Proceedings of the 2010 Industrial Engineering Research Conference
A. Johnson and J. Miller, eds.

Restoring Infrastructure Systems: An Integrated Network Flow and Scheduling Problem

Abstract ID: 263

Sarah G. Nurre and Thomas C. Sharkey
Department of Decision Sciences and Engineering Systems
Rensselaer Polytechnic Institute, Troy, NY 12180, USA

Abstract

We propose a new class of integrated network flow and scheduling problems that can be used to model important applications in restoring the services provided by infrastructure systems after disruptions within the system. This class of problems involves determining a set of arcs to install into a network and scheduling these arcs on a series of work groups in order to maximize the weighted cumulative flow in the network over a set of time periods. We provide the complexity of this class of problems, discuss greedy heuristics for it, and test our heuristics on realistic data representing infrastructure systems in lower Manhattan.

Keywords

Maximum Flows, Parallel Machine Scheduling, Infrastructure Systems, Extreme Events.

1 Introduction

The restoration of services provided by civil infrastructure systems after a disruption is an extremely important problem faced by the managers of the systems. This is especially true when the disruption is caused by an extreme event since society relies on the infrastructure systems to provide services essential to the recovery process. Therefore, models and tools that can serve as decision aids for the managers in restoring services will be quite valuable to both the managers and society. In this paper, we discuss a new class of optimization problems that can serve as such a decision aid. This class of optimization problems integrates network design, network flow, and scheduling problems in order to model the issues faced in restoring the services provided by the infrastructure systems. Further, we apply our models and algorithms to realistic data representing infrastructure systems of lower Manhattan in New York City and demonstrate that they are capable of serving as decision aids to the managers of these systems.

The new class of problems considered in this work involves selecting a set of arcs to install into an existing network and then scheduling these arcs on a series of parallel work groups. The decisions associated with the selection of arcs can be viewed as network design decisions. In contrast to previous network design problems, we will be interested in evaluating the performance of the network (the existing network plus the completed arcs) at intermediate points in time rather than simply the end result of the design. This means that the scheduling decisions associated with the arcs play an important role in the problem. The performance of the network will be evaluated by determining the amount of flow that can be delivered from the source node(s) to the demand node(s). As we will discuss in Section 2, this problem can be viewed as determining the maximum flow from a source node to a sink node in the network. Therefore, the objective function of our new integrated network flow and scheduling problem is to maximize the cumulative amount of (weighted) flow arriving at our sink node over the intermediate time points in the problem.

The motivating applications of this class of problems come from the restoration of services provided by civil infrastructure systems after a disruption. Lee et al. [5] develop network flow models for infrastructure systems and discuss that: (i) the performance of the system can be modeled as flows in the network and (ii) disruptions can be modeled as the removal of arcs from the network. It is clear that after a disruption occurs within an infrastructure system, that the managers of the system must determine components to install or repair in order to restore service

to those affected by the disruption. This corresponds to installing arcs into the network. Further, the infrastructure system will be providing services during the restoration efforts and, therefore, the performance of the system (i.e., the amount of demand that it can meet) at intermediate points in time will have a significant impact on the success of the restoration efforts. This implies that we should be interested in the cumulative performance of the system over time as opposed to simply the 'end' performance.

There has been recent work on developing decision support technologies to aid in the restoration of infrastructure systems. Lee et al. [5] focus on the *network design* decisions, i.e., which arcs to install into the existing network, in order to balance the design costs and the operational costs of the resulting network (where unmet demand is charged penalty costs). Gong et al. [3] focus on determining the schedule to process arcs into the network and focus on determining the efficient frontier of the objectives associated with installation costs, makespan, and weighted tardiness of the arcs. However, they do not focus on the important fact that the completed arcs will interact with one another in terms of the operations of the network. Cavdaroglu et al. [2] develop a model the examines the tradeoffs between the network design costs and the operational costs of the network as the arcs are being processed. The main issue with this model is that unmet demand within the network is penalized and it is difficult to quantify appropriate levels of penalty costs. The work in this paper focuses solely on meeting the unmet demand as efficiently as possible and is more applicable to large-scale disruptions caused by extreme events than Cavdraoglu et al. [2]. Guha et al. [4] develop approximation algorithms for classes of problems relating to the recovery from power outages; however, these problems assume that the demand nodes simply need to be *connected* to the supply nodes in some manner and are thus not concerned with the operational decisions of the power network.

