Resilience Algorithms in Complex Networks

Fred S. Roberts[1][0000-1111-2222-3333]

[1] DIMACS Center, Rutgers University, Piscataway, NJ 08854, USA

Abstract. Today's complex systems are vulnerable to disruptions from natural, deliberate, and accidental events. The paper explores resilience of systems under disruptions due to disease events, fires, outages in the power grid, and damage to critical infrastructure, and describes algorithms for responding to a disruption that minimize the departure from a previous state.

Keywords: Resilience, disease spread, power grid, infrastructure repair

1 Resilience

Today's society has become dependent on complex systems, enabled by increased digitization of our world and the increasingly ubiquitous nature of intelligent machines, that have had a great impact on virtually all facets of our lives and our societies: enabling our financial transactions, running our power grid, underpinning our transportation systems, empowering our health care, supporting the rapid delivery of supplies and materials. Yet these changes have made us vulnerable to natural disasters, deliberate attacks, just plain errors. In recent years, ***"resilience"*** of complex natural and social systems has become a major area of emphasis; resilience in response to hurricanes, disease events, floods, earthquakes, cyber attacks, …

The general concept of resilience is the ability of a system to recover from disasters or attacks and avoid catastrophic collapse. We can think of a system existing within a "normal healthy range" as measured under some parameter or parameters. In a resilient system, values will return to the normal healthy range after a disruption. Or they might establish a new healthy range – one that is not that far from the previous one. See Figure 1.

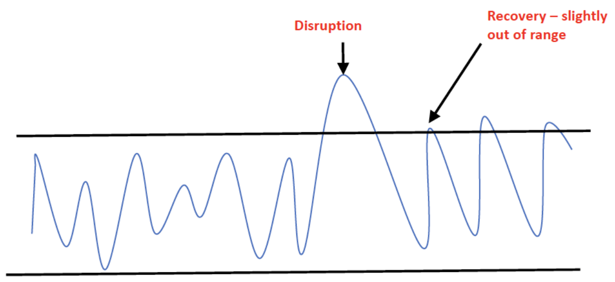


Fig. 1: A resilient system returns to its normal healthy range or one close to it.

There are many parameters that measure a "healthy" system. Some will get back into their normal healthy range faster than others. Do we ask that the longest time to return to this range be small? Or that the average time to return to this range be small? There are many such questions that we need to answer before being able to define resilience precisely.

One approach to resilience is to develop algorithms for responding to a disruption that will minimize the departure from the previous state when things settle down. Another is to design systems that can bounce back from disruptions quickly. In this paper we will emphasize the former. We will illustrate this with four examples built around models using graphs and networks and will present "toy" examples that illustrate the main points about how algorithms might help in creating more resilient systems: containing the spread of disease, mitigating the effect of fires, controlling cascading outages in the power grid, and reopening a damaged infrastructure network.

2 Spread and Control of Disease

The spread of the new Coronavirus COVID-19 is the latest (and most drastic) example of a newly emerging disease that threatens not only lives but our economy and our social systems. SARS, MERS, Ebola, and Zika are other recent examples.
Modern transportation systems allow for rapid spread of diseases. Diseases are spread through social networks. "*Contact tracing*" is an important part of any strategy to combat outbreaks of infectious diseases, whether naturally occurring or resulting from bioterrorist attacks. We will illustrate the ideas with some fairly simple "toy" models that will illustrate concepts of resilience.

We model a social network as an undirected graph. The vertices of the graph represent people and there is an edge between two people if they are in contact. We imagine that the vertices are in different states at different times. In the simplest "toy" model, there are two states, a susceptible state that we will represent with a green vertex and an infected state that we will represent with a red vertex. Times are discrete: t = 0, 1, 2, … Let $s_i(t)$ give the state of vertex i at time t. This is sometimes called an SI model. More complex models might include susceptible S, exposed E, infected I, recovered R, and possibly split by age group, sex, etc.: SI, SEI, SEIR models, ...

