Integrated Network Design and Scheduling Problems with Parallel Identical Machines: Complexity Results and Dispatching Rules

Sarah G. Nurre*[1] and Thomas C. Sharkey[2]

[1]Department of Operational Sciences, Air Force Institute of Technology, WPAFB, OH 45433.

[2]Department of Industrial and Systems Engineering, Rensselaer Polytechnic Institute, Troy, NY 12180.

Abstract

We consider the class of integrated network design and scheduling (INDS) problems that focus on selecting and scheduling operations that will change the characteristics of a network, while being specifically concerned with the performance of the network over time. Motivating applications of INDS problems include infrastructure restoration after an extreme event and building humanitarian logistics networks. We examine INDS problems under a parallel identical machine scheduling environment where the performance of the network is evaluated by solving classic network optimization problems. We prove that all considered INDS problems are *NP*-hard. We propose a novel heuristic dispatching rule algorithm framework that selects and schedules *sets* of arcs based on their interactions in the network. These interactions are measured by examining network optimality conditions. Computational testing of these dispatching rules on realistic data sets representing infrastructure networks of lower Manhattan, New York demonstrates that they arrive at near-optimal solutions in real-time.

Keywords: Network Design, Scheduling, Dispatching Rule, Complexity Results, IP Formulation, Infrastructure Restoration, Parallel Identical Machines

*Corresponding author, e-mail: Sarah.Nurre@afit.edu. The work of this author was supported by a Sandia National Laboratories and Rensselaer Polytechnic Institute (RPI) Excellence in Engineering Research Fellowship.

1 Introduction

Integrated network design and scheduling (INDS) problems are a class of problems that model the selection and scheduling of operations that will change the characteristics of a network. These operations could correspond to enhancing existing network components or installing new components. The integration of these two sets of decisions, network design and scheduling, allows our models to capture how the operations improve the performance of the network over time during their implementation.

Traditional network design problems look at selecting the desired characteristics (e.g., which components to install) of a network. A network design is evaluated by looking at the end performance of the network, i.e., how the network operates after the design is completed, without considering the intermediate network performance as the design is being implemented. INDS problems specifically consider the intermediate network performance by incorporating scheduling decisions required to implement the selected design. These INDS scheduling decisions correspond to traditional scheduling problems that seek to allocate resources over time to process a set of tasks.

INDS problems focus on the allocation of machines[1] to process the set of tasks required to achieve a selected network design. The focus of this paper will be on INDS problems in a parallel identical machine environment. A solution to these problems involves three main decisions: *selecting* which components to process to change the network, *assigning* the selected components to a machine, and *sequencing* the assigned tasks on each machine. We consider INDS problems with objectives that focus on the cumulative (intermediate) performance of the network over time or that focus on how quickly a desired level of network performance can be achieved.

There are a number of real-life situations that motivate the need for models that integrate network design and scheduling problems. We present applications in restoring a disrupted network, upgrading an existing network, and building a new network.

Our first application arises after an extreme event, such as a hurricane, has caused large-scale damage to an infrastructure network. Infrastructure managers must make a restoration plan in order to repair damaged components and re-establish distribution of the services provided by the network. Customers of the infrastructure network evaluate the success of a restoration plan by how quickly their services are restored. Therefore, the repair of damaged components must be scheduled to create an operational network and ensure a quick restoration of services to many customers. The infrastructure will be providing services as it is implementing its restoration plan and, therefore, infrastructure managers will be concerned with the level of satisfied demand in the network over time. An INDS model can help formulate a restoration plan that allocates infrastructure work crews to process selected repairs in the infrastructure network.

Another application area is the implementation of network upgrades in order to improve services

[1]Please note that while we use the term 'machine', in many applications, work groups, work crews, or other resources take the role of the machines. However, for consistency with the scheduling literature, we will use the term machine when examining the mathematics of the INDS problem.

provided by the network. One example of this is constructing a new smart grid design for the power infrastructure. Researchers have looked into how to transform the electrical power grid into a smart grid (see Farhangi [11], Momoh [25], Mahmood et al. [23], DeBlasio and Tom [9]), many of which include installations of solar (see Mulder et al. [27]), wind (see Glinkowski et al. [13]), and other environmentally friendly and sustainable technologies (see Clastres [7], Moslehi and Kumar [26], and Liserre et al. [22]). For large projects like these, resources (e.g., budget) often only allow a number of upgrades to take place during a certain time period (e.g., annually). It is, therefore, important to determine which upgrades will be completed in each phase of the project, which can be modeled with an INDS problem. Note that, in this application area, it is likely that all upgrades (tasks) in the problem will be processed (i.e., the end design is fixed); however, INDS problems can help model the order of the tasks in order to optimize the intermediate network performance. Another example of an application of upgrading a network occurs when a company operating a supply chain chooses to enhance existing network components, such as improving a machine or expanding inventory capacity at a facility.

Our third application of INDS problems involves creating and building a new network, which is often needed during humanitarian logistics activities. After an extreme event, many people do not have access to essential items such as water, food, and shelter. To remedy this situation, humanitarian organizations set up locations to distribute these essential items and to act as shelters. These sites should be located in order to provide the greatest benefit to the affected community. There are limited resources available to set up these locations; therefore, we must allocate the resources over time to a selected subset of the potential distribution and shelter sites. It is clear that the partially completed network will need to be operational and goods will be distributed even when all work is not fully completed on the selected logistics network.

The objective of an INDS problem is based on two factors: (i) how the performance of the network is evaluated and (ii) how we evaluate the scheduling decisions. The performance of the network is often dictated by the application of the INDS problem. For example, the type of infrastructure system being restored will dictate the most applicable mathematical model to apply to determine the performance of the network. A power network might focus on maximizing flow at each time period of a restoration plan, which is equivalent to maximizing met demand. An emergency network might focus on the shortest path between residents and a hospital. We will consider network performance metrics based on four classic network flow problems (see Ahuja et al. [2]). In terms of evaluating the scheduling decisions, we are interested in determining the cumulative performance of the network over time, which means our objective will include the network performance in each period. Alternatively, we may also be interested in determining the minimum amount of time required to achieve some set performance value (e.g., all demand is met).

INDS problems have not been widely examined, despite their important applications. The works of Guha et al. [15], Averbakh [4], and Averbakh and Pereira [5] consider INDS problems that are focused

on the *recovery time* of each node in the network, which is defined as the first time a (demand) node is connected to a 'source' (or supply) node. These works do not consider the capacity constraints within a network, which is important when capturing the network performance in certain applications. Further, the INDS problems considered in this paper can capture the number of recovered nodes in the network at any point in time, so the objectives can model recovery time-based objectives. Xu et al. [31] looks at restoring a power network after an earthquake by scheduling inspection, assessment, and repair operations. This work, again, focuses on the recovery time of each demand node and proposes a genetic algorithm for this problem.

There has been further research on INDS problems that is interested in the cumulative performance of the network over time. Ang [3] provides an integer programming formulation of a model to restore power, where the focus is on minimizing the total cost of power shed over time. Matisziw et al. [24] provide a multicriteria integer programming formulation of an INDS problem whose network performance is the tradeoff between the uncapacitated flow between pairs of origins and destinations and the cost of system use over the restoration period. The number of machines in their scheduling environment is not explicitly modeled, but instead a budget is placed on each time period that limits the number of arcs that can be installed in each time period. Parallel to this work, Elgindy et al. [10] has considered an INDS problem where one arc can be installed in every time period with a focus on minimizing the total length of the shortest path over all time periods. Therefore, this problem is a special case of problems considered in this paper. Elgindy et al. [10] provides a *NP*-hard proof of this problem and approximation algorithms. Cavdaroglu et al. [6] examines an INDS problem with a network performance that is measured as the flow through a set of interdependent networks. They provide an integer programming formulation and a heuristic for this problem that is specifically geared towards interdependent networks and is thus not easily generalized to other network settings. Nurre et al. [28] looks at INDS problems that focus on maximizing the cumulative (over time) flow in the network. They propose a dispatching rule for this INDS problem, which has set the basis for the general framework examined in this paper. However, Nurre et al. [28] only consider a limited scope, in both the set of instances and objectives, of computational tests for their dispatching rule.

The contributions of this work to the study of INDS problems include: (i) the examination of 12 INDS problems with different network performance metrics and scheduling objectives (which can model many of the previous objectives considered in related work), (ii) theoretical analysis of the complexity of INDS problems, including hardness of approximation results for some problems, (iii) the creation of a general dispatching rule framework, that integrates concepts from the distinct fields of network flows and scheduling, which can be customized to different INDS problems, and (iv) a thorough set of computational tests, on realistic infrastructure networks and many different damage scenarios, that demonstrate the robust performance of the general dispatching rule framework.

Note that other work has examined network design problems for scheduling routes and services in the resulting network (see Lederer and Nambimadom [19], Cranic [8], Lai and Lo [18], Agarwal

and Ergun [1], and Guihaire and Hao [16]). These problems are fundamentally different than INDS problems since INDS problems look at utilizing machines to schedule the processing of components to make a selected network design operational.

We motivate the need for heuristic algorithms to solve INDS problems by first examining the complexity of the problems. We prove that the INDS problems considered in this paper are *NP*-hard. We then propose novel heuristic dispatching rule algorithms (see Pinedo [29]) that utilize network optimality conditions (see Ahuja et al. [2]). Dispatching rules are commonly used in scheduling and decide which task should be selected for processing on the next available machine. Contrary to traditional scheduling problems, in an INDS problem, once a task is complete it becomes operational in the network and, therefore, interacts with other network components to improve the performance of the network.

Our dispatching rules focus on how a *set* of tasks interact within the operational network to improve its performance. A dispatching rule framework is presented that first calls a selection routine that selects a set of tasks to become operational in the network. A scheduling routine is then called to schedule these selected components. The selection routine is specifically concerned with selecting a set of tasks that optimizes the ratio of improvement in the network performance to the total processing times of the tasks in the set. This selection routine is customized based on the specific network performance metric and its optimality conditions (see Ahuja et al. [2]).

We test the effectiveness of our dispatching rules by determining a restoration plan for realistic data representing damaged infrastructures in lower Manhattan. We compare this plan to the optimal plan based on an IP formulation (see the Appendix) of the INDS problem. The results show that the dispatching rule can arrive at near-optimal solutions very quickly and performs well over the set of considered performance metrics and objective functions.

The paper proceeds as follows: Section 2 provides the formal definition of INDS problems, the performance metrics, and objective functions studied in this paper; Section 3 examines the complexity of INDS problems; Section 4 provides the INDS dispatching rule framework and customizes it for each performance metric; Section 5 presents the results of the computational tests; and we conclude in Section 6.

2 Problem Statement

The purpose of this section is to provide a formal problem definition of INDS problems. This includes specifying the network characteristics, network performance metrics, objective functions, and scheduling environment associated with INDS problems. We first describe the problem classes which are the focus of this paper and then 'frame' previous research papers in the context of these specifications.