The remainder of this paper is organized as follows. Section 2 provides the formal mathematical description of our problem. We discuss that our problem, even for a single work group, is NP-hard in Section 3.1. We also provide a property of the optimal solution to the problem in Section 3.1. This property sets the foundation for a greedy heuristic which adapts a classic scheduling rule to our network-based problem, which is discussed in Section 3.2. We apply our greedy heuristic to data representing the infrastructure systems of lower Manhattan in Section 4 and show that it obtains high quality solutions to the problem in seconds. We then discuss the significant advantages of our approach over the commercial software package CPLEX.

2 Problem Statement

We will now describe the formal mathematical description of our problem. We are given an existing network, $G = (N,A)$, where N is the set of nodes in the network and A is the set of arcs in the network. We are also given a set of arcs, A', that correspond to the arcs that we can install into the network (an arc in this set could correspond to installing a temporary component into the infrastructure system or repairing a destroyed component in it). Each arc (i,j) in A or A' has a corresponding capacity, which we denote u_{ij}. Every arc $(i,j) \in A'$ has a corresponding processing time, p_{ij}, which is the amount of time required to complete the arc (i,j) on a work group. In terms of the scheduling environment, we assume that each work group $k = 1,\ldots,K$ is capable of processing every arc $(i,j) \in A'$ and that once arc (i,j) is started, it must be continually processed until it is completed (i.e., a *non-preemptive* setting).

The performance of the network, i.e., the set of existing arcs A and the set of completed arcs from A', will be evaluated at time points $t = 1,2,\ldots,T$. We note that it may not be possible for all arcs in A' to be completed prior to T. This situation would be common, for example, when our model is being applied to short-term recovery efforts after an extreme event and at the end of the horizon the managers will shift to long-term efforts to enhance the resiliency of the infrastructure system. At time point t, we are interested in maximizing the flow from the source node to the sink node where the amount of flow on arc (i,j) cannot exceed its capacity u_{ij}. We will denote this maximum flow amount at time period t as f_t. Note that, without loss of generality, this objective function captures the common situation where we have multiple supply nodes and demand nodes within the network and we are interested in determining the amount of demand that can be satisfied from the supply nodes. In this case, we would introduce a super source node with an arc from this to each supply node, whose capacity is equal to the amount of supply at the node, and super sink node with an arc from each demand node to the super sink node, whose capacity is equal to the amount of requested demand at the node.

We must determine a schedule of arcs from A' for each work group (which implicitly provides us with the network

design decisions) in order to maximize the cumulative performance of the network, i.e.,

$$\max \sum_{t=1}^{T} \omega_t f_t, \tag{1}$$

where ω_t provides the weight associated with time period t. We will refer to this problem as the Integrated Maximum Flow and Scheduling Problem (IMFSP).

3 Theoretical Results and Heuristics

3.1 Theoretical Results

We will first prove that the IMFSP is NP-hard even for a single work group. In particular, the scheduling problem $1||\sum_{j\in J} w_j U_j$ (see Pinedo [6]) can be reduced to the IMFSP. In this problem, each job $j \in J$ has a processing time, p_j, weight, w_j, and due date, d_j, associated with it. The variable U_j is an indicator variable that is equal to 1 if job j is completed later than its due date.

