Physica A 392 (2013) 6135–6145

Contents lists available at ScienceDirect

Physica A

journal homepage: www.elsevier.com/locate/physa

A decentralized flow redistribution algorithm for avoiding cascaded failures in complex networks

Saleh Al-Takrouri [a,b,*], Andrey V. Savkin [a]

[a] School of Electrical Engineering and Telecommunications, The University of New South Wales, Sydney, NSW 2052, Australia
[b] Palestine Polytechnic University, West Bank, Palestine

HIGHLIGHTS

- A decentralized algorithm for avoiding cascaded failures in complex networks is proposed.
- The algorithm is based on information about the closest neighbours of each node.
- The algorithm is implemented locally and does not need a centralized control system.
- A mathematically rigorous proof of convergence with probability 1 is provided.
- The maximum flow and the constant flow supply/demand problems are also considered.

ARTICLE INFO

Article history:
Received 14 November 2012
Received in revised form 15 July 2013
Available online 7 August 2013

Keywords:
Complex networks
Cascaded failure
Maximum flow
Distributed algorithms
Randomized algorithms

ABSTRACT

A decentralized random algorithm for flow distribution in complex networks is proposed. The aim is to maintain the maximum flow while satisfying the flow limits of the nodes and links in the network. The algorithm is also used for flow redistribution after a failure in (or attack on) a complex network to avoid a cascaded failure while maintaining the maximum flow in the network. The proposed algorithm is based only on the information about the closest neighbours of each node. A mathematically rigorous proof of convergence with probability 1 of the proposed algorithm is provided.

1. Introduction

The maximum flow problem has been a classical research topic in the field of complex networks. Since Ford and Fulkerson [1] introduced their augmenting paths algorithm in 1956, many algorithms have been developed to study and solve this problem [2–10]. Applications of the maximum flow problem include the determination of the maximum steady-state flow of petroleum products in pipelines, cars on roads, messages in telecommunication networks, and electricity in electrical grids [11]. It is also used in computer vision problems [12] and World Wide Web analysis [13].

Several research works developed decentralized algorithms to solve the maximum flow problem. These algorithms were mainly decentralized versions of originally centralized algorithms. For example, Segall [14] proposed three algorithms based on Refs. [1,2,15], Cheung [16] proposed an algorithm based on Ref. [1], and Awerbuch [17] proposed an algorithm based on Ref. [18]. These algorithms assume that the network contains only one source and one sink. As a result the decentralized versions are limited to single-source single-sink networks.

* Corresponding author at: School of Electrical Engineering and Telecommunications, The University of New South Wales, Sydney, NSW 2052, Australia. Tel.: +61 415210225.
E-mail addresses: saleh@unswalumni.com (S. Al-Takrouri), a.savkin@unsw.edu.au (A.V. Savkin).

0378-4371/$ – see front matter
http://dx.doi.org/10.1016/j.physa.2013.07.078

In a typical centralized algorithm, multi-source multi-sink networks are transformed into a single-source single-sink network and then the maximum flow is computed. Obviously, this transformation is not extendable to decentralized algorithms. Although recent publications considered the maximum flow problem in multi-source multi-sink planer networks without this transformation [12,19], the information regarding the status of all the nodes and the links in the network are assumed to be readily available for a centralized system to apply the algorithm. On the contrary, the algorithm proposed in this paper is natively decentralized and capable of handling multi-source multi-sink networks.

In addition, the flow in the network should be distributed such that the maximum capacities of the network elements are not exceeded. Otherwise, the network would potentially be vulnerable to cascaded failures. Cascaded failures in complex networks have been the subject of many studies [20,21] including, but not limited to, cascaded failures in urban traffic networks [22–24] and power grids [25,26].

In electrical power distribution systems, for example, cascaded failures have been attracting the attention of researchers due to the catastrophic impact they have on the networks [27–29]. A cascaded failure is initiated when a node or a link in the complex network is lost due to a random failure or a targeted attack, and the flow passing through the lost node or link is redistributed to other nodes and links in the network. If not executed properly, the redistribution may cause other nodes and links to become overloaded and therefore disconnected from the network. This effect may propagate throughout the network and the entire network may be affected [30].

Cascaded failures were the reason for large blackouts throughout the world (e.g. Refs. [31–34]). As a consequence, researchers have been developing control algorithms for power flow redistribution with minimum load loss [35,30,36,37]. Typically, the maximum power flow that can be delivered to the loads is calculated by applying linear programming which has been used since the 1970s [35,38–40,36]. However, similar to the maximum flow algorithms, linear programming requires the status of all the nodes and the links in the network to be available for a centralized control system.

Switched networks [41,42] are another type of networks that are controlled by a single server that operates locally on one node at a time, but the location of the server is a control variable that is determined using information from the entire network represented by a hybrid dynamical system [43].

The new algorithm proposed in this paper to drive the flow distribution in a complex network aims to avoid cascaded failure in the network while maintaining the maximum flow. The novel decentralized randomized control algorithm is based on simple local control rules that are based only on information about the closest neighbours of each node in the network. Unlike the algorithms discussed above where a central controller is required, the proposed algorithm is implemented locally at each node. As a result, the computational costs are reduced and less information needs to be transferred on the network.

In addition, the algorithm is directly applicable to multi-source multi-sink networks. Another advantage of the proposed algorithm is that the flow values and capacities in the network can be real numbers and not limited to integer values.

The proposed algorithm is theoretically verified. In particular, this paper provides a mathematically rigorous proof of convergence of the algorithm with probability 1 for any initial flow distribution. The behaviour of the algorithm is described by an absorbing Markov chain and the algorithm is shown to converge to one of the absorbing states. Markov chains with absorbing states are already being used to model networks [44,45]. The randomized algorithm used in this paper is inspired by the distributed random algorithm used in Ref. [46] for blanket coverage self-deployment of a network of mobile wireless sensors.

The remainder of this paper is organized as follows: the flow redistribution problem addressed in this paper is formulated in the next section along with two special cases. The proposed control algorithm is detailed in Section 3 where the algorithm is proved to converge to a solution with probability 1 and a summary of the simulation procedure is provided. Simulation results are shown in Section 4, followed by the conclusion in Section 5.