We will consider a very simple process called an *Irreversible k-Threshold Process* in which an individual changes their state from green to red at time t+1 if at least k of their neighbors are in state red at time t. Moreover, an individual never leaves state red. The disease interpretation is that an individual is infected if sufficiently many of their neighbors are infected. In the special case k = 1, an individual is infected if any of their neighbors is infected. Figure 2 describes an irreversible 2-threshold process. It shows how the disease spreads from two infected vertices at time 0 to four at time 1, because each of the two new infected vertices had two infected neighbors. After that, no more spread can occur since the remaining uninfected vertices have only one infected neighbor. Figure 3 describes an irreversible 3-threshold process. Two of the green vertices turn red at time 1 since they have three red neighbors at time 0, but the third does not. It turns red at time 2 because at time 1 it now has three red neighbors.

A great deal of attention has been paid to a saturation problem, the *Attacker's Problem*: Given a graph, what subsets S of the vertices should we plant a disease with so that ultimately the maximum number of people will get it? This has an economic interpretation: What set of people do we place a new product with to guarantee "saturation" of the product in the population?

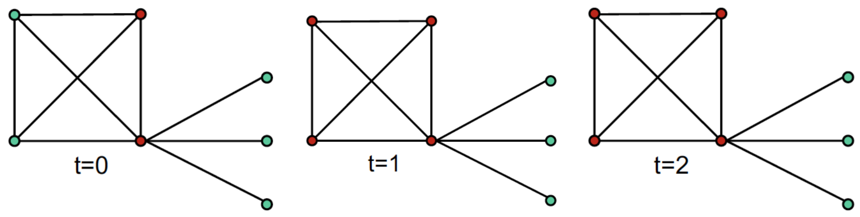


Fig. 2: An irreversible 2-threshold process.

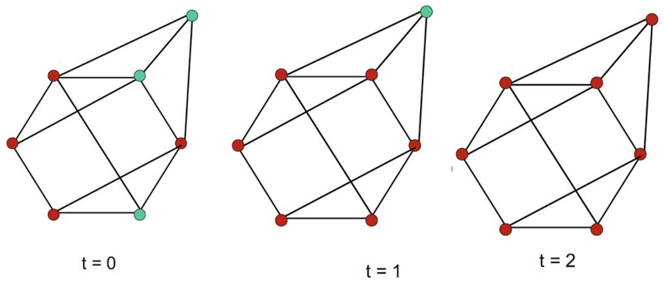


Fig. 3: An irreversible 3-threshold process.

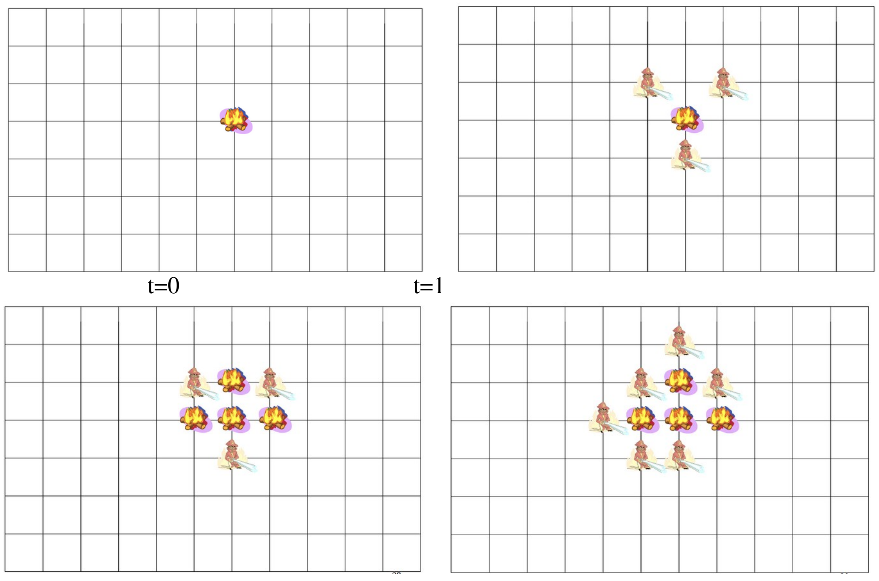
These problems are hard in a formal sense. Consider the problem called *IRREVERSIBLE k-CONVERSION SET*: Given a positive integer p and a graph G, does G have a set S of size at most p so that if all vertices of S are infected at the beginning, then all vertices will ultimately be infected? Dreyer and Roberts [1] showed that
IRREVERSIBLE k-CONVERSION SET is NP-complete for fixed $k > 2$. For many more results about irreversible k-conversion sets, see [1].