Theorem 1 *The IMFSP with a single work group is NP-hard.*

Proof The scheduling problem $1||\sum_{j\in J} w_j U_j$ is NP-hard even when every job $j \in J$ has the same due date, i.e., $d_j = D$ for all $j \in J$. We will create an instance of the IMFSP for any instance of this scheduling problem. In particular, we will define the existing network $G = (N, A)$ to have a source node s and sink node t and $A = \emptyset$. We will place an arc (s,t) into A' for each job $j \in J$ with capacity equal to w_j and processing time equal to p_j. We will include time periods $t = 1,\ldots,D$ where $\omega_t = 0$ for $t = 1,\ldots,D-1$ and $\omega_D = 1$.

This means that the objective function for any schedule will be equal to the maximum flow in the completed network at time D. It is easy to see that each arc that is completed before D will be fully utilized, i.e., the amount of flow on it will be equal to w_j. Therefore, if $J(D)$ represents the set of jobs completed at or before time D, then we have that

$$\sum_{t=1}^{D} \omega_t f_t = f_D = \sum_{j\in J(D)} w_j = \sum_{j\in J(D)} w_j(1-U_j). \tag{2}$$

This means that we are interested in maximizing the cumulative weights of the jobs completed prior to their due dates, which is equivalent to minimizing the cumulative weights of the jobs completed after their due dates. Therefore, the optimal schedule to this instance of the IMFSP is also optimal to the problem $1||\sum_{j\in J} w_j U_j$, which proves our desired result. We note that it is not difficult to extend this proof to networks in which there does not exist any parallel arcs. In particular, we create a network with nodes s and t and then a node for every job $j \in J$. The arc set A contains an arc from s to every node $j \in J$. The arc set A' contains an arc for every $j \in J$ with capacity equal to w_j and processing time equal to p_j. We then follow the same argument as above. □

Despite this negative result, we can still offer some characteristics of the optimal schedule to the IMFSP with a single work group. In particular, we can show that it is best to process all arcs in some *path* in order to increase the flow in the network. Formally, we will define δ_t as the change in flow from $t-1$ to t, i.e., $\delta_t = f_t - f_{t-1}$. We now focus on the arcs processed between two consecutive flow increases, i.e., time periods t, t' such that $\delta_t, \delta_{t'} > 0$ and $\delta_\tau = 0$ for $\tau = t+1, t+2, \ldots, t'-1$. It is clear that the last arc processed between t and t' will directly contribute to the flow increase or, otherwise, the flow increase would have occurred between $t+1$ and $t'-1$. It turns out that there exists an optimal schedule where all arcs processed between two consecutive flow increases contribute to the later flow increase.

Theorem 2 *For the IMFSP problem with a single work group, there exists an optimal schedule where the arcs processed between two consecutive flow increases contribute to the later flow increase.*

Proof Suppose that we are given an optimal schedule S and that the arcs processed between consecutive flow increases in t and t' do not all contribute to the flow increase $\delta_{t'}$. Let $S(t,t')$ denote the schedule of arcs processed between periods t and t'. Furthermore, suppose that arc (k,ℓ) is in this schedule but does not contribute to $\delta_{t'}$. This means that we can

still achieve $\delta_{t'}$ with just the schedule of arcs $S(t,t') \setminus (k,\ell)$. Therefore, we can achieve the flow increase $\delta_{t'}$ at time $t' - p_{k\ell}$. We will alter S to schedule S' by processing the schedule at $S(t,t') \setminus (k,\ell)$ starting at time t and then arc (k,ℓ). The maximum flow in time periods $1,\ldots,t$ and $t',\ldots,T$ will not change since the set of completed arcs in those time periods will not change. Furthermore, we will achieve a maximum flow of $f_t + \delta_{t'}$ by at least time period $t'-1$. Since the weights, ω_t, are non-negative, this means that the schedule S' has an objective function as large as the objective function of schedule S. □

The nature of the maximum flow problem suggests a property that is even stronger than Theorem 2. In particular, in order for the flow in the network to increase, we must install arcs that are along some *augmenting path* (see Ahuja et al. [1]) from the source node to sink node in the resulting network. In other words, if $x^*(t)$ is the optimal solution to the maximum flow problem in time period t and $G(x^*(t))$ is the residual network of the flow, we will install arcs whose inclusion into $G(x^*(t))$ results in an augmenting path from the source node to the sink node. Therefore, Theorem 2 proves that there exists an optimal schedule where we process arcs belonging to the same augmenting path between two consecutive flow increases. We will utilize this fact and previous 'greedy' heuristics for scheduling problems in order to develop a heuristic for the IMFSP.

