networks for a maximum average availability. Computers and Mathematics with applications. 2012; 64: 3729-3746.
[18] Todinov M.T. Fast Augmentation Algorithms for Maximising the Flow in Repairable Flow Networks After a Component Failure. Proceedings of the 2011 11th IEEE International Conference on Computer and Information Technology. Paphos, 2011a; 505-512. DOI 10.1109/CIT.2011.25.
[19] Todinov M.T. A fast augmentation algorithm for optimizing the performance of repairable flow networks in real time. Proceedings of ESREL conference. Troy. 2011b; 1951-1958.
[20] Todinov M.T. Analysis and optimization of repairable flow networks with complex topology. IEEE Transactions on Reliability. 2011c; 60(1): 111-124.
[21] Todinov M.T. Flow networks: Analysis and optimisation of repairable flow networks, networks with disturbed flows, static flow networks and reliability networks. Elsevier. 2013.
[22] Ursani Z. Introducing mass balancing theorem for network flow maximization. International Journal of Industrial Computations. 2012; 3. DOI: 10.5267/j.ijiec.2012.05.006.