2. Problem statement

Consider a complex network modelled by a graph $G = (\mathcal{N}, \mathcal{L})$ defined by a set $\mathcal{N}$ of n nodes connected by a set $\mathcal{L}$ of l links. For a given time k, each node $i \in \mathcal{N}$ is associated with a finite real value $N_i(k)$ that corresponds to the actual net flow in the node and represents the flow supply or demand of the node. The node i is a source (supply) node if

$$0 < N_i(k) \leq N_{i_{\max}} \tag{1}$$

where $N_{i_{\max}}$ is a predefined positive value representing the maximum capacity of the source node i to supply flow. The node i is an intermediate (transshipment) node if

$$N_i(k) = 0. \tag{2}$$

The node i is a sink (demand) node if

$$N_{i_{\min}} \leq N_i(k) < 0 \tag{3}$$

where $N_{i_{\min}}$ is a predefined negative value representing the maximum capacity of the sink node i to demand flow.

Each link $\langle i, j\rangle \in \mathcal{L}$ connecting the two nodes i and j is assigned a flow value $F_{i,j}(k)$ that should not exceed a maximum capacity

$$|F_{i,j}(k)| \leq F_{i,j_{\max}}; \tag{4}$$

otherwise the link will be overloaded and may be disconnected from the network. The two nodes i and j connected through the link $\langle i, j\rangle$ communicate with each other at the discrete time instance $k = 1, 2, \ldots$ for the coordination of their operation.

The relationship between the nodes and the links in the network is represented by the $n \times l$ incidence matrix $\mathbf{A}$. If the direction of flow $F_{i,j}(k)$ in the r^{th} link $\langle i, j\rangle$ is assumed to be from node i to node j, then the r^{th} column of the matrix $\mathbf{A}$ is all zeros except for $A(i, r) = 1$ and $A(j, r) = -1$. A positive value of the flow $F_{i,j}(k)$ indicates that the flow follows the assumed direction whereas a negative value of the flow $F_{i,j}(k)$ indicates that the flow is opposite to the assumed direction.

The mass balance constraint requires the net flow in a node to equal its supply or demand:

$$N_i(k) = \sum_{j=1}^{n} A(i, r)F_{i,j}(k) \tag{5}$$

where r is the column in the matrix $\mathbf{A}$ corresponding to the link $\langle i, j\rangle$. In matrix form

$$\mathbf{N}(k) = \mathbf{A}\mathbf{F}(k) \tag{6}$$

where $\mathbf{A}$ is the incidence matrix, $\mathbf{F}(k)$ is the vector of flow values in the links, and $\mathbf{N}(k)$ is the vector of the supply and demand values of the nodes. It follows that the flow supplied by the source nodes equals the flow demanded by the sink nodes:

$$\sum_{i=1}^{n} N_i(k) = 0. \tag{7}$$

A random failure or an intentional attack on the network may cause one or more links to be disconnected. Moreover, the failure or attack may target one or more nodes disconnecting those nodes and the links connecting them from the network. In such cases it is important to redistribute the flow in the network such that the maximum flow is delivered from the source nodes to the sink nodes and no links are overloaded. Otherwise overloaded links may fail and be disconnected, triggering a cascaded failure in the entire network.

The goal of the proposed algorithm is to determine the flow values $F_{i,j}(k)$ in the network such that flow delivery is maximized with Eqs. (1)–(4) are satisfied. For a computerized controller to operate properly, a discretizing parameter d is defined to be the smallest change in flow. Thus, the algorithm can increment or decrement the flow $F_{i,j}(k)$ in the link $\langle i, j\rangle$ by multiples of d.

Assumption 1. The flow values $F_{i,j}(k)$ and their maximum values in the links are multiples of the discretizing parameter d.

It follows from the constraint (5) and Assumption 1 that the supply and demand values of the nodes $N_i(k)$ are also multiples of the discretizing parameter d.

Lemma 1. *For a complex network satisfying the node net flow limits (1), (2), (3), the link flow limits (4) and Assumption 1, there exists a finite non-empty set of solutions that satisfy the constraint (5).*

Proof. Assumption 1 along with the links' maximum capacities (4) imply that the flow in each link in the network has a finite set of values that can be assigned to it:

$$F_{i,j}(k) \in \{-F_{i,j\max}, -F_{i,j\max} + d, -F_{i,j\max} + 2d, \ldots, -d, 0, d, \ldots, F_{i,j\max} - d, F_{i,j\max}\}. \tag{8}$$

The number of elements in the set in (8) is $1 + \frac{2}{d}F_{i,j\max}$. Therefore, the number of solutions obtained from different combinations of flow values has a theoretical upper limit of

$$\prod \left(1 + \frac{2}{d}F_{i,j\max}\right), \quad \forall \langle i, j\rangle \in \mathcal{L}. \tag{9}$$

It follows that the number of solutions that satisfy the node and link flow limits and the mass balance constraint (5) is finite and less than the theoretical upper limit (9). One solution that satisfies the node and link flow limits and the mass balance constraint is the trivial no-flow solution when the flow in all the links is set to zero and no flow is supplied from the source nodes to the sink nodes

$$\begin{aligned} &N_i(k) = 0 \quad \forall i = \{1, 2, \ldots, n\} \\ &F_{i,j} = 0 \quad \forall i, j = \{1, 2, \ldots, n\}. \end{aligned} \tag{10}$$

Hence there is at least one solution that satisfies (4) and (5). □

The aim of the proposed algorithm is to converge to a solution $\bar{\mathbf{F}}(k)$ such that

$$\bar{\mathbf{F}}(k) = \arg\max_{\mathbf{F}(k)} \sum_{i=1}^{n} |N_i(k)| \tag{11}$$

for all the solutions $\mathbf{F}(k)$ in the finite non-empty set of solutions that exists as stated by Lemma 1, where $|\cdot|$ denotes the absolute value. Eq. (11) may have one unique solution or a finite number of solutions at which the maximum possible flow is delivered from the source nodes to the sink nodes. The proposed algorithm will converge to one of these solutions.