3.2 The Greedy Heuristic

A classic scheduling rule (which can be proven optimal for certain classes of problems) is to schedule the job with the largest value of w_j/p_j, i.e., weight per unit processing time, whenever a machine becomes available. This rule is known as the weighted shortest processing time first (WSPT) rule (see, for example, Pinedo [6]). In order for this type of rule to be applicable and successful to the IMFSP, the definition of 'job', 'weight,' and 'processing time' needs to be more closely examined. In particular, we should not view the 'job' as a single arc since this may not lead to an increase in the flow. Therefore, we will view the 'job' as a path whose installation corresponds to an augmenting path in the network. We will define the 'weight' as the increase in the maximum flow by installing the path and the 'processing time' will be the cumulative processing times of the arcs that need to be installed in the path. This means that the weight associated with installing the path should be equal to its *residual capacity* in the network after it is installed. At time t, the arcs in the residual network, $G(x^*(t))$, will have a residual capacity of r_{ij} and, since they already appear in the residual network, a processing time of 0. The arcs that we have yet to install in the network (i.e., those in A' but not yet processed) will have a residual capacity equal to their capacity as well as their original processing time. We then define the residual capacity of path P as $r(P) = \min_{(i,j)\in P} r_{ij}$ and the processing time of a path as $p(P) = \sum_{(i,j)\in P} p_{ij}$. We are then interested in scheduling the uninstalled arcs in the path that is the optimal solution to the problem

$$\max_{P\in\Phi} \frac{r(P)}{p(P)} \tag{3}$$

where Φ is the set of all potential paths. We note that for any path such that $p(P) = 0$ (i.e., all arcs are installed in the network at time t), that $r(P) = 0$ since $x^*(t)$ is the maximum flow in the network and, therefore, this path will not be optimal to (3). Given the definition of the arcs in a path as a 'job,' we need to be careful on defining this heuristic to problems with more than a single work group. This is because we only need to determine the next job when a machine becomes available if we have completed all arcs in the previous job first. In other words, we can view the arcs that need to be processed as a queue and we will process the next arc in the queue when a work group becomes available. If no arcs are in the queue, then we determine the next path that will be processed.

It is not immediately clear on how to obtain the path P that is optimal to (3). It is more difficult to determine this path than it is to determine the next job according to the WSPT rule since we cannot decompose (3) by arcs. We will now discuss a combinatorial algorithm to determine the optimal solution to (3). The idea for the algorithm is motivated by the following observation: if we know that $r(P^*)$ is the numerator in the optimal solution to (3), then P^* is the path with the shortest processing time in the network where we only include arcs whose residual capacities are above $r(P^*)$. In other words, if we know the numerator, then we want to determine the smallest denominator possible. This immediately leads to an algorithm to solve (3): for each potential value of the numerator (i.e., the residual capacity of a path), we determine the shortest processing time path in the network containing only arcs whose residual capacities are above the numerator. The optimal solution is then the path obtained in this procedure that has the maximum ratio of residual capacity to processing time. We note that this procedure is easily adapted to situations where a constraint is placed on the length of the processing times of the path (for example, if we are in a single work group setting and at

time t, we do not want to select a path with a processing time greater than $T-t$).