2.1 INDS Problem Classes

Let $G_t = (N, A_t, A'_t)$ represent the network at time t, where N represents the set of nodes, A_t represents the set of operational arcs at time t, and A'_t represents the set of non-operational arcs at time t. The performance of the network, denoted by $P(G_t)$, is directly impacted by the set of operational arcs, A_t. The set of arcs $A_t \cup A'_t$ remains constant over time. An arc $(i,j) \in A'_t$ can become operational and transition to $A_{\bar{t}}$, for some $\bar{t} > t$ once it has been processed by a machine. The initial operational network is made up of nodes N, operational arcs A_0 at time 0, and non-operational arcs A'_0 at time 0.

The set of nodes is not indexed by t because we assume all processing in the network occurs on the arcs. This is without loss of generality since the potential to install a node in the network can be modeled as installing an arc using standard network expansion techniques (e.g., split the node into two nodes and an arc). Within the set of nodes N, we may have (depending on the network performance metric) a subset of supply nodes S and demand nodes D. Each supply node $i \in S$ has an associated supply level s_i, and each demand node $i \in D$ has an associated demand level d_i.

Each arc has a set of associated parameters. Arc $(i,j) \in A'_0$ has an associated (constant) processing time $p_{ij} > 0$, representing the length of time it needs to be assigned to a machine to become operational. Every arc $(i,j) \in A_0 \cup A'_0$ has an associated capacity u_{ij} and cost c_{ij}.

Machines must be allocated to a non-operational arc for it to become operational. We assume m parallel identical machines in a non-preemptive environment, commonly denoted Pm. Parallel identical machines means that all machines can complete arc $(i,j) \in A'_0$ in p_{ij} time periods. In terms of our motivating applications, the differences between the processing times of different machines would tend to be negligible compared to the scale of the processing times. We do note that our proposed approaches could be potentially modified to handle situations in which the processing times are machine dependent, which we discuss in Section 6. The non-preemptive environment signifies that once a machine starts to work on arc $(i,j) \in A'_0$ it must continue to work on arc (i,j) for p_{ij} consecutive time periods to complete arc (i,j). Further note that in a Pm setting, each machine can be working on up to one task at any point in time. In a traditional scheduling environment, jobs tend to arrive at machines. However, for INDS problems a machine might need to relocate to a specific portion of the network in order to process its next task. Our models do not specifically consider this routing/relocation time since for our motivating applications this time is typically on a much smaller scale than the processing times associated with the arcs and is, therefore, negligible.

We consider two objectives: the *Cumulative* and the *Makespan-Threshold* (which we denote as $C_{\max}$-*Threshold*) objectives. The *Cumulative* objective examines a set time horizon T and optimizes the cumulative weighted network performance over time, $\sum_{t=1}^{T} w_t P(G_t)$, where w_t is a weight associated with each time period. The term 'optimize' can mean either minimize or maximize. The $C_{\max}$-*Threshold* objective considers a desired performance value P and minimizes the time (or makespan) needed to reach or exceed this threshold performance value, i.e., $\min \bar{T}$ such that $P(G_{\bar{T}}) \geq P$ for

maximization problems ($P(G_{\bar{T}}) \leq P$ for minimization problems).

The performance $P(G_t)$ of the network at each time period is influenced by the set of operational components at time t. Hence, as more components become operational the performance of the network will improve. The manner in which we evaluate the performance will be based on four core network flow problems (see Ahuja et al. [2]). The performance metrics are as follows:

1. Maximum Flow (MF): maximize the flow from a supply node to a demand node while adhering to arc capacities and flow balance constraints. This includes, through standard network expansion techniques, problems with multiple supply nodes and/or demand nodes.

2. Minimum Cost Flow (MCF): minimize the cost of flow that satisfies demand at a set of demand nodes from a set of supply nodes while adhering to arc capacities and flow balance constraints.

3. Shortest Path

   - Single-source, single-destination (SP): minimize the length of a path from a single source to a single destination. This includes, through standard network expansion techniques, problems where we are interested in finding the shortest path length from one of a set of source nodes to one of a set of destination nodes.
   - Sum of Shortest Path Lengths to Multiple Destinations ($\sum SP$): minimize the sum of shortest path lengths from a single source to multiple destinations. This can also model problems where we are interested in summing the shortest path length from one of a set of multiple source nodes to each of the multiple destinations.
   - Shortest Path Centering amongst Multiple Destinations (SPC): minimize the maximum length of a shortest path from a source to one of a set of multiple destinations. In other words, the source node is best 'centered' amongst the destination nodes.[2] This can also capture problems where we are interested in minimizing the maximum length of a shortest path from some source node (across a set of potential sources) to some destination node (across a set of potential destinations).

4. Minimum Spanning Tree (MST): minimize the cost of a tree that spans all nodes.

These 6 performance metrics in combination with 2 objective functions results in 12 total problems considered in this work. For ease of notation we utilize Graham's triplet notation for scheduling problems (see Graham [14]) and present the problems in $\alpha \mid \beta \mid \gamma$ format where α represents the machine environment, β represents special processing characteristics (performance metric) or constraints, and γ represents the objective function. We consider the following problems in this paper:

[2]This performance metric is similar to the p-center problem that seeks to locate p facilities to minimize the maximum distance from each demand node to its closest facility. Note though, our INDS problems can model both constructing nodes and arcs in the network, where the number of arcs and nodes constructed is determined by specifics of the INDS problem objective, unlike the known number of p facilities.

- $Pm|MF|Cumulative$
- $Pm|MCF|Cumulative$
- $Pm|SP|Cumulative$
- $Pm|\sum SP|Cumulative$
- $Pm|SPC|Cumulative$
- $Pm|MST|Cumulative$
- $Pm|MF|C_{\max} - Threshold$
- $Pm|MCF|C_{\max} - Threshold$
- $Pm|SP|C_{\max} - Threshold$
- $Pm|\sum SP|C_{\max} - Threshold$
- $Pm|SPC|C_{\max} - Threshold$
- $Pm|MST|C_{\max} - Threshold$

2.2 Related Literature: Formal Problem Definitions and Approaches

The purpose of this section is to provide an overview, in the context of INDS problems, of previous related research. In particular, we highlight, for each related research paper, the type of INDS problem studied by discussing its network performance metrics, objective functions, and scheduling environments, as well as the approaches applied to the problem.

Many previous papers related to our work (Guha et al. [15], Xu et al. [31], Averbakh [4], and Averbakh and Pereira [5]) have focused on objectives related to the *recovery time* of a node, which is defined as the first time a node is connected to a source node (e.g., there is a path from the source node to that particular node). In particular, Guha et al. [15] and Xu et al. [31] focus on total weighted recovery time for a set of demand nodes. Averbakh [4] and Averbakh and Pereira [5] focus on the total recovery time (all nodes have weight of 1) of all nodes in a network. The recovery time of a node can be captured by classes of INDS problems, which we consider, namely, the INDS problem with the minimum cost flow performance metric[3] with a cumulative objective function. In particular, for the minimum cost flow performance metric, we set the supply of the source node to be equal to one less than the number of nodes in the network and every other node has a demand equal to one. We then consider all 'original' arcs in the recovery time problem (which includes both arcs in the network and those that can be constructed) to have zero cost and unlimited capacity. We then place 'dummy' arcs in the network from the source node to every other node with a capacity equal to 1 and a cost equal to 1. The minimum cost flow at time t in this modified network then provides the number of unconnected nodes in the original network since, if a node is connected to the source, a zero cost path exists in the modified network. Note that node i will 'impact' the objective function in every time period until its recovery time and, therefore, minimizing the total recovery times of the nodes is equivalent to the INDS problem in the modified network. Note, if we are interested in the weighted total recovery time (we multiply the recovery time of node i by some weight), we can set

[3]For recovery time objectives where all nodes have the same weight, we could also model the problem with a maximum flow performance metric by maximizing the flow from the source node to all other nodes where each node can absorb at most one unit of flow.

Paper	Scheduling Environment	Network Performance Metric and Other Considerations	Objective	Successful Approaches/Results
Guha et al. [15]	$p_j = 1$, Budget constraints in each time period	Weighted Recovery Time	Cumulative	IP Formulation LP Rounding Heuristic
Ang [3]	Rm for $m = 1, 2, 3$	Cost of Unmet Demand	Cumulative	IP Formulation
Xu et al. [31]	Fm	Average Recovery Time	Cumulative $C_{\max}$-Threshold	IP Formulation Genetic Algorithm
Matisziw et al. [24]	$p_j = 1$, Budget constraints in each time period	Flow between Origin/Destination Pairs Overall System Cost	Multi-objective: Cumulative and Cost	Multicriteria IP Formulation
Averbakh [4]	Qm	Recovery Time of Nodes in Path Network	Cumulative $C_{\max}$-Threshold	*NP*-hard Result Recursive Heuristic Algorithms
Averbakh and Pereira [5]	$m = 1$	Recovery Time of Nodes	Cumulative	*NP*-hard Result, IP Formulation, Branch+Bound, Network-based Heuristic
Cavdaroglu et al. [6]	Pm for $m = 1, 2, 3$	Unmet Demand in Interdependent Layered Network	Cumulative	IP Formulation Network Design Heuristic
Nurre et al. [28]	Pm for $m = 1, 2, 3$	Maximum Flow	Cumulative	IP Formulation Dispatching Rule
Elgindy et al. [10]	$m = 1$, $p_j = 1$	Shortest Path	Cumulative	*NP*-hard Result Network-based Greedy Approach
This Paper	Pm for $m = 1, 2, 3$	Maximum Flow, Minimum Cost Flow Shortest Path, Minimum Spanning Tree	Cumulative $C_{\max}$-Threshold	*NP*-hard Results Dispatching Rules IP Formulations

Table 1: Summary of related research on INDS problems.

the cost of the dummy arc from the source node to this node equal to its weight.

Table 1 provides an overview of the research related to INDS problems. This paper provides a rigorous classification of the complexity (e.g., ordinarily or strongly NP-hard, inapproximability results) of a variety of classes of INDS problems (see Section 3). It is the first paper to show that certain classes of INDS problems cannot be approximated within a constant factor unless NP $\subset$ *TIME* $(n^{O(\log \log n)})$. It builds upon the work of Nurre et al. [28] by showing that integrating concepts from the area of network flows (in particular, optimality conditions of network performance metrics) and the area of scheduling (in particular, dispatching rules) can be used to create real-time heuristic algorithms that provide near-optimal solutions. It accomplishes this by looking at three new network performance metrics (minimum cost flow, shortest path, and minimum spanning tree) and develops dispatching rules based on their associated optimality conditions (negative cycles, distance labeling, and path optimality conditions). These integrated network design and scheduling approaches are distinct from heuristic algorithms that tend to separate these two sets of decisions, like those in Averbakh and Pereira [5] (which finds the minimum spanning tree (MST) of processing times in the network and then constructs the arcs in this MST) and Elgindy et al. [10] (which finds the shortest possible path in the network and constructs this path). A rigorous set of tests across these three performance metrics and the maximum flow metric demonstrate the robustness of these dispatching rules for both the *Cumulative* and $C_{\max}$-*Threshold* objectives.

The network performance metrics, the scheduling environment, and the objectives are more general than the models considered by Averbakh and Pereira [5], Nurre et al. [28], and Elgindy et al. [10]. For the work of Matisziw et al. [24], we provide an *alternative* network performance metric to capture their motivating application. The motivation of their work regards telecommunications infrastructures and are, therefore, focused on examining a measure of connectivity between pairs of nodes. They capture this connectivity by maximizing the flow between origin/destination pairs in the network.