Lemma 2. *Assuming that the intermediate nodes in the network are defined by (2) and that Assumption 1 holds, a complex network satisfies the node net flow limits (1) and (3) if and only if it satisfies the link flow limits (4).*

Proof. Consider a complex network that satisfies the link flow limits (4) but does not provide maximum capacities for its sources and sinks. The maximum flow any source can supply or a sink can demand cannot exceed the total maximum flow that can be delivered by the links of the network. Therefore, a theoretical maximum net flow in each source node and sink node such that (1) and (3) are satisfied can be found as follows:

$$-N_{\min} = N_{\max} = \sum_{i,j=1}^{n} F_{i,j_{\max}}. \tag{12}$$

Also, consider a complex network that satisfies the node net flow limits (1), (2), (3) but does not have maximum flow capacities in its links. The maximum flow in any link in the network cannot exceed the total maximum flow that can be supplied by the source nodes in the network. Therefore, a theoretical maximum flow in each link such that (4) is satisfied can be found as follows:

$$F_{\max} = \sum_{i=1}^{n} N_{i_{\max}}. \quad \square \tag{13}$$

Lemma 2 implies that for a complex network that satisfies either the node net flow limits (1) and (3) or the link flow limits (4), the other condition can be formulated and hence a finite non-empty set of solutions exists as stated in Lemma 1.

2.1. Special case: maximum possible flow

The classical maximum flow problem can be stated as a special case of the general problem described above. By virtue of Lemma 2, the supply flow capacity limits of the source nodes (1) and the demand flow capacity limits of the sink nodes (3) are removed. Then starting from an arbitrary initial condition (possibly the zero-flow condition) the proposed algorithm is used to find the maximum flow that can be delivered to each sink node in the network considering the flow capacities of the links (4).

2.2. Special case: constant flow supply/demand

Consider a complex network in which each source node i is required to maintain a constant supply capacity $N_{i_s} > 0$, and each sink node i is required to maintain a constant demand capacity $N_{i_t} < 0$. The intermediate nodes in the network satisfy (2). Then, starting from an arbitrary initial condition, the network is required to satisfy

$$\begin{cases} N_i(k) \to N_{i_s}, & i \text{ is a source node;} \\ N_i(k) \to 0, & i \text{ is an intermediate node;} \\ N_i(k) \to N_{i_t}, & i \text{ is a sink node.} \end{cases} \tag{14}$$

When (14) is satisfied, the network is at steady state with the flow supply from the source nodes and the flow demand by the sink nodes are fixed and equal to the desired values N_{i_s} and N_{i_t} respectively.

If some links and/or nodes are disconnected from the network due to a random failure or an intentional attack, the flow in the network will be disrupted and has to be redistributed such that the desired supply and demand values of the nodes are restored. The proposed algorithm can achieve this by setting the maximum supply capacity of each source node i to $N_{i_{\max}} = N_{i_s}$ and the maximum demand capacity of each sink node i to $N_{i_{\min}} = N_{i_t}$. This is true because the maximum and minimum of a constant function are equal to the constant value of the function.

Assuming that the flow capacities in the links (4) do not limit the delivery of the flow and from (7), the constant supply/demand is achieved in the network when

$$\sum_{i=1}^{n} N_{i_{\max}} + \sum_{i=1}^{n} N_{i_{\min}} = 0. \tag{15}$$

3. The decentralized flow distribution algorithm

The goal of the proposed algorithm is to select the flow values in the links such that the flow supplied by the source nodes to the sink nodes is maximized without overloading the links. The flow $F_{i,j}(k)$ in each link is determined locally based on the status of its end nodes.

The algorithm assigns a control variable $U_i(k)$ for each node $i \in \mathcal{N}$ to represent the desired net flow in the node. The desired net flow values $U_i(0)$ are initialized to reflect the aim of maximizing the delivered flow as follows:

$$U_i(0) = \begin{cases} N_{i_{\max}}, & N_{i_{\min}} = 0 \text{ and } N_{i_{\max}} > 0; \\ 0, & N_{i_{\min}} = 0 \text{ and } N_{i_{\max}} = 0; \\ N_{i_{\min}}, & N_{i_{\min}} < 0 \text{ and } N_{i_{\max}} = 0. \end{cases} \tag{16}$$

This is equivalent to setting $U_i(0) = N_{i_{\max}} + N_{i_{\min}}$. Based on the desired and actual net flow values, the nodes in the network are divided into two sets.

Definition 1. A node i is said to be a *virtual source* if its desired flow value is greater than or equal to its actual flow value. The set of virtual source nodes $\mathcal{S}(k)$ is defined as

$$\mathcal{S}(k) = \{i : U_i(k) - N_i(k) \geq 0\}. \tag{17}$$

The set of virtual source nodes includes actual source nodes that have not reached their maximum capacities, intermediate nodes that have not previously tried to get more flow, and actual sink nodes that have reached their maximum capacities. While actual source nodes supply additional flow into the network, intermediate and sink nodes pass extra incoming flow they do not need to destination nodes in their neighbourhood.

Definition 2. A node i is said to be a *virtual sink* if its desired flow value is less than its actual flow value. The set of virtual sink nodes $\mathcal{T}(k)$ is defined as

$$\mathcal{T}(k) = \{i : U_i(k) - N_i(k) < 0\}. \tag{18}$$

The set of virtual sink nodes includes actual sink nodes that have not reached their maximum capacities, intermediate nodes that have previously tried to get more flow, and actual source nodes that have reached their maximum capacities. Actual sink nodes demand flow from the network, whereas intermediate and source nodes request the flow to pass to destination nodes in their neighbourhood.

Each virtual source node $i \in \mathcal{S}(k)$ tries to pass more flow to a neighbouring virtual sink node $j \in \mathcal{T}(k)$. The selection of the destination node is random. Note that $\mathcal{S}(k) \cup \mathcal{T}(k) = \mathcal{N}$.

Let $\mathcal{Z}_i(k)$ be the set of nodes connected to the node i through non-overloaded links. Then the neighbourhood of i is defined as

$$\mathcal{Z}_i(k) = \{j : \exists \langle i, j\rangle \in \mathcal{L},\ A(i, r)F_{i,j}(k) < F_{i,j_{\max}}\} \tag{19}$$

where r is the column in $\mathbf{A}$ corresponding to $F_{i,j}$. The utilization of the coefficient $A(i, r)$ is to account for the fact that if the flow direction is towards the node i, then producing an opposing flow out of the node reduces the flow in the link.

For each node $i \in \mathcal{S}(k)$ a neighbourhood of candidate destination nodes $\mathcal{C}_i(k)$ is defined such that each candidate node is connected to the node i through a non-overloaded link:

$$\mathcal{C}_i(k) = \{j : j \in \mathcal{T}(k) \cap \mathcal{Z}_i(k)\} \cup \{i\}. \tag{20}$$

This definition adds the node i to its neighbouring candidate destination nodes in order to avoid the situation where $\mathcal{C}_i(k)$ is empty.