In order to determine the complexity of this algorithm, we must examine the number of potential values of the numerator, i.e., the number of distinct values of the residual capacity of a path. We note that the residual capacity of a path is the minimum residual capacity of the arcs in the path, so there are most $O(|A|+|A'|)$ distinct values since there are at most $O(|A|+|A'|)$ distinct residual capacities of the arcs. This means that we can determine the next set of arcs to be processed by solving $O(|A|+|A'|)$ shortest path problems. However, it is not always necessary to solve a shortest path problem for every distinct numerator. If the shortest path in the network composed of arcs with residual capacities over r has a residual capacity of $r' > r$, it is not necessary to consider the values of the numerator in the interval (r, r'). This is because this path will remain the shortest path in the network as we increase the threshold of the numerator since we will not remove arcs in this path until the threshold increases above r' since the path has a residual capacity of r'.

4 Computational Results

We will test our greedy heuristic on a realistic data set representing the power infrastructure of lower Manhattan in New York City. Lee et al. [5] developed a realistic representation of the infrastructure systems of lower Manhattan through data and discussions obtained with the managers of these systems. They also created a realistic scenario of a large scale disruption to the infrastructure system. We will implement our heuristic on this network and examine its performance for different objective functions and sets of work groups.

We will also attempt to solve an integer programming formulation of the IMFSP with the commercial software package CPLEX 11.0. Cavdaroglu et al. [2] discuss various integer programming formulations of a problem related to IMFSP (where the objective function focuses on the design and the operational costs) and conclude that a formulation based on determining which arc a work group is installing on a particular day is most desirable. In this formulation, we define binary variables α_{ktij} to represent the decision of work group k working on arc (i,j) during time period t and also define binary variables β_{tij} to represent the availability of arc (i,j) during time period t. Finally, we define binary variables z_{kij} to represent the decision that arc (i,j) will be processed by work group k during the horizon of the problem. We then place the following constraints on these variables:

$$\beta_{tij} + \sum_{k=1}^{K} \alpha_{ktij} \leq \sum_{k=1}^{K} z_{kij} \quad \text{for } (i,j) \in A', t = 1,\ldots,T \tag{4}$$

$$\sum_{(i,j)\in A'} \alpha_{ktij} \leq 1 \quad \text{for } k = 1,\ldots,K, t = 1,\ldots,T \tag{5}$$

$$\sum_{t=1}^{T} \alpha_{ktij} = p_{ij} z_{kij} \quad \text{for } (i,j) \in A', k = 1,\ldots,K, t = 1,\ldots,T \tag{6}$$

$$\beta_{(t+1)ij} \geq \beta_{tij} \quad \text{for } (i,j) \in A', t = 1,\ldots,T-1 \tag{7}$$

$$\alpha_{k(t+1)ij} + \beta_{(t+1)ij} \geq \alpha_{ktij} + \beta_{tij} \quad \text{for } (i,j) \in A', k = 1,\ldots,K, t = 1,\ldots,T. \tag{8}$$

Constraints (4) ensure that arc (i,j) cannot be both available and worked on during the same time period. Constraints (5) ensure that only a single arc is worked on at a time by each work group. Constraints (6) ensure that we fully complete arc (i,j) on the work group to which it is assigned. Constraints (7) and constraints (8) help ensure that the arc is continually processed until it is completed. We note that we could place a constraint that exactly one z_{kij} variable is equal to one across the work groups but since the availability variable is binary (β_{tij}) there is no benefit in the objective function of assigning the arc to multiple work groups since each one would have to complete it. Finally, we can determine the maximum flow in each time period in a standard manner (see Ahuja et al. [1]) by defining flow variables x_{ijt} for each arc $(i,j) \in A \cup A'$ and each time period. The only noteworthy modification is that the capacity constraint of $(i,j) \in A'$ should be $x_{ijt} \leq u_{ij}\beta_{tij}$.