An alternative measure for a telecommunications network would be to minimize the costs of the operational arcs used to connect the nodes in the network, e.g., solve a minimum spanning tree problem over the network. Therefore, we provide analysis of an alternative class of INDS problems for the motivating application of Matisziw et al. [24]. This paper thus advances the literature in this emerging area of network design and scheduling.

3 Complexity Results

We now examine the complexity of these INDS problems, proving that they are all at least *NP*-hard. Table 2 provides an overview of the complexity results, showing whether each problem is ordinarily or strongly *NP*-hard along with a potential hardness of approximation classification. Most of these proofs use reductions that lead to the single-machine environment (i.e., $m = 1$) and, therefore, imply that the complexity results extend to any number m of machines.

Problem	*NP*-hard		
	Ordinarily	Strongly	Strongly No Approximation within $\ln \lvert N \rvert$
$Pm\lvert MF\rvert Cumulative$ $Pm\lvert MF\rvert C_{\max} - Threshold$		Theorem 1	 Theorems 1, Corollary 5
$Pm\lvert MCF\rvert Cumulative$ $Pm\lvert MCF\rvert C_{\max} - Threshold$		Theorem 3	 Theorem 3, Corollary 5
$Pm\lvert SP\rvert Cumulative$ $Pm\lvert SP\rvert C_{\max} - Threshold$	 Theorem 6	Elgindy et al. [10]	
$Pm\lvert \sum SP\rvert Cumulative$ $Pm\lvert \sum SP\rvert C_{\max} - Threshold$		Elgindy et al. [10]	 Theorem 4, Corollary 5
$Pm\lvert SPC\rvert Cumulative$ $Pm\lvert SPC\rvert C_{\max} - Threshold$		Elgindy et al. [10]	 Theorem 4, Corollary 5
$Pm\lvert MST\rvert Cumulative$ $Pm\lvert MST\rvert C_{\max} - Threshold$	Theorem 7 Theorem 7		

Table 2: Summary of the *NP*-hard results for the twelve INDS problems considered.

3.1 Maximum Flow INDS Complexity

Theorem 1. *$1\lvert MF\rvert C_{\max} - Threshold$ and $1\lvert MF\rvert Cumulative$ are strongly NP-hard.*

Proof. We reduce the strongly *NP*-hard problem Set Cover (see Karp [17]) to an instance of an INDS problem. The Set Cover problem has a finite family of finite sets $a_1, \ldots, a_k$ and a set of elements $e_1, \ldots, e_n$. Each element e_i belongs to one or more set, such that $\cup a_j = \{e_1, \ldots, e_n\}$. The Set Cover problem seeks to select a subfamily $\{b_h\} \subseteq \{a_j\}$ of minimum size such that $\cup b_h = \{e_1, \ldots, e_n\}$. In other words $\{b_h\}$ 'covers' every element e_i, where e_i appears in at least one of the selected sets b_h.

A network representation of an INDS problem (see Figure 1) can be created by representing each set a_j and each element e_i as nodes. The operational arc (a_j, e_i) with a capacity value of 1 and processing time of 0 is included if $e_i \in a_j$. We create a super source $\bar{S}$ and non-operational arcs $(\bar{S}, a_j)$ with capacities equal to $|a_j|$ and processing times equal to 1 for all sets. We also create a super sink node S' and operational arcs (e_i, S'), with capacities equal to 1 and processing times equal to 0. Using this representation, we define the desired threshold performance value to be n, which can only be achieved by sending a unit of flow through each node e_i. In the INDS problem, selecting to install an arc $(\bar{S}, a_j)$ (requiring one unit of time) corresponds to selecting set a_j to be a part of the subfamily $\{b_h\}$ in the Set Cover problem. The objective function value of this INDS problem represents the minimum size of the subfamily $\{b_h\}$ (i.e., the minimum number of sets) needed to cover all elements. The hardness for the *Cumulative* objective results from observing that if we define $w_1 = w_2 = \cdots = w_{T-1} = 0$ and $w_T = 1$; solving this problem then answers the question of whether we can achieve a maximum flow of n in T time periods (or, equivalently, the existence of a cover with T sets). In other words, this instance of $1|MF|Cumulative$ is the decision version of the problem $1|MF|C_{\max} - Threshold$.

Figure 1: Graphical representation of a flow-based INDS network representing the Set Cover problem used in the proofs of Theorems 1 and 3.

□

A shortcoming of the proof that $1|MF|Cumulative$ is *NP*-hard is that it focuses on the special case where only the last period has a positive weight. It would not be difficult to alter the reduction so that all time periods have positive weights and the last time period has a much larger one, so the driving factor will still be the flow in time period T. Our next proof focuses on the *Cumulative* problem where all weights are equal and there are two machines.

Theorem 2. *$P2|MF|Cumulative$ where all weights are equal, i.e., $w_1 = w_2 = \cdots = w_T = 1$, is strongly NP-hard.*

Proof. We will reduce the Set Cover problem to an instance of this INDS problem with a network very similar to that presented in the proof of Theorem 1. The main exception is that we will add in a new sink node S'' and a non-operational arc (S', S'') with a processing time of T and a capacity of n. We are then interested in maximizing the flow from $\bar{S}$ to S''. This means that it is not possible to achieve a positive flow in the network until time T since one machine must process arc (S', S''). The second machine can process up to T arcs of the form $(\bar{S}, a_j)$. This implies that the objective function value of this problem is equal to the maximum number of elements that can be covered with T sets and the problem, therefore, provides an answer to the Set Cover problem. □

3.2 Minimum Cost Flow INDS Complexity

Theorem 3. *$1|MCF|Cumulative$, $1|MCF|C_{\max} - Threshold$, and $P2|MCF|Cumulative$ with equal weights are strongly NP-hard.*

Proof. These proofs follow almost as a direct result of Theorems 1 and 2, respectively. This is because we can transform the maximum flow performance metric to a minimum cost flow performance metric by adding an arc from S' to $\bar{S}$ (or S'' to $\bar{S}$) with a cost of -1 and a large capacity. All other arcs will have a cost of zero. Therefore, the minimum cost flow value will equal the negative of the maximum flow value from $\bar{S}$ to S' (S''). □

3.3 Shortest Path INDS Complexity

In this section, we focus on the complexity of INDS problems with the three shortest path performance metrics. Parallel to this work, Elgindy et al. [10] have examined the incremental network design problem with a shortest path performance metric, which is equivalent to the INDS problem $1|SP|Cumulative$ where all arcs have a processing time of 1. In this technical report, they show that this problem is strongly *NP*-hard. This implies that all our shortest path metrics with a *Cumulative* objective are *NP*-hard. The results in this section, therefore, focus on the $C_{\max}$-*Threshold* objective.

Theorem 4. *$1|\sum SP|C_{\max} - Threshold$ and $1|SPC|C_{\max} - Threshold$ are strongly NP-hard.*

Proof. For this proof, we again reduce the known strongly *NP*-hard problem Set Cover to an instance of an INDS problem. The reduction and the network representation of the INDS problem (see Figure 2) is quite similar to the one presented in the proof of Theorem 1. The node set is the same with the exception of removing the sink node S'. The arc set is the same with the addition of operational arcs $(\bar{S}, e_i)$ with a length of one. Every other arc in the network has a length of zero. This means that there is a zero-length path from $\bar{S}$ to element node e_i only if we have constructed an arc to a set node that contains element e_i. Therefore, viewing the set of element nodes as the set of destination nodes, we set the threshold to be zero. This means that the objective function value of this problem is equal to the minimum number of sets required to cover all elements.

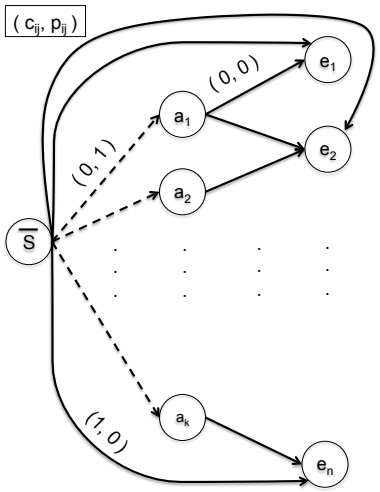

Figure 2: Graphical representation of a path-based INDS network representing the Set Cover problem used in the proof of Theorem 4.

□

Corollary 5. $1|MF|C_{\max} - Threshold$, $1|MCF|C_{\max} - Threshold$, $1|\sum SP|C_{\max} - Threshold$, *and* $1|SPC|C_{\max} - Threshold$ *cannot be approximated to within a factor of* $\ln n$ *unless NP* $\subset$ TIME $(n^{O(\log \log n)})$, *where* $n = |N|$ *represents the number of nodes in the network.*

Proof. Feige [12] showed that the Set Cover problem has an approximation threshold, unless *NP* $\subset$ *TIME* $(n^{O(\log \log n)})$, of $\ln n$ when $k < n$, where k is the number of sets and n is the number of elements. Theorems 1, 3, and 4 are all proved to be *NP*-hard through a reduction from the Set Cover problem. More importantly for this proof, the objective of a feasible solution to these $C_{\max}$-*Threshold* problems provides the number of sets required to produce a cover. This means that any approximation algorithm for $1|MF|C_{\max} - Threshold$, $1|MCF|C_{\max} - Threshold$, $1|\sum SP|C_{\max} - Threshold$, and $1|SPC|C_{\max} - Threshold$ also is an approximation algorithm for the Set Cover problem. This proves our desired result. □

Theorem 6. $1|SP|C_{\max} - Threshold$ *is NP-hard.*

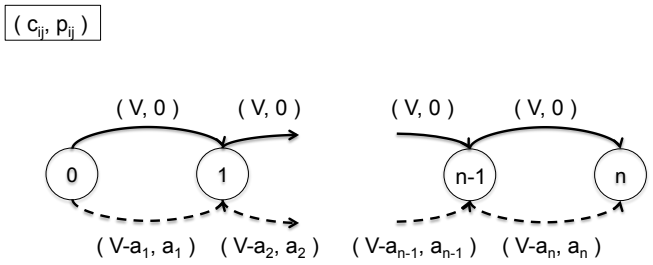

Figure 3: Graphical representation of a path-based and tree-based INDS network representing the Partition problem used in the proofs of Theorems 6 and 7.

Proof. We reduce the known *NP*-hard problem Partition to an instance of an INDS problem, visually presented in Figure 3. The Partition problem has a set of elements $a_1, \ldots, a_n$. We seek to find a subset S of these elements such that $\sum_{a_i \in S} a_i = \sum_{i=1}^{n} \frac{a_i}{2}$. A network representation of an INDS problem can be created by representing each of the n elements a_i in the Partition problem by a pair of parallel directed arcs. Specifically, the network has $n+1$ nodes and $2n$ arcs. The pair of parallel arcs representing each element a_i connects nodes $i-1$ and i, where one arc in the pair is operational and the other is non-operational. All operational arcs have a cost equal to V, a large scalar value, and a processing time of 0. Each non-operational arc has a cost equal to $V - a_i$ and a processing time of a_i, where a_i is the element this arc represents.