There are various complications that have been considered in this model. For example, one could take $k = 1$, but a person only gets infected with a certain probability. Or, a person is automatically cured after being in the infected state for d time periods. Or, a public health authority could have the ability to "vaccinate" a certain number of vertices, making them immune from infection. It is the vaccination strategy that relates to the resilience question and it is natural to ask: Can we find vaccination strategies (algorithms) that minimize the number of people who get sick, i.e., minimize departure from the normal. It is also natural to ask: Can we find vaccination strategies that minimize the amount of time before an outbreak is controlled, i.e., minimize time to return to normal. We turn to these questions next.

3 Vaccination Strategies: Vaccinations and Fighting Fires

Mathematical models are very helpful in comparing alternative vaccination strategies. The problem is especially interesting if we think of protecting against deliberate infection by a bioterrorist attacker but applies if we think of "nature" as the attacker. Another simple toy example considers an irreversible k-threshold model where a defender can vaccinate v people per time period but an attacker can only infect people at the beginning. We might ask: What vaccination strategy minimizes number of people infected? This is also sometimes called the *firefighter problem*. Here, we think of a forest or a city where a fire spreads to neighboring trees or buildings and firefighters can be placed at trees or buildings each time period to try to control the fire. We alternate fire spread which occurs according to an irreversible k-threshold model, and firefighter placement. It is usually assumed that $k = 1$ and we will assume this. The firefighter problem goes back to Hartnell [2]. A variation has the firefighter/vaccinator and fire/infector alternate turns, having v new firefighters to place/vaccination doses to give per time period and i new fires/new doses of pathogen per period. What is a good strategy for the firefighter/vaccinator?

Figure 4 illustrates the firefighter model with $k = 1$ (catch fire/get infected if one neighbor is on fire/infected) and $v = 3$ (you can place three firefighters/vaccinate three people each time period.) In successive grids, either the fire (disease) spreads or one places firefighters (doses of vaccine). It takes four time periods and 11 firefighters (vaccinations) to control the fire (disease outbreak).



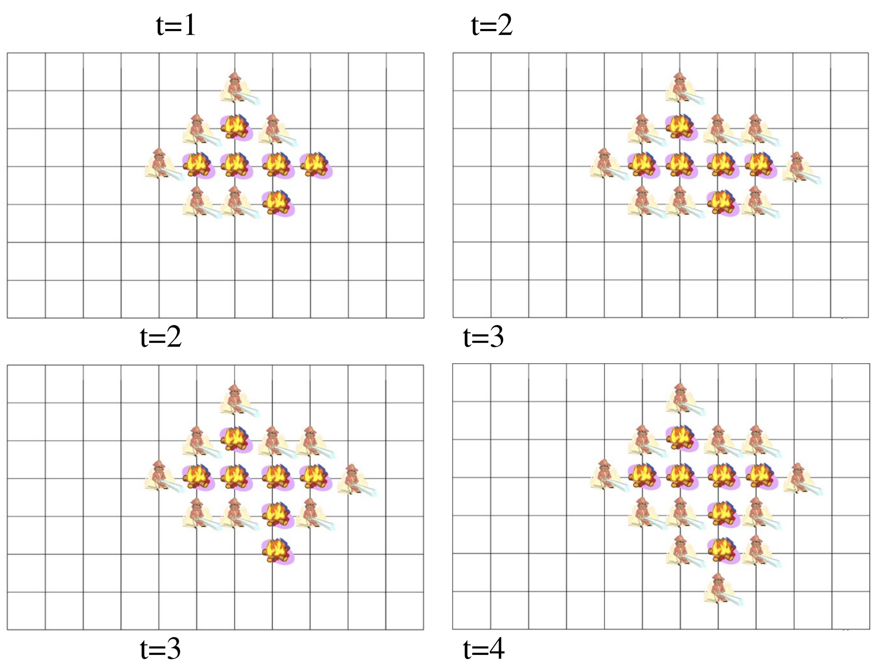


Fig. 4: The firefighter model with $k = 1$, $v = 3$. Thanks to Kah Loon Ng for help with this figure.