The proposed algorithm starts by enforcing the link flow limits (4). Any link flow value $F_{i,j}(k)$ that exceeds its maximum capacity is reduced to equal $F_{i,j_{\max}}$.

The next step is to verify that the nodes in the network are not supposed to supply more flow than their maximum capacities. If the net flow of a node $N_i(k)$ is calculated to be greater than its maximum supply capacity $N_{i_{\max}}$, then the excess outflow cannot be sustained and is reduced uniformly in all the links carrying the flow out of the node.

Then, the randomized algorithm is used to maximize the delivered flow from the source nodes to the sink nodes as follows.

For each virtual source node $i \in \mathcal{S}(k)$ the set of neighbouring candidate nodes $\mathcal{C}_i(k)$ is found. Each node $j \in \mathcal{C}_i(k)$ is assigned a weight w_j

$$w_j = -N_{j_{\min}} + 2, \quad j \neq i. \tag{21}$$

The virtual source node i is assigned a weight $w_i = 1$. This scheme gives the priority to actual sink nodes to be selected, followed by other virtual sinks, and finally the virtual source node in consideration. The algorithm randomly selects a node $j \in \mathcal{C}_i(k)$ to be the destination of the additional flow when available:

$$\begin{aligned} &\text{Select } j \in \mathcal{C}_i(k) \text{ with probability } \frac{w_j}{W} \\ &F_{i,j}(k+1) = F_{i,j}(k) + dA(i, r) \end{aligned} \tag{22}$$

where W is the sum of all weights $W = \sum_j w_j$ including w_i, r is the column in $\mathbf{A}$ corresponding to $\langle i, j\rangle$, and d is the discretizing parameter satisfying Assumption 1. The desired flow of the node i is decremented by d to indicate the need of more flow to pass into the node, and the destination of the extra flow is randomly selected from $\mathcal{C}_i(k)$. When the required flow arrives to the node i, its desired flow value is incremented by d to indicate that the flow demand is satisfied and passed to the selected destination through the link $\langle i, j\rangle$ by increasing its flow $F_{i,j}(k)$. If the selected destination node is i itself, then the flow in the network is not changed.

If the node i is an actual supply node that has not reached its maximum capacity, then the flow $F_{i,j}(k)$ is changed immediately according to (22). Otherwise, the desired net flow of the node $U_i(k)$ is decremented by the value d to convert

the node i into a virtual sink. Once additional flow is available the algorithm (22) is executed and the desired net flow of the node $U_i(k)$ is incremented by the value d to its original value and the node i is a virtual source again.

Finally, if the net flow of a node $N_i(k)$ is calculated to be less than its minimum demand capacity $N_{i_{\min}}$ and none of its neighbouring nodes are virtual sink nodes, then the excess inflow is not needed and is reduced uniformly in all the links carrying the flow into the node.

For a network where (4), (5) and Assumption 1 hold, the flow values $\mathbf{F}_{i,j}(k)$ in the links belong to the finite non-empty set of solutions that exists as stated by Lemma 1. The algorithm aims to remove the virtual sink nodes from the set $\mathcal{T}(k)$ by supplying the maximum flow from the source nodes to the sink nodes. Assuming that $\mathcal{C}_i(k)$ contains at least one virtual sink node $j \in \mathcal{T}$, the flow from i to j is increased such that the net flow in j becomes closer to its desired value:

$$N_j(k) \to U_j(k), \quad i \neq j \tag{23}$$

and the process continues until $N_j(k) = U_i(k)$ at which the node j is removed from the set $\mathcal{T}(k)$ and added to the set $\mathcal{S}(k)$.

Theorem 1. *Consider a complex network that satisfies the conditions of Lemma 1 and in which the flow behaves according to the algorithm (22). Then with probability 1 there exists a time $k_0 \geq 0$ such that the solution satisfies (11) for all $k > k_0$.*

Proof. The proposed algorithm defines an absorbing Markov chain which has a finite number of states since the network satisfies the conditions of Lemma 1. The absorbing Markov chain includes absorbing states that are impossible to leave. These absorbing states are categorized into three groups.

The first group of absorbing states is reached when the net flow values of all the sink nodes in the network reach their maximum capacities and no additional flow is required. The source nodes are supplying enough flow to the sink nodes and the links are able to carry the required flow. Therefore, the set of virtual sinks is empty and all the nodes are virtual sources

$$\mathcal{S}(k) = \mathcal{N}, \qquad \mathcal{T}(k) = \varnothing. \tag{24}$$

In this case all the virtual sources do not have candidate destinations in their neighbourhoods

$$\mathcal{C}_i(k) = \{i\}, \quad \forall i \in \mathcal{S}(k) \tag{25}$$

and the probability for each node i to maintain its current net flow is $\frac{w_i}{w_i} = \frac{1}{1} = 1$.

The second group of absorbing states is reached when the source nodes reach their maximum capacities and the supplied flow is not enough to meet the demand of the sink nodes in the network. The set of virtual sources is empty and all the nodes are virtual sinks

$$\mathcal{S}(k) = \varnothing, \qquad \mathcal{T}(k) = \mathcal{N}. \tag{26}$$

In this case there are no virtual source nodes to drive the algorithm, and the flow reaches a steady state.

The third group of absorbing states is reached when all the remaining source nodes have no paths to connect them to a sink node due to the limit (4) and delivering additional flow to the sink nodes is not possible. In this case all the virtual sources do not have candidate destinations in their neighbourhoods

$$\mathcal{C}_i(k) = \{i\}, \quad \forall i \in \mathcal{S}(k) \tag{27}$$

and the probability for each node i to maintain its current net flow is $\frac{w_i}{w_i} = \frac{1}{1} = 1$.

In any of the previous states it is impossible to increase the flow in the network and hence the value of the existing flow delivered to the sink nodes is maximum and the solution satisfies (11). □

4. Simulation results

Four simulation tests were executed to demonstrate the performance of the proposed algorithm when applied to the general problem and its two special cases as described in Section 2. In the first simulation a network is started from zero initial conditions where the flow in all the links is set to zero and the algorithm is allowed to increase the flow in the network until reaching steady state. Once at steady state, a node in the network is disconnected along with the links attached to it and the algorithm is used to redistribute the flow in the network to maximize the delivered flow to the sink nodes and in the same time avoid a cascaded failure in the network. In the second simulation the flow capacity limits on the source and sink nodes are removed, and the maximum possible flow in the network is found. In the third simulation the constant flow supply/demand operation of a network is examined, and in the final simulation the rate of convergence is investigated.