In this network, we are concerned with the shortest path length between node 0 and node n with a desired performance value of $nV - \sum_{i=1}^{n} \frac{a_i}{2}$. Note that the reduction in the shortest path length for processing the non-operational arc $(i-1, i)$ is precisely the processing time of the arc. If the objective function value equals $\sum_{i=1}^{n} \frac{a_i}{2}$ then we know a partition exists because a reduction of $\sum_{i=1}^{n} \frac{a_i}{2}$ has been achieved in exactly $\sum_{i=1}^{n} \frac{a_i}{2}$ time. The arcs selected for installation in the INDS problem correspond to the elements in S in the partition problem. □

3.4 Minimum Spanning Tree INDS Complexity

Theorem 7. $1|MST|C_{\max} - Threshold$ *and* $1|MST|Cumulative$ *are NP-hard.*

Proof. This proof that $1|MST|C_{\max} - Threshold$ is *NP*-hard is quite similar to that of Theorem 6 with the exception that we have undirected arcs in the network from Figure 3. The hardness for the *Cumulative* objective results from observing that if we set $w_1 = w_2 = \cdots = w_{T-1} = 0$ and $w_T = 1$, then solving this problem answers the question of whether we can achieve a network performance of $nV - \sum_{i=1}^{n} \frac{a_i}{2}$ in the MST in T time. In other words, this *Cumulative* problem is the decision version of $1|MST|C_{\max} - Threshold$. □

4 Dispatching Rules

Dispatching rules are a common heuristic approach for scheduling problems (see Pinedo [29]) that selects the next task to be processed by an available machine by examining the characteristics of the set of unprocessed tasks. These rules can be applied in real-time and often provide near-optimal solutions to scheduling problems. Therefore, the focus of this section is on a framework for creating dispatching rules for INDS problems. Dispatching rules are only one potential solution approach for these INDS problems; for example, integer programming formulations of our INDS problems can be found in the Appendix.

The success of a dispatching rule for a scheduling problem often relies on its ability to approximate the impact of completing a task on the objective of the problem. Dispatching rules that focus on scheduling the next task are often successful for traditional scheduling problems since once a task is processed by a machine, it impacts the objective function and then essentially leaves the scheduling system. INDS problems are unique in the field of scheduling since once an arc is processed and becomes operational, it interacts with other operational arcs in the network to improve its performance. For example, a newly operational arc can interact with other operational arcs in the network to allow additional flow to reach a sink node from a source node. In other words, in the problem $Pm|MF|Cumulative$, the newly operational arc can help form an *augmenting path* with other operational arcs in the network to deliver more flow through the network. These interactions (e.g., opening up new augmenting paths in a maximum flow network) will ultimately improve the performance of the network and, therefore, our dispatching rule framework will focus on understanding these interactions. In particular, our framework will focus on selecting the next *set of arcs* to be processed by the machines based on how they interact with each other and other arcs already in the operational network. For example, the dispatching rules for $Pm|MF|Cumulative$ and $Pm|MF|C_{\max}-Threshold$ will select sets of arcs that build new augmenting paths in the network. These selected arcs will then be scheduled on the machines prior to determining the next set of arcs.

The general dispatching rule framework is presented in Algorithm 1 and can be applied to any INDS problem, including ones that are not considered in this paper (e.g., network performance is measured through solving a multi-commodity flow problem or a problem has a different machine scheduling environment). The inputs for this framework are: (i) the network performance metric, (ii) the objective function, (iii) the network, and (iv) the scheduling environment. The dispatching rule then alternates between a selection routine that selects the next set of arcs to be scheduled and the scheduling routine that schedules the set of selected arcs, until a stopping criterion is met.

We now discuss the selection and scheduling routines in more detail. The selection routine depends on both how we measure the performance of the network and how we measure the impact of selected arcs on the machines. For the INDS problems considered in this paper, these considerations can be

[4]for a maximization problem, $P(G_t) \leq P$ for a minimization problem

Algorithm 1 Integrated Network Design and Scheduling Problem Dispatching Rule Framework

1: Input: Performance metric and objective function
2: Input: Network: G_0 with associated sets of operational arcs A_0 and non-operational arcs A'_0
3: Input: Resources: m parallel identical machines and objective function stopping criterion (for example, time horizon T or desired performance value P)
4: Set time t to 0
5: Calculate current performance of network and set to $P(G_t)$
6: **while** objective function stopping criterion not met (for example, $t < T$ or $(P(G_t) \geq P)^4$) **do**
7: Select set of arcs $\bar{A}_t \subseteq A'_t$ to become operational = Selection routine with input: G_t, performance metric
8: Update resources (time t) and network G_t = Scheduling routine with input: $\bar{A}_t$
9: Calculate current performance of the network $P(G_t)$
10: **end while**
11: Calculate and return objective function value

captured by answering four questions associated with the performance metric. The scheduling routine is customizable too; we present a scheduling routine that attempts to minimize the 'makespan' of completing the selected arcs. The remainder of this section is then dedicated to discussing how we customize the selection routine to each network performance metric.

Selection From a greedy perspective, a set of arcs that greatly improve the performance of the network while requiring little processing on the set of machines would be ideal to be selected. In other words, we are interested in selecting (for a maximization problem) the set of arcs that optimizes

$$\max_{\bar{A}_t \subseteq A'_t} \frac{\left|P(N_t, A_t \cup \bar{A}_t, A'_t \backslash \bar{A}_t) - P(G_t)\right|}{R(\bar{A}_t)} \tag{1}$$

where $R(\bar{A}_t)$ measures the machine (resource) requirements to process arcs in $\bar{A}_t$. For example, $R(\bar{A}_t)$ could measure the makespan of completing this set of arcs on the machines or, for parallel identical machines, could simply measure the total processing time of the arcs. For a set of arcs $\bar{A}_t$, the ratio from (1) essentially provides the network performance improvement per-unit resource consumption. The problem to determine the set of arcs that maximizes (1) is a combinatorial optimization problem with a fractional objective function and can, therefore, be quite difficult to solve to optimality. For our proposed selection routines, we will solve an approximation of (1) where we estimate the improvement in the network as measured through the optimality conditions of the network performance. The machine (resource) requirements (i.e., the denominator in (1)) will be measured as the total processing time of the arcs in the set, since this provides a good approximation of the makespan requirements for this set.

From a scheduling perspective, selecting the set of arcs that maximizes, or nearly maximizes, (1) is a generalization of the weighted shortest processing time (WSPT) rule. The WSPT rule calculates a priority ranking for each task j based on its weight w_j and processing time p_j. The task that is

selected for processing next is the one that maximizes its ratio of weight to processing time, $\frac{w_j}{p_j}$. This rule is often applied to a parallel machine environment where we seek to minimize the total weighted completion time. In order to provide this generalization of the WSPT rule for each performance metric, we will focus on a series of questions: (i) What are the 'tasks'?, (ii) What are the weights of 'tasks'?, (iii) What are the processing times of 'tasks'?, and (iv) How do we find the 'task' that maximizes (minimizes) $\frac{w_j}{p_j}$? The answers to these questions will be based on the network optimality conditions of the performance metric, e.g., the augmenting path optimality conditions will help answer these questions for the maximum flow objective.

Scheduling Once the selection routine selects a set of arcs to be scheduled and become operational, they are fed into the scheduling routine. The scheduling routine determines how the machines in the problem will process this set of selected arcs. This routine does not depend on the specific INDS problem and should, typically, focus on scheduling the selected arcs so they are completed as quickly as possible. For our INDS problems, we will implement the longest processing time (LPT) rule to schedule the selected arcs. The arcs within the selected set are ordered from longest to shortest processing time and added to a queue. When a machine becomes available, the next arc in the queue is assigned and time is allocated for its processing. The LPT rule was chosen because it often performs well when we want to complete the selected arcs as quickly as possible (see Pinedo [29]).

4.1 Maximum Flow INDS Dispatching Rule

The idea behind the Maximum Flow INDS Dispatching Rule was presented in Nurre et al. [28], but we include an overview of it here since it provides an intuitive motivation for the selection routine. This rule relies on the augmenting path optimality conditions: the flow in a network is maximum if and only if there does not exist an augmenting path from the source to the sink with residual capacity greater than 0 (see Ahuja et al. [2]). This means that we need to construct an augmenting path in the operational network to increase the flow in it and, therefore, we will view augmenting paths with at least one non-operational arc as a 'task' in the dispatching rule. The weight of the task is the amount of benefit we receive from processing it or, equivalently, the improvement in the performance of the network. Therefore, the weight of a task (or augmenting path) P will be the amount of flow it can carry, which is equal to the minimum residual capacity of arcs in it, $\min_{(i,j)\in P} r_{ij}$. The processing time of the task will then be equal to the sum of the processing times of the non-operational arcs in it (i.e., we set the processing times of all operational[5] arcs to zero).

In order to finish customizing the selection routine to create the dispatching rule, we must provide a method to determine the task that maximizes the weight to processing time ratio. In other words, we need to find the augmenting path P^* that optimizes

[5]Note that we view all arcs that have already been selected in previous iterations of the selection routine as operational, even though they may not yet be completed.

$$\max_{P \in \Phi} \frac{\min\limits_{(i,j) \in P} r_{ij}}{\sum\limits_{(i,j) \in P} p_{ij}}, \tag{2}$$

where Φ is the set of all augmenting paths in the residual network $(N_t, A_t \cup A_{t'})$ with at least one non-operational arc. Nurre et al. [28] propose a combinatorial algorithm for (2) using the following observation: Assume that we know the weight of the optimal augmenting path (equivalently, its residual capacity) that maximizes (2). The shortest processing time path in the network that only contains arcs whose residual capacity is greater than or equal to the weight of the optimal augmenting path is the optimal solution to (2). This shortest processing time path is by definition an augmenting path, as only arcs with a residual capacity greater than or equal to the optimal weight are considered for inclusion in the shortest path. The algorithm to solve (2) then iterates through all possible values of the numerator, which is equal to the number of distinct residual capacities in the network. For each of these distinct residual capacities, the shortest processing time path in the network with only arcs whose residual capacity exceeds the current threshold is determined and provides a candidate solution to (2). Therefore, we can provide the next 'task' by solving $O(|A_t \cup A'_t|)$ shortest path problems.

Table 3 gives a summary of the answers to the four questions needed to customize the selection routine for the Maximum Flow performance metric. Algorithm 2 gives the step by step process for the selection routine at time t, which utilizes the answers from Table 3. The answer to Question (i) is used in line 3 to determine the specific set of arcs. The answers to Questions (ii) and (iii) are used in lines 10 and 11, respectively, to appropriately calculate the weight and processing time of a task. The answer to Question (iv) is used in line 9, which dictates which subproblem to solve (shortest path) on the altered network. The iterative algorithm is shown in lines 7 through 16, which alters the network and keeps track of the best ratio. The selection routine appearing in Algorithm 2 can be customized to the other performance metrics by appropriately modifying these lines. The Maximum Flow INDS Dispatching rule is created by calling the selection routine, Algorithm 2, in the dispatching rule framework provided in Algorithm 1.