There are many resilience questions that can be asked. Can the fire (epidemic) be contained? How many time steps are required before fire (epidemic) is contained? How many firefighters (doses of vaccine) per time step are necessary? What fraction of all vertices will be saved (burnt/infected)? Does where the fire (disease) breaks out matter? What if the fire (disease) starts at more than one vertex?

Consider the case of an infinite d-dimensional grid. With $d = 2$ (similar to the grids in Figure 4), it is easy to see that a fire (disease outbreak) cannot be controlled with one firefighter (vaccine dose) per time period. With $d = 2$ and $v = 2$ (two firefighters or vaccinations per time period), one can show that a fire (disease) starting at one vertex can be controlled in eight time steps, with 18 trees burned (people infected). Develin and Hartke [3] showed that one cannot do better than 18 steps and Wang and Moeller [4] that one cannot contain the fire (disease outbreak) in less than eight steps. If $d \geq 3$, note that every vertex has 2d neighbors. Thus: $2d$-1 firefighters (vaccine doses) per time step are sufficient to contain any outbreak starting at a single vertex. Develin and Hartke [3] showed that if $d \geq 3$, $2d - 2$ firefighters (vaccine doses) per time step are not enough to contain an outbreak. However, with $2d - 1$ firefighters (doses) per time step containment can be attained in two time steps. This is just one example of a result, and there is an extensive literature on the firefighter problem by now.

There are many variants of the irreversible k-threshold model that would make the firefighter problem more realistic. A vertex could stay in the burning (infected state) for *T* time periods after entering it and then go back to the non-burning (uninfected) state – which is more appropriate for the disease application than the firefighting application. We could explore the strategy of bringing in a firefighter (vaccinating a person) once *k*-1 neighbors are burning (infected). We could consider the case of starting to burn (getting infected) with a certain probability if a neighbor is burning (infected). We could consider the case where firefighters are only successful at blocking a fire at a vertex (vaccines are only successful at protecting against a disease) with a certain probability. The amount of time you remain burning (infective) could also exhibit a probability distribution.

4 Cascading Outages in the Power Grid

Today's electric power systems operate under considerable uncertainty. Cascading failures can have dramatic consequences, including widespread blackouts. How resilient is the power grid? How can we design "control" procedures so that the power grid can quickly and efficiently respond to disturbances and quickly be restored to its healthy state? Grid disruptions can cascade so fast that a human being may not be able to react fast enough to prevent the cascading disaster, leading to a major blackout. We are dependent on rapid response through algorithms. We need fast, reliable algorithms to respond to a detected problem. The algorithm should not necessarily require human input, has to be able to handle multiple possible "solutions," and has to be able to understand what to do if all possible solutions are "bad."

One tool of interest is the cascade model of Dobson, et al. [5,6]. In this model, an initial "event" takes place, then demands and generator output levels are reconfigured, new power flows are instantiated, the next set of faults takes place according to some stochastic model, and the process repeats with reconfiguration of demands and generator output levels.

The power grid model is not the same as a disease-spread model. Energy flows from generators through power lines (edges in the power grid graph). Each edge has a maximum capacity. When a vertex (substation) or edge (transmission line) outage occurs, power reroutes according to physical laws (Kirchhoff's Law, Ohm's Law). Because of the rerouting, flows on parallel paths are increased. This could cause an overload in a distant transmission line. So failures can take place non-locally. Figure 5 demonstrates the model. In (b), there are a lightning strike and a fire, leading to destruction of the two lines (seen in (c)). This leads to increased flows on some lines (d) and loss of some more lines (e). Some demand is lost at yellow vertices in (f). More lines are lost (g) and more demand is lost (h).