4.1. General problem test

The proposed algorithm was tested using the network shown in Fig. 1. The network contains eleven nodes of which three are source nodes, three are sink nodes, and the remaining five are intermediate nodes. The nodes are connected by seventeen links. The network and the maximum flow capacity of each link are shown in Fig. 1. The maximum flow supply and demand values of the nodes are also shown.

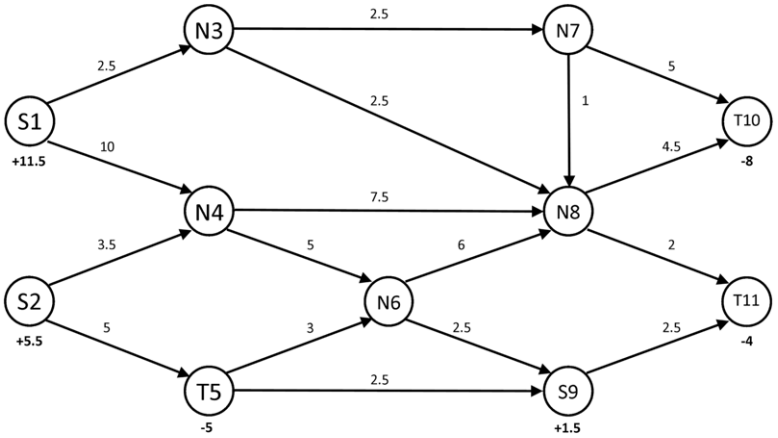

Fig. 1. Multi-source multi-sink network example.

Table 1
Steady-state node net flow: complete network.

Source node	Flow supply	Sink node	Flow demand
S1	11	T5	−5
S2	5	T10	−8
S9	1	T11	−4

Table 2
Steady-state node net flow: node *N*8 removed.

Source node	Flow supply	Sink node	Flow demand
S1	4.5	T5	−5
S2	4.5	T10	−2.5
S9	1	T11	−2.5

Table 3
Effect of the discretizing parameter.

Zero initial condition				
d	0.1	0.125	0.25	0.5
Number of iterations	370	298	156	85
Standard deviation	23.5	21.4	15.2	13.5
Recovery				
d	0.1	0.125	0.25	0.5
Number of iterations	20	16	8	3
Standard deviation	3.3	3.0	2.5	2.1

Selecting the discretizing parameter to equal $d = 0.5$ and starting from zero initial conditions, the algorithm converges to the steady-state values shown in Table 1. In this case all the demands of the sink nodes are met.

Once the network has reached steady state as in Table 1, the node *N*8 is assumed to be targeted by a random failure or an intentional attack and therefore disconnected from the network along with the links connecting it to the network. The proposed algorithm is used to redistribute the flow in the network to maximize the delivered flow to the sink nodes and in the same time avoid a cascaded failure in the network. The new steady-state values are shown in Table 2.

To show the effect of the discretizing parameter on the convergence speed of the algorithm, different values of d were tested and the number of iterations for each discretizing parameter value is presented in Table 3. The effect of d was tested for both the complete network starting from the zero initial condition and the modified network following the disconnection of node *N*8. The number of iterations is the average of 1000 runs of the algorithm for each case.

4.2. Maximum flow test

For the second test, the maximum supply and demand limits of the source and sink nodes are removed and the maximum flow that can be delivered to each sink node in the network considering the flow capacities of the links is found and presented

Table 4
Maximum possible flow.

Sink node	T5	T10	T11	Total
Maximum delivered flow (complete network)	−10.5	−8	−4.5	−23
Maximum delivered flow (modified network)	−10.5	−2.5	−2.5	−15.5

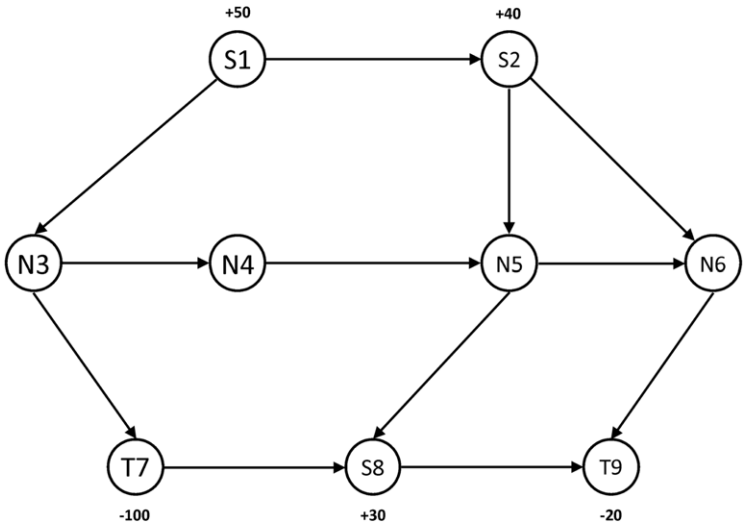

Fig. 2. Constant flow supply/demand network example.

in Table 4. As in the first test, the maximum possible flow that can be delivered to each sink node in the network is found for two cases: the complete network as in Fig. 1, and the modified network when the node $N8$ is removed.

4.3. Constant flow supply/demand test

The constant flow supply/demand simulation was performed using the network shown in Fig. 2. The network is composed of nine nodes of which three are source nodes, two are sink nodes, and the remaining four are intermediate nodes. The nodes are connected by twelve links with the maximum flow capacity of each link is assumed to be 100. The network is required to operate with constant flow supply from each source node and constant flow demand by each sink node with the values as shown in Fig. 2.

Starting from the zero initial condition where the flow in the network is set to zero, the proposed algorithm was allowed to control the flow in the network. Steady state was reached and the flow supply and demand of the source and sink nodes converged to the desired constant values. Then, the node $N5$ and the four links connected to it were disconnected from the network. As a result, the flow in the network was disrupted and the proposed algorithm was used to drive the network back to steady state. Noting that the flow capacity limits of the links are not exceeded, the flow supply and demand of the source and sink nodes converged back to the desired constant values. The dynamic behaviour of the source and sink nodes during the recovery process is shown in Fig. 3.