The data set for our computational testing has an existing network of $|N| = 1810$ nodes and $|A| = 3316$ arcs. There are a total of $|A'| = 695$ arcs that can be selected to be installed into the network. We have selected a horizon of $T = 60$ days, which is a typical amount of time for restoration activities to be performed after a large scale extreme event. As

Lee et al. [5] note, the processing times of the arcs in the network are expressed in *days* and are almost always integral. Therefore, we have focused on processing times that are integral. We have examined classes of problems with a single work group and three work groups (i.e., $K = 1$ and $K = 3$). We have also examined two classes of weights in the problem: (i) constant weights (i.e., $\omega_t = 1$ for $t = 1, \ldots, T$) and (ii) scaled weights (i.e., $\omega_t = t/T$ for $t = 1, \ldots, T$). For each class of problems, we examined 5 number of instances for the processing times.

		Greedy Heuristic		CPLEX 11.0		
K	Weights	Time (s)	Performance (%)	Time (s)	Performance(%)	Best 4 Performance (%)
1	Constant	12.20	1.81%	21600	97.22%	6.58%
1	Scaled	12.20	1.02%	21600	98.15%	7.01%
3	Constant	17.60	0.92%	21600	98.34%	7.05%
3	Scaled	18.00	0.16%	21600	98.30%	7.05%

Table 1: Computational results on the Manhattan data set.

In these tests, we have set a time limit of 6 hours (21600 seconds). The performance of CPLEX 11.0 was determined to be the optimality gap between the best integer solution obtained and the current upper bound at the stopping point. We calculated the performance of our greedy heuristic by comparing the solution obtained by it with the current upper bound of CPLEX 11.0 at its stopping point and determining the percentage it is within the upper bound. Therefore, the performance reported by our heuristic is an *upper bound* on its actual performance since the optimal integral solution will have an objective function less than or equal to the current upper bound. For each class, there was always one instance that performed poorly for CPLEX 11.0 with an optimality cap over 400%. To combat this outlier in the data, we have also calculated the Best 4 Performance, which is the average of the best 4 instances within each class. Finally, we note that we have let CPLEX 11.0 run for 48 hours on an instance with a single work group and constant penalties and it still had an optimality gap of 6.49%.

Table 1 demonstrates the power and robustness of our greedy heuristic. It returns schedules of very high-quality in mere seconds and therefore can be utilized in real-time planning of restoration activities for infrastructure systems. This will be quite valuable to decision-makers in these systems since it will be easy for them to examine potentially different scenarios and to see the benefits of obtaining work groups from other areas (which is quite common in recovering from hurricanes) to assist with the restoration activities. Therefore, the IMFSP has the potential to serve as a powerful decision support tool for these managers. In the future, it will be interesting to see if we can integrate the solutions returned by the heuristic into exact methods in order to determine the optimal solution to the problem.

References

[1] R.K. Ahuja, T.L. Magnanti, and J.B. Orlin. *Network flows: Theory, algorithms, and applications*. Prentice-Hall, Englewood Cliffs, New Jersey, 1993.

[2] B. Cavdaroglu, J.E. Mitchell, T.C. Sharkey, and W.A. Wallace. Integrating restoration and scheduling decisions for disrupted interdependent infrastructure systems. Technical report, Department of Decision Sciences and Engineering Systems, Rensselaer Polytechnic Insitute, 2009.

[3] J. Gong, E.E. Lee, J.E. Mitchell, and W.A. Wallace. Logic-based multi-objective optimization for restoration planning. In W. Chaovalitwongse, K.C. Furman, and P.M. Pardalos, editors, *Optimization and Logistics Challenges in the Enterprise*. 2009. In print.

[4] S. Guha, A. Moss, J.S. Naor, and B. Schieber. Efficient recovery from power outage. In *Proceedings of the thirty-first annual ACM symposium on Theory of Computing (STOC)*, 1999.

[5] E.E. Lee, J.E. Mitchell, and W.A. Wallace. Restoration of services in interdependent infrastructure systems: A network flows approach. *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, 37(6):1303–1317, 2007.

[6] M.L. Pinedo. *Scheduling: Theory, Algorithms, and Systems*. Springer, New York, New York, 2008.