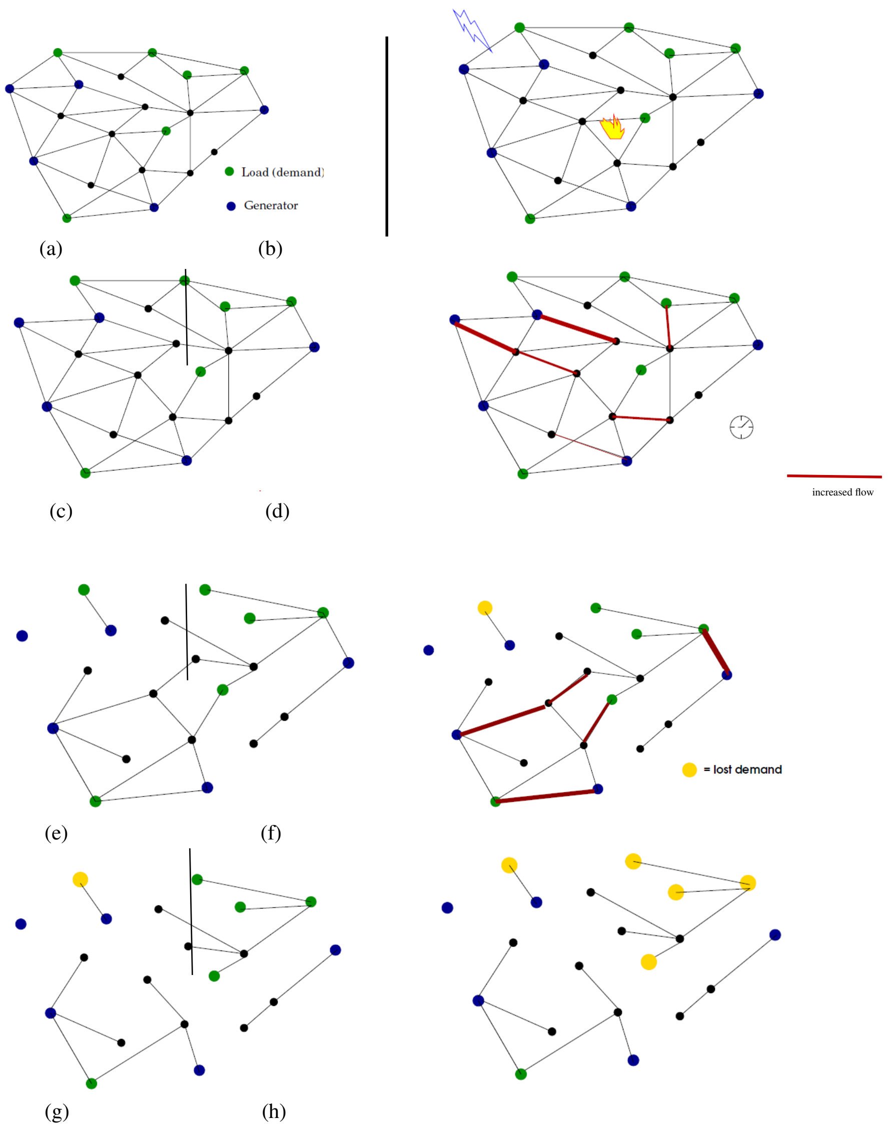


Fig. 5: The cascade model for power grids. Red edges show increased flow. Yellow nodes show lost demand. Thanks to Daniel Bienstock for the figures.

The cascade model shows what might happen if a system is left alone. However, it can be used to develop algorithms to “take control.” Instead of waiting for the next set of faults to take place according to some stochastic process, one can use the cascade model to learn how to take measurements and apply control to shed demand and to reconfigure generator outputs and get new power flows. One can also use the model to learn how best to create islands to protect part of the grid. Hopefully the islands are small and in the rest of the grid, supply is greater than demand. Figure 6 illustrates the point. By cutting some lines, one divides the grid into islands, hopefully a small one and a large one, where the supply is larger than the demand.

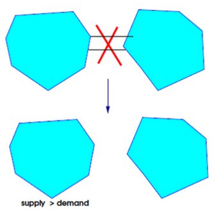


Fig. 6: Cutting out lines creates islands and in the rest of the grid (left-most part), supply is greater than demand. Thanks to Daniel Bienstock for the figures.

5 Repairing a Damaged Infrastructure Network

Critical infrastructure systems include transportation systems, telecom, water supply systems, wastewater systems, electric power systems, etc. After a disruption, a system begins to restore service until returning to a performance level *at or below* the level before the disruption. A series of models that allow us to reason about infrastructure resilience under disruptions were developed by Sharkey and Pinkley [7]. We describe their ideas in this section. In these models, service is described by flows in networks.