The random nature of the proposed algorithm means that every simulation run generates a slightly different dynamic response. This is demonstrated in Fig. 4 where the dynamic behaviour of the source node $S2$ obtained from four distinct runs of the simulation test are shown.

In complex networks where the flow in the links is not directly assigned (e.g. in power transmission networks) the transient dynamics may overload the links and trigger the cascaded failure [47]. The networks considered by this paper are constructed such that the flow in each link can be directly assigned by the two nodes it connects. This allows the proposed algorithm to prevent overloading the links when a disturbance affects the network.

The algorithm implementation gives priority to maintaining the flow values in the links below their maximum capacities. In fact, both the neighbourhood of a node $i\mathcal{Z}_i(k)$ and the neighbourhood of candidate destination nodes $\mathcal{C}_i(k)$ are defined such that the connecting links are not overloaded. When a failure or attack occurs, the lost flow delivered to the sink nodes is gradually recovered to ensure that the links are not overloaded. This behaviour can be seen in Figs. 3 and 4.

4.4. Rate of convergence vs number of nodes

Finding the rate of convergence for Markov chains is not an easy task. Researchers have investigated the convergence of particular chains that satisfy specific conditions (e.g. Refs. [48,49]). Although the proposed algorithm is based on absorbing

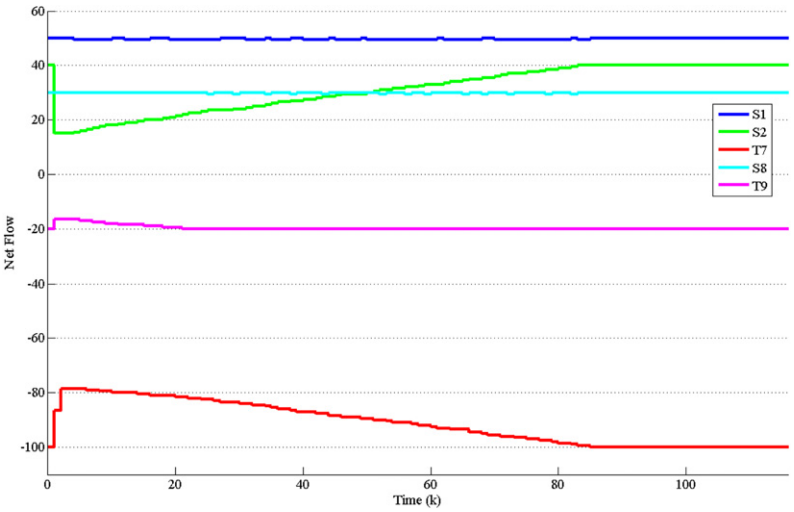

Fig. 3. Dynamic response of source and sink nodes.

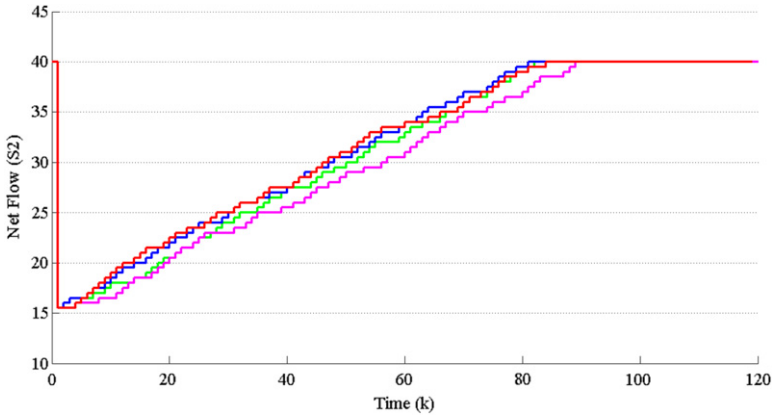

Fig. 4. Different dynamic responses of the source node *S2*.

Table 5
Rate of convergence vs number of nodes for the network in Fig. 1.

Initial nodes: S1, N4, N8 and T10								
Nodes	4 nodes	+S2	+N3	+T5	+N6	+N7	+S9	+T11
Iterations	37	33	33	35	37	62	59	85

Markov chains, in this paper simulations are used to investigate the convergence of the proposed algorithm for growing networks. The results show that the rate of convergence depends on several factors including the type of the added nodes to the network and how they are connected.

In the first test, the network in Fig. 1 is used starting with the four nodes S1, N4, N8 and T10 and the number of iterations for convergence is found. Then the other nodes in the network are added one node at a time and the number of iterations for convergence is found after each addition. The results are shown in Table 5 and the number of iterations are for convergence from zero initial conditions to steady-state.

The same test is repeated for the network in Fig. 2 and the results are shown in Table 6. An interesting observation is that the iterations increase with the addition of nodes then decrease again. When the network consists of the three nodes S1, N3 and T7, the source S1 supplies a flow of 50 through N3 which is less than the flow value of 100 demanded by the sink T7. The addition of the source S2 to the network almost doubles the number of iterations because the source S1 has to reach its maximum capacity of 50 then the source S2 starts to supply another 40 to the sink T7. The addition of the nodes N4, N5 and N6 has no effect on the rate of convergence, whereas the addition of the source S8 makes supplying flow to the sink T7 faster. Finally when the sink T9 is connected, it draws its flow in the same time as T7 and thus the number of iterations

Table 6
Rate of convergence vs. the number of nodes for the network in Fig. 2.