Customized Selection Routine for Maximum Flow INDS Problems
(i) Tasks = augmenting paths P
(ii) Weights = minimum residual capacity of augmenting path P
(iii) Processing time = $\sum_{(i,j) \in P \cap A'_t} p_{ij}$
(iv) Task maximizes ratio $\frac{w_j}{p_j}$ = Iterative algorithm using Shortest Path calculations

Table 3: Summary of the answers needed to customize the Maximum Flow Selection Routine

In the practical implementation of Algorithm 2, we can improve the run time by making some observations. Let P be a shortest path found in the altered network corresponding to residual capacity value r_{ij}. If the minimum residual capacity value of P is r_{kl}, where $r_{kl} > r_{ij}$, then we know that P will

Algorithm 2 Maximum Flow Selection Routine at time t

1: Input: Network: G_t with associated sets of operational arcs A_t and non-operational arcs A'_t
2: Set Best_Ratio $= 0$
3: Set best set of arcs (augmenting path) $P^* =$ null
4: Solve for the performance of the network $P(G_t)$ (maximum flow) and resulting residual network with residual capacity values r_{ij}
5: Put the residual capacities of all arcs $(i,j) \in A_t \cup A'_t$ or $(j,i) : (i,j) \in A_t \cup A'_t$ in array R
6: Sort R in non-decreasing order and remove duplicate entries
7: **for** $\ell = 1, \ldots, |R|$ **do**
8: Alter G_t where arc $(i,j) \in A_t \cup A'_t$ or $(j,i) : (i,j) \in A_t \cup A'_t$ are deleted if $r_{ij} < R[\ell]$
9: Solve for the shortest path $\bar{P}$ from source $\bar{S}$ to sink S' in the updated network G_t
10: Calculate the weight of the path $w_{\bar{P}} = \min_{(i,j)\in\bar{P}} r_{ij}$
11: Calculate the processing time of the path $p_{\bar{P}} = \sum_{(i,j)\in\bar{P}} p_{ij}$
12: **if** Best_Ratio $< \frac{w_{\bar{P}}}{p_{\bar{P}}}$ **then**
13: Set Best_Ratio $= \frac{w_{\bar{P}}}{p_{\bar{P}}}$
14: Set $P^* = \bar{P}$
15: **end if**
16: **end for**
17: Return best set of arcs P^*

also be the shortest path in the altered network for all residual capacity values in the range $[r_{ij}, r_{kl}]$. This means that we can skip ahead to residual capacity values greater than r_{kl}. Also, if we cannot find a path from the source node to the sink node in the altered network corresponding to residual capacity value r_{ij}, then we can break from the for loop on line 7, since no path will be found for all residual capacity values greater than r_{ij}.

4.2 Minimum Cost Flow INDS Dispatching Rule

We customize the selection routine for the Minimum Cost Flow performance metric by looking at the negative cycle optimality conditions (see Ahuja et al. [2]), which state that the flow in a network is minimum if and only if there does not exist a negative cost directed cycle in the residual network. Therefore, we view sets of residual arcs that form a negative directed cycle C as the tasks. The weight of a task equals the minimum residual capacity of C multiplied by the cost of the cycle, $\sum_{(i,j)\in C} c_{ij}$, which provides the improvement in the objective function by pushing flow through this negative cycle. The processing time of a task equals the sum of the processing times of non-operational arcs within the cycle. In order to determine the tasks that minimize the weight to processing time ratio, we need to find the cycle C^* that optimizes

$$\min_{C \in \Gamma} \frac{(\min_{(i,j) \in C} r_{ij}) \sum_{(i,j) \in C} c_{ij}}{\sum_{(i,j) \in C \cap A'_t} p_{ij}}, \tag{3}$$

where Γ is the set of all negative cycles in the residual network $(N_t, A_t \cup A_{t'})$ with at least one non-operational arc. Note that, because the minimum cost flow performance metric is a minimization problem, we appropriately focus on minimizing the weight to processing time ratio.

We use a similar observation as the one presented for the maximum flow INDS dispatching rule to find the task that minimizes (3): If we know the minimum residual capacity value of the cycle that minimizes (3) and alter the network to include only residual arcs at or above this residual capacity value, then the cycle that minimizes the cost to time ratio problem (see Ahuja et al. [2]) also minimizes (3). We do not know the optimal minimum residual capacity value, and therefore must solve at most $O(|A_t \cup A'_t|)$ minimum cost to time problems to find the cycle that minimizes (3).

Table 4 summarizes the answers to the four customizable features of the selection routine, catered to the minimum cost flow performance metric. As we saw with the maximum flow performance metric, the answers to these questions directly influence steps in the selection routine algorithm. We do not explicitly present the entire minimum cost flow selection routine pseudocode but instead identify the changes necessary to Algorithm 2. In line 3, we focus on negative cycles instead of augmenting paths; in line 4, the performance of the network is the minimum cost flow value; in line 9, we input the answers from Question (iv) and solve a minimum cost to time problem instead of a shortest path problem; in lines 10 and 11, the weight and processing times are calculated according to the answers to Questions (ii) and (iii) in Table 4; and in line 12, we switch to a greater than sign to replace the less than sign. The Minimum Cost Flow INDS Dispatching rule is then created by calling this modified selection routine in the dispatching rule framework provided in Algorithm 1.

Customized Selection Routine for Minimum Cost Flow INDS Problems
(i) Tasks = Negative cycles C
(ii) Weights = min residual capacity of cycle C * $\sum_{(i,j) \in C} c_{ij}$
(iii) Processing time = $\sum_{(i,j) \in C \cap A'_t} p_{ij}$
(iv) Task minimizes ratio $\frac{w_j}{p_j}$ = Iterative algorithm using Minimum Cost to Time calculations

Table 4: Summary of the answers needed to customize the Minimum Cost Flow Selection Routine

4.3 Shortest Path INDS Dispatching Rule

It is well known that a minimum cost flow problem can be used to solve a shortest path problem. We use this observation and the minimum cost flow INDS dispatching rule to form the dispatching

rule for INDS problems with the shortest path performance metrics. We create a network where the source node of the shortest path problem has a supply of $|D|$ and the demand of each destination node $i \in D$ is equal to 1. The cost of all arcs are their lengths and their capacity is set equal to $|D|$. The route that a unit of flow takes from the source node to destination node $i \in D$ is the shortest path to node i. Note that, given an integral flow, there are only $O(|D|)$ number of distinct residual capacities which implies that we need to only solve $O(|D|)$ minimum cost to time problems for the Shortest Path INDS Dispatching rule.

The previous discussion provides a clear approach for INDS problems where we are summing the shortest path lengths to the multiple destinations. For the performance metric SPC, that focuses on minimizing the maximum shortest path length to destination nodes $i \in D$ (called the destination node on the maximum shortest path i'), we know that we need to shorten the path to i' to improve the performance of the network. Therefore, the selection routine for this INDS problem will only focus on the node with the current maximum shortest path length and view this as the *only* current destination node. Note that this node will change over time as more arcs become operational and the selection routine will appropriately focus on the correct single destination node (i.e., the one that currently has the maximum shortest path length).

The implementation of the minimum cost flow INDS dispatching rule for the shortest path INDS problems can be related to the distance label optimality conditions for shortest path problems. We know that if the dispatching rule selects some set of arcs C^* that form a negative cycle in the minimum cost flow network with a residual capacity of r_{ij}, then the shortest paths of r_{ij} destination nodes will be decreased by the length of the negative cycle. This means that the dispatching rule is measuring the improvement in the performance of the network by the number of destination nodes whose distance labels decrease times the amount by which they decrease.

4.4 Minimum Spanning Tree INDS Dispatching Rule

We now customize the selection routine for INDS problems with minimum spanning tree performance metrics by utilizing the path optimality conditions: a tree T^* is a minimum spanning tree if and only if for every arc $(i,j) \notin T^*$ we have $c_{ij} \geq c_{k\ell}$ for all arcs (k,ℓ) in the unique path P_{ij} from i to j in T^*. Without loss of generality, we assume first that a feasible minimum spanning tree exists in the starting network G_0[6].

In order to improve the network performance, we must install some non-operational arc (i,j) whose inclusion in the network would violate the path optimality conditions. In other words, this arc (i,j) will have a cost lower than some $(k,\ell) \in P_{ij}$ in the path P_{ij} currently in the minimum spanning tree. Therefore, we will view the tasks in the problem as non-operational arcs and measure the improvement in the network performance as $c_{ij} - \max_{(\kappa,\lambda)\in P_{ij}} c_{\kappa\lambda}$. The processing time of a task

[6]If G_0 does not start with a spanning tree, one can be created by connecting a new node to every node in the network through arcs with arbitrarily high cost.

then simply equals the processing time of the arc. The calculations to find the next arc to be selected become easier since we simply need to calculate the improvement for each non-operational arc and its appropriate ratio. This calculation requires knowing the paths between each pair of nodes, which can be done by applying a search algorithm $O(|N|)$ times.

Table 5 presents the summary of answers for the four questions needed to customize the selection routine. These answers are again used in the step by step algorithm for the selection routine. The specific pseudocode is not included, due to the similarities to Algorithm 2. Instead, we outline the differences. In line 3, the set of arcs is just a single non-operational arc; in line 4, the performance of the network is the cost of a minimum spanning tree and the resulting network includes paths between each set of nodes P_{ij}; lines 5, 6, 8, and 9 are not necessary due to the linear search; line 7 should iterate over all non-operational arcs in A'_t; lines 10 and 11 should calculate the weight and processing time according to the answers to Questions (ii) and (iii) from Table 5; and line 12 should focus on the minimum ratio by utilizing a greater than sign to replace the less than sign.

Customized Selection Routine for Minimum Spanning Tree INDS Problems
(i) Tasks = Uninstalled arcs (i, j)
(ii) Weights = $c_{ij} - \max_{(k,l)\in P_{ij}} c_{kl}$
(iii) Processing time = p_{ij}
(iv) Task minimizes ratio $\frac{w_j}{p_j}$ = linear over all non-operational arcs, $O(\|N\|)$ searches

Table 5: Summary of the answers needed to customize the Minimum Spanning Tree Selection Routine

5 Computational Results

We now present the results of case studies testing the performance of the INDS dispatching rules on realistic data sets. INDS problems can be applied to a wide range of real-life applications. For these case studies, we focus on realistic networks representing the power and telecommunications infrastructures of lower Manhattan in New York City. These data sets were created through close collaboration with the infrastructure managers (see Lee et al. [21]). Through the application of an INDS model, we create restoration plans (what to repair, who performs the repair, and when the repair is performed) for different damage scenarios, performance metrics, and objective functions. The focus of these computational tests, in contrast to Nurre et al. [28], is to capture the performance of the dispatching rule framework on a wide variety of different damage scenarios and performance metrics. Nurre et al. [28] only focus on a single damage scenario for this data set.