In contrast to the undirected graphs we have worked with so far, in the Sharkey-Pinkley models, a network now has vertices and directed edges (called *arcs*). A flow can only go from vertex i to vertex j along an arc directed from i to j. Vertices represent components that generate services (supply vertices), locations where one alters the routes of the services (transshipment vertices), or components that consume services (demand vertices). Arcs move the services from one vertex to another.

Consider for example a water supply system. The supply vertices correspond to water companies, the transshipment vertices to substations, and the demand vertices to households, factories, hospitals, malls, etc. Pipes are the arcs, and water is the flow. Meeting as much demand as possible is modeled as the classical maximum flow problem – both before and after a disruption.

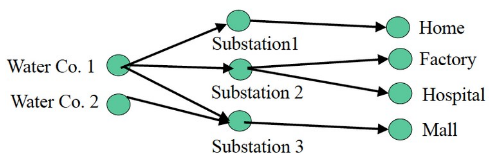


Fig. 7: A water supply system with two supply vertices, three transshipment vertices, and four demand vertices.

Consider a network $G = (V,A)$ where V is a set of vertices and A a set of arcs. The arc i to j has a *capacity* u_{ij}. In the simplest case, we have one supply vertex s and one demand vertex t. There is a *supply* $A(s)$ at s and a *demand* $B(t)$ at t. We seek to assign a *flow* x_{ij} to the arc from i to j. The flow along that arc must be at most the capacity:

$$x_{ij} \leq u_{ij}.$$

The classic maximum flow problem assumes that we have *flow conservation*: The sum of flows on arcs into a vertex equals the sum of flows out of the vertex. If $A(i) = \{j: (i,j) \in A\}$, then this says:

$$\sum_{j \in A(i)} x_{ij} = \sum_{j:i \in A(j)} x_{ji}$$

We also assume that the total flow out of s cannot exceed the supply $A(s)$ and the total flow into t cannot exceed the demand $B(t)$. We seek to maximize the total flow that reaches t. Thus, the *Maximum Flow Problem* seeks to determine the largest amount of flow that can reach t while keeping the flow on each arc at most the capacity, not exceeding total supply and demand, and satisfying the flow conservation requirement at each vertex. The famous augmenting path algorithm (Ford-Fulkerson Algorithm) finds the maximum flow.

Note that the maximum flow problem is a simplification. It assumes that there are no other constraints on flow. This might apply to supply chain networks, for example when physical goods move through intermediate warehouses and distribution centers. For more complicated infrastructure, there are things like physical laws offering additional constraints. For example, there are Kirchhoff's and Ohm's Laws for power grid networks. In water distribution networks, there are constraints involving the relation between flow of water and pressure.

Figure 8 shows a network with capacities on arcs and supply and demand at the supply and demand vertices and Figure 9 shows the maximum flow in the network. The flow is maximum because the flow from x to y is at most 2, so s to x is at most 2, so out of s is at most 40.

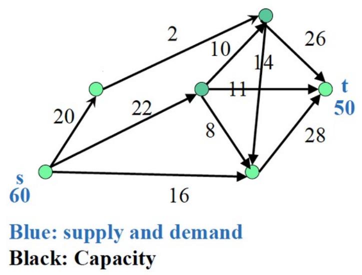


Fig. 8: A network with capacities on arcs and supply and demand at supply and demand vertices.

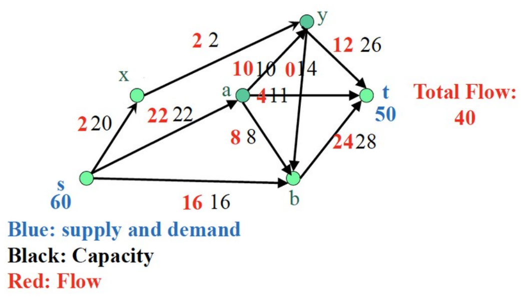


Fig. 9: The flow in this network is maximum.