Initial nodes: S1, N3, and T7							
Nodes	3 nodes	+S2	+N4	+N5	+N6	+S8	+T9
Iterations	128	236	239	222	227	127	124

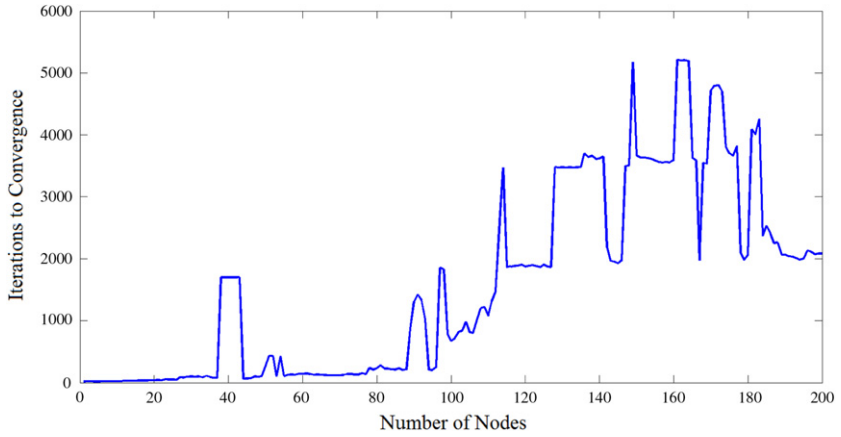

Fig. 5. Rate of convergence vs. the number of nodes for randomly generated networks.

is not affected. This shows the complexity of finding a relation between the number of nodes in a network and the rate of convergence.

In a third test a random network is generated by starting with two nodes connected by one link. Then, at each time, a new node is added to the network and connected by a link to a randomly selected existing node. The type and flow rating of the new node is also random whereas the flow capacities of the links are equal. The network is allowed to grow until it consists of 200 nodes and 201 links. The average number of iterations to converge from zero initial conditions to steady-state for six different randomly growing networks are shown in Fig. 5.

5. Conclusion

A distributed flow redistribution algorithm for complex networks after a failure/attack was proposed. In addition, the algorithm was used to find the maximum possible flow delivered to the sink nodes in the network, and also to maintain a constant supply/demand operation of the network. The convergence with probability 1 of this algorithm with maximum flow delivery was proved. Simulation results verified the performance of the algorithm.

Some improvements to this algorithm are to be investigated in future research. In particular, maximum flow algorithms applied to multi-source multi-sink networks may lead to "unfair" flows as noticed by Megiddo [50]. While the proposed algorithm is proven to converge to a solution at which the maximum possible flow is delivered from the source nodes to the sink nodes, additional constraints may be added to the decentralized algorithm to achieve a fair solution.

The work of Megiddo [50] applies only to centralized networks. Extending centralized algorithms to decentralized networks may not be the best idea. Future research will investigate the incorporation of rules within the proposed decentralized algorithms to achieve a consensus among the nodes of the network in terms of their assigned flow values compared to their flow capacities, hence providing the fairness in flow distribution. Flow consensus variables are to be introduced for each node in the network and simple local rules for updating the consensus variables of each node based on the average of its own value and the consensus variables' values of its neighbours. This method was used in numerous papers on control of multi-agent systems (see e.g. Refs. [51,52]) and has a potential to be applied to flow networks.

Acknowledgement

This work was supported by the Australian Research Council (ARC).

References

[1] L.R. Ford, D.R. Fulkerson, Maximum flow through a network, Canadian Journal of Math 8 (5) (1956) 399–404.
[2] E.A. Dinic, Algorithm for solution of a problem of maximum flow in a network with power estimation, Soviet Math Doklady 11 (8) (1970) 1277–1280.
[3] A.V. Karzanov, Determining the maximum flow in a network by the method of pre-flows, Soviet Math Doklady 15 (3) (1974) 434–437.
[4] D.D. Sleator, R.E. Tarjan, A data structure for dynamic trees, Journal of Computer and System Sciences 26 (3) (1983) 362–391.