The Manhattan power network has $|N| = 1603$ nodes and $|A_0 \cup A'_0| = 2621$ arcs. We note that this power network does not include any temporary design alternatives (power shunts) as was included in [28], but instead only considers repairing damaged arcs as design options. This network has 14 supply nodes and 134 demand nodes. By applying network expansion techniques (see Section

2.1), we can model this network, when appropriate, as one with either or both a single supply node and single demand node. The Manhattan telecommunications network has $|N| = 547$ nodes and $|A_0 \cup A_0'| = 548$. The maximum flow, minimum cost flow, and shortest path performance metrics are tested on instances based on the power network. The minimum spanning tree performance metric is tested on instances based on the telecommunications network. The specific procedures to determine the set of damaged arcs for each performance metric will be discussed in their respective subsections.

Each damaged arc is assigned a random processing time equal to 1, 2, or 3 time periods. We test these scenarios using 1, 2, and 3 machines which is representative of the machines used in Lee et al. [20], and correspond to sets of trained power work crews. The objective function value and elapsed time needed to arrive at a solution is captured for each run of the dispatching rules. These values are benchmarked against the objective function value and elapsed time of IP formulations solved using CPLEX 12.0 by calculating the optimality gap (percentage difference) between the objective function values found. The IP formulations use a time-indexed formulation of the scheduling decisions; full details of these formulations can be found in the Appendix. A time limit of 4 hours is set and the heuristic solution is fed as a warm start (initial feasible integer solution). We experimented with the probing parameter for the 4 hour time limit tests but found no significant improvement and therefore use the default CPLEX settings. We further consider the use of CPLEX as a real-time heuristic by setting time limits of 5 minutes (300 seconds) and 30 minutes (1800 seconds) and focusing the search on obtaining high-quality solutions (CPLEX parameter MIPEmphasis set to 1 focusing on finding feasible solutions). We have experimented with the MIPEmphasis parameter focusing on optimality and found that the focus on feasibility produces higher quality solutions. All tests for both the dispatching rules and IP formulations were performed on a laptop with 2.16 GHz Intel Core 2 Duo Processor with 3GB of RAM, which is representative of the computing resources present among infrastructure managers. Also, note that, for the motivating application of infrastructure restoration, decision-makers may not have access to commercial optimization software packages (due to license costs) or open-source optimization packages (due to the required technical support in utilizing them). Therefore, the dispatching rules, from an application perspective, are a very important alternative to solving integer programming formulations of INDS problems.

The *Cumulative* objective function time horizon T is set to 60 for all performance metrics, which is identical to the horizons used by Cavdaroglu et al. [6] and Nurre et al. [28]. The $C_{\max}$-*Threshold* objective function desired network performance values are set to represent re-establishing 75% of the 'disrupted' performance. In other words, we examine the best possible performance in the network with all arcs operational and then set our performance threshold to be the *(current performance in the damaged network) + 75%(best performance-current performance).* We further test the maximum flow performance metrics for a desired network performance value of 100% of the best performance, since these will be critical in the instance generation for the minimum cost flow and shortest path performance metrics.

		Dispatching Rule		CPLEX Heuristics		CPLEX 12.0	
Machines	Percentage	Time (s)	Gap (%)	300s Gap (%)	1800s Gap (%)	Time (s)	Gap (%)
1	25%	28.75	0.72%	1.98%	0.50%	12,335.11	0.34%
	50%	31.97	0.07%	0.10%	0.04%	5,178.08	0.03%
	75%	19.33	0.40%	20.83%$^{\dagger 3}$	1.06%	14,400.00	0.40%
2	25%	32.13	0.85%	12.05%	2.61%	14,400.00	0.66%
	50%	68.10	0.15%	-$^{\dagger 5}$	0.22%	7,363.34	0.03%
	75%	43.22	0.39%	-$^{\dagger 5}$	1.13%	12,100.49	0.27%
3	25%	29.33	0.61%	13.97%$^{\dagger 1}$	8.87%	14,400.00	0.51%
	50%	139.59	0.67%	-$^{\dagger 5}$	1.05%	14,400.00	0.46%
	75%	57.45	0.51%	-$^{\dagger 5}$	10.93%$^{\dagger 2}$	14,400.00	0.35%

Table 6: $Pm|MF|Cumulative$ computational results comparing the performance of the dispatching rule, CPLEX heuristics, and CPLEX 12.0.
$^{\dagger i}$ signifies that i out of 5 instances did not find a feasible solution and therefore are not incorporated in the average calculation.

5.1 Maximum Flow Performance Metric

We test the Manhattan power network with maximum flow performance metrics for many different scenarios. We define the set of damaged arcs in A'_0 by randomly selecting 25%, 50%, and 75% of the arcs. Five random instances are created for each of these levels of damage and tests were run on each instance for each number of machines. Each instance is tested for the *Cumulative*, 75% $C_{\max}$-*Threshold*, and 100% $C_{\max}$-*Threshold* objective functions. The results are presented in Tables 6, 7, 8, for *Cumulative*, $C_{\max}$-*Threshold* 75% and $C_{\max}$-*Threshold* 100%, respectively.

Each row in the table shows the average time and optimality gap over the 5 random instances. When running the IP formulations, if CPLEX could not find the optimal solution within the set time limit of 4 hours, we capture both the best known feasible solution (lower bound) and the current relaxed solution value (upper bound). If the optimal solution is found, then the lower bound and upper bound are equal.

For each of the heuristics (dispatching rule, CPLEX with 5 minute time limit, and CPLEX with 30 minute time limit), we capture the best known feasible integer solution and benchmark this value against the upper bound found during the CPLEX 4 hour run. An optimality gap is then calculated by taking the percentage difference between these two values. Therefore, if CPLEX was unable to find an optimal solution within the given 4 hour time limit, the heuristic optimality gaps are an *overestimate* of the actual optimality gap value, since the upper bound from CPLEX is a relaxed value. We note that since the dispatching rule solution was fed as a warm start, the optimality gap of CPLEX with the 4 hour time limit will always be closer to 0% than that of the dispatching rule.

In Table 6, we see that the dispatching rule arrives at near optimal solutions in less than 2.5 minutes and in some cases as quick as 20 seconds. Both CPLEX heuristics do not perform well, with the dispatching rule always finding a higher quality solution (smaller optimality gap). For the 5

		Dispatching Rule		CPLEX Heuristics		CPLEX 12.0	
Machines	Percentage	Time (s)	Gap (%)	300s Gap (%)	1800s Gap (%)	Time (s)	Gap (%)
1	25%	18.90	2.34%	0.00%	0.00%	1.37	0.00%
	50%	61.35	1.32%	0.00%	0.00%	3.79	0.00%
	75%	87.02	0.36%	0.00%	0.00%	8.29	0.00%
2	25%	16.63	3.27%	0.91%	0.91%	2,881.07	0.91%
	50%	57.87	1.82%	0.27%	0.27%	2,888.42	0.27%
	75%	81.84	0.70%	0.22%	0.22%	3,916.89	0.22%
3	25%	18.77	6.36%	1.33%	1.33%	2,882.46	1.33%
	50%	82.56	1.56%	0.00%	0.00%	98.60	0.00%
	75%	98.73	0.88%	0.71%	0.71%	11,521.32	0.71%

Table 7: $Pm|MF|C_{\max} - Threshold$ computational results comparing the performance of the dispatching rule, CPLEX heuristics, and CPLEX 12.0 for the 75% threshold.

		Dispatching Rule		CPLEX Heuristics		CPLEX 12.0	
Machines	Percentage	Time (s)	Gap (%)	300s Gap (%)	1800s Gap (%)	Time (s)	Gap (%)
1	25%	21.03	4.88%	0.00%	0.00%	1.61	0.00%
	50%	112.35	6.99%	0.00%	0.00%	1,347.62	0.00%
	75%	130.07	7.11%	1.51%	1.51%	$13,759.31^{*2}$	$1.33\%^{*2}$
2	25%	18.93	6.70%	0.98%	0.98%	6,155.98	0.98%
	50%	83.55	10.81%	4.17%	4.17%	14,400.00	3.44%
	75%	136.73	11.05%	5.86%	5.86%	14,400.00	3.17%
3	25%	16.68	10.01%	3.77%	3.77%	8,677.14	2.34%
	50%	75.89	11.46%	4.78%	4.62%	14,400.00	3.50%
	75%	121.09	12.04%	7.31%	6.74%	14,400.00	3.15%

Table 8: $Pm|MF|C_{\max} - Threshold$ computational results comparing the performance of the dispatching rule, CPLEX heuristics, and CPLEX 12.0 for the 100% threshold.
^{*2} Two of the 5 instances ran into memory errors. The solution and time right before running into the memory errors were captured and averaged into the values displayed.

minute heuristic, it is often the case that no feasible integer solution is identified. Further, the full 4 hour run of CPLEX 12.0 often does not determine the optimal solution within the time limit and tends to only identify solutions with slightly better objectives than the one returned by the dispatching rule.

Tables 7 and 8 indicate that CPLEX performs much better on the $C_{\max}$-*Threshold* objective than the *Cumulative* objective for the maximum flow performance metric, which can be partially attributed to the compactness of the $C_{\max}$-*Threshold* IP formulation. The dispatching rule still delivers solutions of reasonably high quality in under 2 minutes for both the 75% and 100% $C_{\max}$-*Threshold* objectives. The gaps for the 75% $C_{\max}$-*Threshold* objective vary from under 1% to 6.36%. The gaps for the dispatching rule do increase to around 10% for the 100% $C_{\max}$-*Threshold* objective; however, CPLEX also struggles with these instances as the time limit is reached on some instances. Therefore, the maximum flow dispatching rule is a robust real-time algorithm for INDS problems that can be applied to instances with either the *Cumulative* or $C_{\max}$-*Threshold* objectives.

5.2 Minimum Cost Flow Performance Metric

We test the minimum cost flow performance metric on instances that are based on the ending network resulting from the maximum flow 100% $C_{\max}$-*Threshold* INDS problem. In other words, we create a damage instance in the network, determine a solution (through the use of the dispatching rule) to the associated maximum flow 100% $C_{\max}$-*Threshold* INDS problem, and then view this network as the starting network of the minimum cost flow INDS problem. This ensures that a feasible flow exists in the starting network (since all demand can be met) and also means that slightly less than 25%, 50%, and 75% of the network is damaged.

The motivation for this generation procedure is that it mimics a potential approach of power infrastructure managers when repairing damage to their network. The first 'phase' of the approach is an immediate restoration phase which focuses on restoring all power as quickly as possible, i.e., the maximum flow 100% $C_{\max}$-*Threshold* objective INDS problem. Once this phase is complete, the focus then shifts to a recovery phase that seeks to improve the operational characteristics of the current network. This would correspond to the operational costs of meeting demand in the network, implying that our network performance metric will be the minimum cost flow metric.

Table 9 shows the results for the minimum cost flow performance metric with the *Cumulative* objective. These results indicate that the dispatching rule arrives at near-optimal solutions within, on average, 0.2% of the lower bound on the optimal solution. The 5 minute CPLEX heuristic performs well for smaller damage (25%) but struggles with the larger damage scenarios. The 30 minute CPLEX heuristic performs comparable to the dispatching rule; however, we note that the dispatching rule takes less computational time and determines better solutions for the largest amount of damage (75%) without requiring the use of a commercial software package. Further, we note that for the instances with large amounts of damage (50% and 75%), CPLEX does not always determine the optimal solution within the 4 hour time limit.