If some of the arcs in a network are destroyed, we might ask in what order we should reopen them. There are various possible objectives. One goal might be to get closest to the original maximum flow as early as possible. Consider the example of Figure 10 where a lightning storm has taken out two arcs of the network of Figure 9. Assume that the arc *b* to *t* can be repaired fully but the arc *s* to *a* can only be repaired to a lower capacity of 15, as shown in Figure 11. That figure also shows the maximum flow if arc *s* to *a* is fixed first and arc *b* to *t* is not yet repaired so has no capacity. The maximum flow is 17. Figure 12 shows the maximum flow if arc *b* to *t* is fixed first and arc *s* to *a* is not yet repaired so has no capacity. That flow is 18, which is better, so to obtain the best flow after one repair we would repair the arc *b* to *t* first. Figure 13 shows the maximum flow once both arcs are repaired, but arc s to a only to reduced capacity. We only regained a reduced max flow of 33 so didn't fully restore the flow of 40.

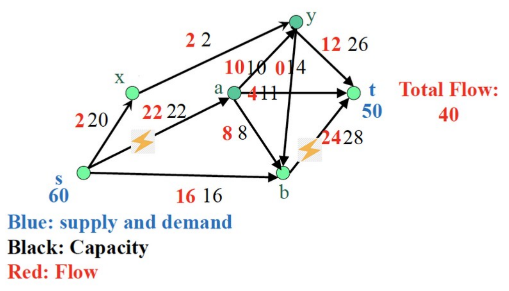


Fig. 10: A lightning storm takes our arcs *s* to *a* and *b* to *t*.

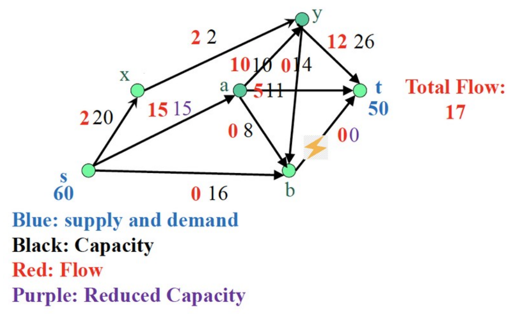


Fig. 11: Maximum flow if arc *s* to *a* is repaired to reduced capacity of 15 while arc *b* to *t* is not yet repaired.

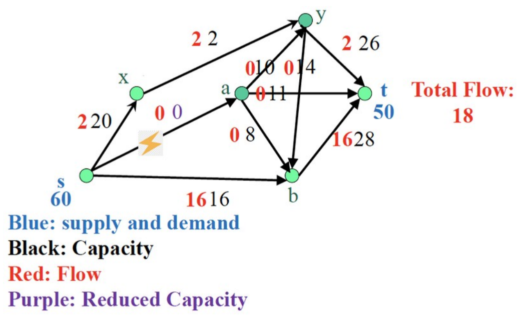


Fig. 12: Maximum flow if arc *b* to *t* is repaired to original capacity while arc *s* to *a* is not yet repaired.

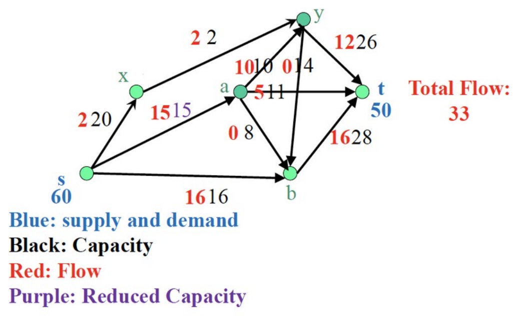


Fig. 13: Maximum flow if arcs *s* to *a* and *b* to t are repaired, the former to reduced capacity.

We made the simplifying assumption that there was one supply vertex and one demand vertex. In practice, there are many supply vertices s_1, s_2, …, and demand vertices t_1, t_2, …, with supply $A(s_i)$ at s_i and demand $B(t_i)$ at t_i. But we can reduce this to a single supply and demand vertex by adding a supply vertex S with supply $A(S) = \sum A(s_i)$ and an arc from S to each s_i with capacity $A(s_i)$ and a demand vertex T with demand $B(T) = \sum B(t_i)$ and an arc from each t_i to T with capacity $B(t_i)$.

As different components of a network are repaired (to the extent possible), the maximum flow increases. How far off it is from the original max flow when repairs are done is one metric for resilience. How long it takes to complete the repairs is another metric for resilience. We turn next to the repair process.