[5] A.V. Goldberg, R.E. Tarjan, A new approach to the maximum flow problem, in: Proceedings of The Eighteenth Annual ACM Symposium on Theory of Computing, 1986, pp. 136–146.
[6] A.V. Goldberg, R.E. Tarjan, Finding minimum cost circulations by successive approximation, Mathematics of Operations Research 15 (3) (1990) 430–466.
[7] A.V. Goldberg, S. Rao, Beyond the flow decomposition barrier, Journal of the ACM 45 (5) (1998) 783–797.
[8] D.S. Hochbaum, The pseudoflow algorithm: a new algorithm for the maximum-flow problem, Operations Research 56 (4) (2008) 992–1009.
[9] G. Borradaile, P. Klein, An $O(n \log n)$ algorithm for maximum st-flow in a directed planar graph, Journal of the ACM 56 (2) (2009).
[10] S. Zhao, X. Xu, B. Hua, Y. Zhang, Contraction network for solving maximum flow problem, in: Proceedings of the ACM SIGKDD Workshop on Mining Data Semantics, 2012.
[11] R.K. Ahuja, T.L. Magnanti, J.B. Orlin, Network Flows: Theory, Algorithms, and Applications, Prentice Hall, 1993.
[12] G. Borradaile, P.N. Klein, S. Mozes, Y. Nussbaum, C. Wulff-Nilsen, Multiple-source multiple-sink maximum flow in directed planar graphs in near-linear time, in: Annual IEEE Symposium on Foundations of Computer Science, 2011.
[13] G.W. Flake, S. Lawrence, C.L. Giles, F.M. Coetzee, Self-organization and identification of web communities, Computer 35 (3) (2002) 66–70.
[14] A. Segall, Decentralized maximum-flow protocols, Networks 12 (1982) 213–230.
[15] J. Edmonds, R.M. Karp, Theoretical improvements in algorithmic efficiency for network flow problems, Journal of the Association for Computing Machinery 19 (2) (1972) 248–264.
[16] T.-Y. Cheung, Graph traversal techniques and the maximum flow problem in distributed computation, IEEE Transactions on Software Engineering SE-9 (4) (1983) 504–512.
[17] B. Awerbuch, Reducing complexities of the distributed max-flow and breadth-first-search algorithms by means of network synchronization, Networks 15 (1985) 425–437.
[18] Y. Shiloach, U. Vishkin, An $O(n^2 \log n)$ parallel max-flow algorithm, Journal of Algorithms 3 (2) (1982) 128–146.
[19] P.N. Klein, S. Mozes, Multiple-source single-sink maximum flow in directed planar graphs in O (diameter $n \log n$) time, in: Proceedings of the 12th Algorithms and Data Structures Symposium, 2011.
[20] J. Wang, L. Rong, L. Zhang, Z. Zhang, Attack vulnerability of scale-free networks due to cascading failures, Physica A 387 (2008) 6671–6678.
[21] Z.J. Bao, Y.J. Cao, L.J. Ding, G.Z. Wang, Comparison of cascading failures in small-world and scale-free networks subject to vertex and edge attacks, Physica A 388 (2009) 4491–4498.
[22] J.J. Wu, Z.Y. Gao, H.J. Sun, Effects of the cascading failures on scale-free traffic networks, Physica A 378 (2007) 505–511.
[23] J.-F. Zheng, Z.-Y. Gao, X.-M. Zhao, Modeling cascading failures in congested complex networks, Physica A 385 (2007) 700–706.
[24] J.J. Wu, H.J. Sun, Z.Y. Gao, Cascading failures on weighted urban traffic equilibrium networks, Physica A 386 (2007) 407–413.
[25] H.J. Sun, H. Zhao, J.J. Wu, A robust matching model of capacity to defense cascading failure on complex networks, Physica A 387 (2008) 6431–6435.
[26] D.Q. Wei, X.S. Luo, B. Zhang, Analysis of cascading failure in complex power networks under the load local preferential redistribution rule, Physica A 391 (2012) 2771–2777.
[27] S.V. Buldyrev, R. Parshani, G. Paul, H.E. Stanley, S. Havlin, Catastrophic cascade of failures in interdependent networks, Nature 464 (2010) 1025–1028.
[28] M. Vaiman, K. Bell, Y. Chen, B. Chowdhury, I. Dobson, P. Hines, M. Papic, S. Miller, P. Zhang, Risk assessment of cascading outages: Part I—overview of methodologies, in: IEEE Power and Energy Society General Meeting, 2011.
[29] M. Papic, K. Bell, Y. Chen, I. Dobson, L. Fonte, E. Haq, P. Hines, D. Kirschen, X. Luo, S. Miller, N. Samaan, M. Vaiman, M. Varghese, P. Zhang, Survey of tools for risk assessment of cascading outages, in: IEEE Power and Energy Society General Meeting, 2011.
[30] J. Asha, D. Newth, Optimizing complex networks for resilience against cascading failure, Physica A 380 (2007) 673–683.
[31] D.N. Kosterev, C.W. Taylor, W.A. Mittelstadt, Model validation for the August 10, 1996 wscc system outage, IEEE Transactions on Power Systems 14 (3) (1999) 967–979.
[32] U.S.–Canada Power System Outage Task Force, Final report on the August 14, 2003 blackout in the United States and Canada: causes and recommendations, 2004, URL https://reports.energy.gov/BlackoutFinal-Web.pdf.
[33] V. Rosato, L. Issacharoff, F. Tiriticco, S. Meloni, S. Porcellinis, R. Setola, Modelling interdependent infrastructures using interacting dynamical models, International Journal of Critical Infrastructures 4 (2008) 63–79.
[34] Union for the Co-Ordination of Transmission of Electricity, Final report on the disturbances of 4 November 2006, 2007, URL https://www.entsoe.eu/fileadmin/user_upload/_library/publications/ce/otherreports/Final-Report-20070130.pdf.
[35] B. Stott, E. Hobson, Power system security control calculations using linear programming, Part I, IEEE Transactions on Power Apparatus and Systems PAS-97 (5) (1978) 1713–1720.
[36] G. Chen, Z.Y. Dong, D.J. Hill, Y.S. Xue, Exploring reliable strategies for defending power systems against targeted attacks, IEEE Transactions On Power Systems 26 (3) (2011) 1000–1009.
[37] D. Bienstock, Optimal control of cascading power grid failures, in: IEEE Conference on Decision and Control and European Control Conference, 2011, pp. 2166–2173.
[38] J. Chen, J.S. Thorp, I. Dobson, Cascading dynamics and mitigation assessment in power system disturbances via a hidden failure model, Electrical Power and Energy Systems 27 (2005) 318–326.
[39] H. Ren, I. Dobson, B.A. Carreras, Long-term effect of the $n - 1$ criterion on cascading line outages in an evolving power transmission grid, IEEE Transactions on Power Systems 23 (3) (2008) 1217–1225.
[40] G. Chen, Z.Y. Dong, D.J. Hill, G.H. Zhang, K.Q. Hua, Attack structural vulnerability of power grids: a hybrid approach based on complex networks, Physica A 389 (2010) 595–603.
[41] A.S. Matveev, A.V. Savkin, Qualitative Theory of Hybrid Dynamical Systems, Birkhauser, Boston, 2000.
[42] A.V. Savkin, A.S. Matveev, Globally periodic behaviour of switched flow networks with a cyclic switching policy, Systems & Control Letters 38 (1999) 151–155.
[43] A.V. Savkin, A.S. Matveev, Cyclic linear differential automata: a simple class of hybrid dynamical systems, Automatica 36 (5) (2000) 727–734.
[44] M.E. Rusli, R. Harris, A. Punchihewa, Markov chain-based analytical model of opportunistic routing protocol for wireless sensor networks, in: IEEE Region 10 Conference, TENCON, 2010, pp. 257–262.
[45] S.A. Hassan, M.A. Ingram, A quasi-stationary Markov chain model of a cooperative multi-hop linear network, IEEE Transactions on Wireless Communications 10 (7) (2011) 2306–2315.
[46] A.V. Savkin, F. Javed, A.S. Matveev, Optimal distributed blanket coverage self-deployment of mobile wireless sensor networks, IEEE Communications Letters 16 (6) (2012) 949–951.
[47] I. Simonsen, L. Buzna, K. Peters, S. Bornholdt, D. Helbing, Transient dynamics increasing network vulnerability to cascading failures, Physical Review Letters 100 (218701) (2008) 1–4.
[48] J. Suzuki, A Markov chain analysis on simple genetic algorithms, IEEE Transactions On Systems, Man, And Cybernetics 25 (4) (1995) 655–659.
[49] S.F. Jarner, G.O. Roberts, Polynomial convergence rates of Markov chains, The Annals of Applied Probability 12 (1) (2002) 224–247.
[50] N. Megiddo, Optimal flows in networks with multiple sources and sinks, Mathematical Programming 7 (1) (1974) 97–107.
[51] A. Jadbabaie, J. Lin, A.S. Morse, Coordination of groups of mobile autonomous agents using nearest neighbor rules, IEEE Transactions on Automatic Control 48 (6) (2003) 988–1001.
[52] M. Cao, D.A. Spielman, A.S. Morse, A lower bound on convergence of a distributed network consensus algorithm, in: Proceedings of the 2005 IEEE Conference on Decision and Control, 2005.