		Dispatching Rule		CPLEX Heuristics		CPLEX 12.0	
Machines	Percentage	Time (s)	Gap (%)	300s Gap (%)	1800s Gap (%)	Time (s)	Gap (%)
1	25%	530.32	0.05%	0.03%	0.00%	1,231.73	0.00%
	50%	479.12	0.08%	0.43%	0.03%	13,038.57	0.02%
	75%	336.10	0.12%	3.84%	0.14%	14,400.00	0.09%
2	25%	872.37	0.05%	0.18%	0.01%	2,845.95	0.00%
	50%	871.60	0.09%	7.96%	0.07%	11,156.28	0.01%
	75%	621.17	0.15%	5.66%	0.81%	14,400.00	0.12%
3	25%	1,263.40	0.05%	0.49%	0.02%	3,868.60	0.00%
	50%	1,199.96	0.09%	9.88%	0.11%	11,933.47	0.01%
	75%	886.28	0.15%	7.13%	1.02%	14,400.00	0.13%

Table 9: $Pm|MCF|Cumulative$ computational results comparing the performance of the dispatching rule, CPLEX heuristics, and CPLEX 12.0.

		Dispatching Rule		CPLEX 12.0	
Machines	Percentage	Time (s)	Gap (%)	Time (s)	Gap (%)
1	25%	785.41	1.90%	1.11	0.00%
	50%	1,017.60	1.95%	1.29	0.00%
	75%	957.72	2.09%	1.72	0.00%
2	25%	786.58	4.02%	3.64	0.00%
	50%	1,016.83	10.51%	2.37	0.00%
	75%	953.65	14.76%	1.42	0.00%
3	25%	784.22	6.00%	1.69	0.00%
	50%	1,015.84	13.24%	1.61	0.00%
	75%	947.52	17.25%	1.92	0.00%

Table 10: $Pm|MCF|C_{\max} - Threshold$ computational results comparing the performance of the dispatching rule and CPLEX 12.0.

Table 10 shows the results for the minimum cost performance metric with $C_{\max}$-*Threshold* objective, where we seek to repair 75% of the rise in cost due to the damage in the network. CPLEX 12.0 provides the optimal solution extremely quickly for this set of instances. The dispatching rule provides solutions of high-quality for the single-machine setting. The computational time requirements of the dispatching rule for the minimum cost flow $C_{\max}$-*Threshold* objective is quite similar to those for the minimum cost flow *Cumulative* objective. We exclude the CPLEX heuristic columns in this table, as they would be identical to the CPLEX 12.0 columns, due to the quick computational time needed.

5.3 Shortest Path Performance Metric

We test the shortest path performance metric with multiple destinations on the same set of damage instances that were used for the minimum cost flow performance metric tests. The main difference is that each demand node in the infrastructure network is viewed as a destination node for the shortest path performance metric. A super-supply node is added that connects to each supply node and, therefore, the network performance metric is the sum of the shortest path lengths from some supply node to each demand node. Note that a path must exist to each of these destination nodes in the initial operational network, because the instance was generated by meeting 100% of demand in the network.

		Dispatching Rule		CPLEX Heuristics		CPLEX 12.0	
Machines	Percentage	Time (s)	Gap (%)	300s Gap (%)	1800s Gap (%)	Time (s)	Gap (%)
1	25%	99.96	0.64%	0.00%	0.00%	199.50	0.00%
	50%	97.61	0.25%	0.04%	0.03%	14,400.00	0.03%
	75%	65.36	0.57%	0.14%	0.07%	14,400.00	0.06%
2	25%	104.90	1.85%	0.01%	0.00%	530.30	0.00%
	50%	114.35	1.77%	0.18%	0.00%	2,730.73	0.00%
	75%	66.09	2.25%	6.21%	0.08%	13,154.02	0.02%
3	25%	84.47	2.40%	0.07%	0.00%	1,634.27	0.00%
	50%	108.73	3.14%	2.18%	0.15%	14,290.50	0.01%
	75%	60.25	3.66%	8.99%	0.23%	14,400.00	0.05%

Table 11: $Pm|\sum SP|Cumulative$ computational results comparing the performance of the dispatching rule, CPLEX heuristics, and CPLEX 12.0.

Table 11 shows the results for the tests on the INDS problem with the sum of shortest path lengths to multiple destinations performance metric with *Cumulative* objective. The dispatching rule arrives at near-optimal solutions in less than 2 minutes, with gaps less than 3.7% for all instances. The two CPLEX heuristics also perform well; however, the 5 minute heuristic struggles with the larger damage percentages. The 4 hour CPLEX run (CPLEX 12.0 column) does find the optimal solution for instances with 25% damage in under 30 minutes but fails to determine the optimal solution within the 4 hour time limit for almost all instances with larger amounts of damage.

		Dispatching Rule		CPLEX 12.0	
Machines	Percentage	Time (s)	Gap (%)	Time (s)	Gap (%)
1	25%	112.83	1.94%	1.11	0.00%
	50%	157.71	3.35%	1.29	0.00%
	75%	148.65	3.66%	1.72	0.00%
2	25%	114.64	5.51%	3.64	0.00%
	50%	161.08	4.45%	2.37	0.00%
	75%	148.77	4.23%	1.42	0.00%
3	25%	112.75	3.50%	1.69	0.00%
	50%	158.99	5.18%	1.61	0.00%
	75%	150.12	5.70%	1.92	0.00%

Table 12: $Pm|\sum SP|C_{\max} - Threshold$ computational results comparing the performance of the dispatching rule and CPLEX 12.0.

Table 12 shows the results of the tests for the INDS problem with the sum of shortest path lengths to multiple destinations performance metric with $C_{\max}$-*Threshold* objective with a desired $C_{\max}$-*Threshold* performance value set to 75% of the performance increase due to damage. Similar to the results found for the minimum cost flow $C_{\max}$-*Threshold* instances, CPLEX 12.0 arrives at the optimal solution extremely quickly. The dispatching rule does yield solutions that are near optimal in under 3 minutes for these instances.

5.4 Minimum Spanning Tree Performance Metric

We test the minimum spanning tree (MST) performance metric on the Manhattan telecommunications network. This network has a topology similar to many telecommunications network: a main 'transmission' network that connects hub nodes in the network and then 'star' networks, originating from these hub nodes, that disperse to the demand points.

The topology of the telecommunications network means that a demand point typically can only be connected to the tree through one arc connecting it to its hub node. This implies that if this arc is damaged, then we must repair it in order to provide services to the demand node. Due to this structure, we damage the same percentage of arcs in the main transmission network and star network connecting hubs to demand nodes. This means that, for example, 25% of arcs in the main transmission network are damaged and then 25% of arcs in the star networks are damaged, resulting in 25% of the total network damaged.

To ensure that the initial damaged network has a starting operational spanning tree, a 'dummy' node is added that connects to every node in the network through operational 'dummy' arcs. We apply different penalty costs for the 'dummy' arcs for the *Cumulative* and the $C_{\max}$-*Threshold* objectives.

For the *Cumulative* objective, the cost of a dummy arc is set to be the maximum arc cost in the network + 1, i.e., $\max_{(i,j)\in A_0 \cup A'_0} c_{ij} + 1$. These costs ensure that a dummy arc will never be included in

the MST in place of an original arc in the network and, more importantly, do not skew the *Cumulative* objective function as arbitrarily large dummy arc costs will drive the cost of the starting MST high, thus artificially inflating the objective function value (decreasing the optimality gaps). For the $C_{\max}$-*Threshold* objective, the dummy arc costs do not directly impact the objective function value, and therefore are set high without affecting the optimality gaps. We set the costs of the dummy arcs to be $1 + \sum_{(i,j)\in A_0 \cup A_0'} c_{ij}$.

		Dispatching Rule		CPLEX 12.0	
Machines	Percentage	Time (s)	Gap (%)	Time (s)	Gap (%)
1	25%	8.29	8.92%	-*5	-*5
	50%	8.89	9.37%	-*5	-*5
	75%	10.66	8.87%	-*5	-*5
2	25%	10.91	11.38%	-*5	-*5
	50%	11.78	12.62%	-*5	-*5
	75%	12.99	12.05%	-*5	-*5
3	25%	11.17	12.21%	-*5	-*5
	50%	13.57	15.07%	-*5	-*5
	75%	15.01	14.58%	-*5	-*5

Table 13: $Pm|MST|Cumulative$ computational results comparing the performance of the dispatching rule and CPLEX 12.0.
*5 All 5 instances ran into memory errors.

For these computational tests, we utilize a multi-commodity network flow integer programming formulation of the MST. This leads to $O(|N|^2)$ variables to represent the MST decisions for *a single time period*. This means that we require $O(|N|^2T)$ variables in the integer programming formulation of the *Cumulative* objective. CPLEX 12.0 runs into memory issues for all test instances of the *Cumulative* problem, as shown in Table 13.

To benchmark the performance of the *Cumulative* MST dispatching rule, we calculate a lower bound on the *Cumulative* MST by reformulating the $C_{\max}$-*Threshold* IP to minimize the performance of the MST subject to the constraint that all work is completed by time T. In other words, we seek to minimize the resulting MST from processing arcs for T time units on the available machines. The value of this MST times the time horizon yields a lower bound for the *Cumulative* MST problem because this value represents the best possible MST at the end of the horizon. CPLEX 12.0 also had difficulty solving this problem to optimality quickly and, therefore, we used the optimal value of the *linear relaxation* of this problem in place of the best possible MST. The optimality gap values for the dispatching rule presented in Table 13 are based on this lower bound value. We show that the dispatching rule continues to arrive at solutions very quickly, and of high quality, even with a loose lower bound benchmark.

In Table 14, we display the results of the computational tests with the MST $C_{\max}$-*Threshold*

		Dispatching Rule		CPLEX Heuristics		CPLEX 12.0	
Machines	Percentage	Time (s)	Gap (%)	300s Gap (%)	1800s Gap (%)	Time (s)	Gap (%)
1	25%	4.10	2.12%	14.21%$^{\dagger 4}$	8.04%$^{\dagger 3}$	14400.00	2.12%
	50%	8.47	2.04%	-$^{\dagger 5}$	-$^{\dagger 5}$	14400.00	2.04%
	75%	16.33	7.22%	-$^{\dagger 5}$	-$^{\dagger 5}$	14400.00	7.22%
2	25%	3.99	2.57%	-$^{\dagger 5}$	-$^{\dagger 5}$	14400.00	2.57%
	50%	8.46	2.36%	-$^{\dagger 5}$	-$^{\dagger 5}$	14400.00	2.36%
	75%	17.37	7.43%	-$^{\dagger 5}$	-$^{\dagger 5}$	14400.00	7.43%
3	25%	4.63	2.59%	-$^{\dagger 5}$	-$^{\dagger 5}$	11555.40	2.59%
	50%	8.74	2.39%	-$^{\dagger 5}$	-$^{\dagger 5}$	14400.00	2.39%
	75%	15.43	7.93%	-$^{\dagger 5}$	-$^{\dagger 5}$	14400.00	7.93%

Table 14: $Pm|MST|C_{\max} - Threshold$ computational results comparing the performance of the dispatching rule, CPLEX heuristics, and CPLEX 12.0.
$^{\dagger i}$ signifies that i out of 5 instances did not find a feasible solution and therefore are not incorporated in the average calculation.

objective. First, we notice that both CPLEX heuristics do not perform well, as in most cases no feasible integer solution is identified. For the 4 hour run of the IP (CPLEX 12.0 column), the dispatching rule solution was not improved upon as is shown by identical optimality gaps for the dispatching rule and CPLEX 12.0 column. Further, we point out the quick computational time needed for the dispatching rule to identify high quality solutions.