A different approach to reopening damaged components uses the theory of machine scheduling. After a disruption, repairs are made so services can be restored. Repairs use scarce *resources*: work crews, equipment. Let us make the simplifying assumption that one can only repair one *component at a time (one vertex or arc). We need a schedule for when a resource will be repairing* a component. In the scheduling literature, we talk about jobs on a set of machines, and processing them. *Jobs* here correspond to damaged components. *Machines* correspond to work crews. Each job (damaged component) k has a different level of *importance* w_k. Each job also has a *duration* p_k. In the scheduling literature, each job k is assigned to a machine (work crew) m_k. The jobs assigned to a machine (work crew) m are given an order. So the *completion time* C_k of job k is the sum of the durations of all jobs assigned to the machine (crew) m_k that precede job k plus the duration of job k.

There are various objectives for a good repair schedule. One might be to minimize the weighted average completion time over all jobs, with the weight measuring the importance of the job:

$$\min \sum_k w_k C_k$$

This is sometimes called the *restoration performance*. If there is just one work crew, a greedy algorithm minimizes this: Repair component k in non-increasing order of the ratio w_k/p_k. A similar algorithm works if there many machines but each has the same processing time for repairing a given component. However, in general, most such scheduling problems are hard, NP-hard. See [7] for more details.

The problem gets even harder if there are multiple interdependent infrastructures. In a complex city, there are many such infrastructures and they have interdependencies. For example, as observed in [7], a subway (transportation infrastructure) needs power (electrical infrastructure) before it can be reopened. A hospital needs both power and water before it can be reopened. One can approach the multiple infrastructure repair problem by studying a collection of networks, one for each infrastructure. A given infrastructure cannot operate until there is sufficient level of service (flow) on certain specific vertices in other infrastructures. Scheduling repair of different infrastructures will therefore depend on these interdependencies. There is a considerable literature on this topic. Another complication: There could be interdependencies among repair jobs, sometimes in different infrastructures. Scheduling repair of different infrastructures will therefore depend on these interdependencies. Another complication: interdependencies among repair jobs – sometimes in different infrastructures. To give an example from [7], note that to reopen subway lines, you need to repair a line. Once you repair the lines, you need to run a test train on the line to check for safety and quality of the repair. But power to the line must be restored before you can run a test train. To give another example from [7], suppose that trees bring power lines down on a road. First one needs to do a safety inspection to make sure it's safe to enter the road. Then one must clear debris from the road. Then one can repair downed power lines. There is a considerable literature on this and related topics. See [7] for a discussion.

6 Closing Comments

We have presented several simple examples of how to generate responses to disruptive events. Even these simple examples lead to problems that are "hard" in a precise sense.

Another approach is to study ways to design graphs or networks so as to make them more resilient in case of disruption. That is a topic for another paper.

References

1. Dreyer, P.A. Jr., Roberts, F.S.: Irreversible *k*-threshold processes: Graph-theoretical threshold models of the spread of disease and of opinion. Discrete Applied Mathematics 157, 1615-1627 (2009).
2. Hartnell, B.: Firefighter! an application of domination, Presentation, in: 20th Conference on Numerical Mathematics and Computing, University of Manitoba, Winnipeg, Canada, September 1995.
3. Develin, M., Hartke, S.G., Fire containment in grids of dimension three and higher. Discrete Applied Mathematics 155, 2257-2268 (2007).
4. Wang, P., Moeller, S.A.: Fire control on graphs. J. Combin. Math. Combin. Comput. 41, 19–34 (2002).
5. Dobson, I., Carreras, B.A., Lynch, V.E., Newman, D.E.: Complex systems analysis of series of blackouts: Cascading failure, critical points, self-organization. Chaos 17, 026103 (2007); https://doi.org/10.1063/1.2737822
6. Dobson, I., Carreras, B.A., Newman, D.E.: A loading-dependent model of probabilistic cascading failure. Probab. Engrg. Inform. Sci. 19(1), 15–32 (2005).
7. Sharkey, T.C., Pinkley, S.G.N.: Quantitative models for infrastructure restoration after extreme events: Network optimization meets scheduling. In Kaper, H.G., Roberts, F.S. (eds.),
   Mathematics of Planet Earth: Protecting Our Planet, Learning from the Past, Safeguarding for the Future, pp. 313-336, Springer, Cham (2019).

Acknowledgement: The author thanks the National Science Foundation for support under grant CCF-1934924 to Rutgers University.