6 Conclusions

This paper has presented a class of problems that integrates network design and scheduling decisions. The combination of these two sets of decisions allows for a realistic modeling of many real-world applications, such as infrastructure restoration and humanitarian logistics. We considered INDS problems with parallel identical machines and network performance metrics based on classic network flow problems. These performance metrics were then incorporated into objective functions focusing on the *Cumulative* performance of the network over time and the amount of time required to reach a certain $C_{\max}$-*Threshold* level of performance.

We provided the complexity of all INDS problems considered, showing that they were all at least *NP*-hard. Certain performance metrics with the $C_{\max}$-*Threshold* objective were shown to not have approximation algorithms within a factor of $\ln |N|$, where N is the set of nodes in the network. This motivated the need for effective heuristic algorithms for INDS problems.

We created a novel heuristic dispatching rule algorithm framework that can be applied to all INDS problems. This dispatching rule is novel in the sense that a *set* of arcs are selected and then processed by machines to become operational in the network. By looking at the optimality conditions of the individual performance metrics, we make small customizations to the dispatching

rule framework to define which *set* of arcs should be considered and how we find the best set of arcs to make operational. The dispatching rules were applied to determine restoration plans for many different damage scenarios for data sets representing the power and telecommunications networks of lower Manhattan. The dispatching rule was benchmarked against IP formulations of the problems solved using the commercial software package CPLEX 12.0.

The results of these tests show that the dispatching rule consistently arrives at near-optimal solutions quickly. This means that the dispatching rule can be used for both planning and real-time situations. CPLEX 12.0 and heuristics that used CPLEX 12.0 were not capable of providing a level of consistency across all INDS problems; the solution of many instances reached the 4 hour time limit without verifying an optimal solution, were inhibited by memory errors, or the CPLEX-based heuristics did not determine feasible integer solutions. This lack of consistency was especially true when the difficulty of the problem increased (i.e., more machines and greater damage to the network). In application areas, the speed of the dispatching rule allows for many tests with different input (i.e., potential damage scenarios) to be completed and analyzed in a short amount of time. In addition, the dispatching rules do not require the use of commercial optimization software, which may not be readily available to decision-makers in the application areas. Therefore, the dispatching rules are a valuable tool to aid decision-makers due to their high-quality performance.

Our complexity results would extend to similar INDS problems when the processing times of the arcs are dependent on the machine which will process them (which is referred to as unrelated machines, or Rm, in the scheduling literature). This is because the identical machine environment is a special case of the unrelated machine environment. Our proposed dispatching rules could be applied to this setting by setting the processing time of an arc to be the average processing time across machines, although we would expect these rules to perform worse. Instead, future work could look at customizing the dispatching rules to this setting to incorporate the differences between the machines. The integer programming formulations could be easily modified to handle the unrelated machine environment by including the appropriate processing time in each of the machine constraints.

The INDS problems presented are a class of problems that is general enough to allow for many future avenues of research. INDS problems should also be examined under different scheduling environments such as parallel machines with different speeds, which captures the different skill levels of work groups for the infrastructure restoration application. Also, these skill levels should be captured by defining which machines can process specific tasks. Further, INDS problems could be examined where the tasks have more requirements, such as those with precedence constraints or release dates. Large-scale integer programming techniques should also be examined with hopes to arrive at optimal solutions in a reasonable amount of time for INDS problems.

Disclaimer

The views expressed in this article are those of the authors and do not reflect the official policy or position of the United States Air Force, Department of Defense, or the United States Government.

References

[1] R. Agarwal and O. Ergun, Ship scheduling and network design for cargo routing in liner shipping, Transportation Sci 42 (2008), 175–196.

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

[3] C.C. Ang, Optimized recovery of damaged electrical power grids, Master's Thesis, Department of Operations Research, Naval Postgraduate School, Monterey, California, 2006.

[4] I. Averbakh, Emergency path restoration problems, Discr Optim 9 (2012), 58–64.

[5] I. Averbakh and J. Pereira, The flowtime network construction problem, IIE Trans 44 (2012), 681–694.

[6] B. Cavdaroglu, E. Hammel, J.E. Mitchell, T.C. Sharkey, and W.A. Wallace, Integrating restoration and scheduling decisions for disrupted interdependent infrastructure systems, Ann Oper Res 203 (2013), 279–294.

[7] C. Clastres, Smart grids: Another step towards competition, energy security and climate change objectives, Energy Policy 39 (2011), 5399–5408.

[8] T.G. Crainic, Service network design in freight transportation, Eur J Oper Res 122 (2000), 272–288.

[9] R. DeBlasio and C. Tom, Standards for the smart grid, Proc 2008 IEEE Energy 2030 Conference, Atlanta, GA, 2008, pp. 1–7.

[10] T. Elgindy, A.T. Ernst, M. Baxter, M.W.P. Savelsbergh, and T. Kalinowski, Incremental network design with shortest paths, Technical report, CSIRO Mathematics Informatics and Statistics, Australia, 2013, Available online at `www.optimization-online.org/DB_FILE/2013/01/3752.pdf`.

[11] H. Farhangi, The path of the smart grid, IEEE Power Energy Magazine 8 (2010), 18–28.

[12] U. Feige, A threshold of ln n for approximating Set Cover, J ACM 45 (1998), 634–652.

[13] M. Glinkowski, J. Hou, and G. Rackliffe, Advances in wind energy technologies in the context of smart grid, Proc IEEE 99 (2011), 1083–1097.

[14] R.L. Graham, E.L. Lawler, J.K. Lenstra, and A.H.G. Rinnooy Kan, Optimization and approximation in deterministic sequencing and scheduling: A survey, Ann Discr Math 5 (1979), 287–326.

[15] S. Guha, A. Moss, J.S. Naor, and B. Schieber, Efficient recovery from power outage, Proc 31st Ann ACM Symp Theory Comput (STOC), 1999, pp. 574–582.

[16] V. Guihaire and J.K. Hao, Transit network design and scheduling: A global review, Transportation Res Part A: Policy Practi 42 (2008), 1251–1273.

[17] R.M. Karp, "Reducibility among combinatorial problems," Complexity of computer computations, R.E. Miller and J.W. Thatcher (Editors), Plenum, New York, 1972, pp. 85–103.

[18] M.F. Lai and H.K. Lo, Ferry service network design: Optimal fleet size, routing, and scheduling, Transportation Res Part A: Policy Practi 38 (2004), 305–328.

[19] R.C. Larson, M.D. Metzger, and M.F. Cahn, Responding to emergencies: Lessons learned and the need for analysis, Interfaces 36 (2006), 486–501.

[20] E.E. Lee, J.E. Mitchell, and W.A. Wallace, Restoration of services in interdependent infrastructure systems: A network flows approach, IEEE Trans Syst, Man, Cybernetics, Part C: Appl Reviews 37 (2007), 1303–1317.

[21] E.E. Lee, J.E. Mitchell, and W.A. Wallace, "Network flow approaches for analyzing and managing disruptions to interdependent infrastructure systems," Wiley handbook of science and technology for homeland security, J.G. Voeller (Editor), John Wiley & Sons, Inc. Hoboken, NJ, 2009, Vol. 2, pp. 11419–1428.

[22] M. Liserre, T. Sauter, and J.Y. Hung, Future energy systems: Integrating renewable energy sources into the smart power grid through industrial electronics, IEEE Indust Electronics Magazine 4 (2010), 18–37.

[23] A. Mahmood, M. Aamir, and M.I. Anis, Design and implementation of AMR smart grid system, Proc 2008 IEEE Electrical Power & Energy Conference, Vancouver, BC, 2008, pp. 1–6.

[24] T.C. Matisziw, A.T. Murray, and T.H. Grubesic, Strategic network restoration, Networks Spatial Economics 10 (2010), 345–361.

[25] J.A. Momoh, Smart grid design for efficient and flexible power networks operation and control, Proc 2009 IEEE/PES Power Syst Conference Exposition, Seattle, WA, 2009, pp. 1–8.

[26] K. Moslehi and R. Kumar, A reliability perspective of the smart grid, IEEE Trans Smart Grid 1 (2010), 57–64.

[27] G. Mulder, F. De Ridder, and D. Six, Electricity storage for grid-connected household dwellings with PV panels, Solar Energy 84 (2010), 1284–1293.

[28] S.G. Nurre, B. Cavdaroglu, J.E. Mitchell, T.C. Sharkey, and W.A. Wallace, Restoring infrastructure systems: An integrated network design and scheduling problem, Eur J Oper Res 223 (2012), 794–806.

[29] M.L. Pinedo, Scheduling: Theory, algorithms, and systems, Springer, New York, NY, Fourth edition 2012.

[30] R.T. Wong, A dual ascent approach for Steiner tree problems on a directed graph, Math Program 28 (1984), 271–287.

[31] N. Xu, S.D. Guikema, R.A. Davidson, L.K. Nozick, Z. Çağnan, and K. Vaziri, Optimizing scheduling of post-earthquake electric power restoration tasks, Earthquake Eng & Structural Dynamics 36 (2007), 265–284.

Appendix

The integer programming (IP) formulation of any INDS problem includes network constraints, scheduling constraints, and constraints that link these two sets of decisions. Many of the constraints are similar for the formulations of the different performance metrics; however we present each full IP formulation for the sake of completeness. Table 15 displays the list of all decision variables, their corresponding definitions, and the specific performance metrics and objectives that utilize the decision variables.

Decision Variable	Definition	Problems Used
x_{ij}, x_{ijt}	Continuous, Flow on arc (i,j) (at time t)	MF, MCF, SP
v,v_t	Continuous, Maximum Flow (at time t)	MF
x_{ij}^{ℓ}, x_{ijt}^{ℓ}	Continuous, Flow on arc (i,j) for commodity ℓ (at time t)	MST
ω_{ij},ω_{ijt}	Binary, equals 1 if arc (i,j) is in the MST (at time t)	MST
β_{ijt}	Binary, equals 1 if arc (i,j) is operational at time t	All *Cumulative*
$\alpha_{\mu ijt}$	Binary, equals 1 if arc (i,j) is completed by machine μ at time t	All *Cumulative*
$\bar{T}$	Integer, completion time of last processed task	All $C_{\max}$-*Threshold*
$z_{\mu ij}$	Binary, equals 1 if arc (i,j) is assigned to machine μ	All $C_{\max}$-*Threshold*

Table 15: Decision Variables used in the INDS IP formulations

Maximum Flow INDS IP Formulation

Cumulative We first present the IP formulation of $Pm|MF|Cumulative$. In each time period, we seek to maximize the weighted flow between the source node $\bar{S}$ and sink node S'[7], denoted as $w_t v_t$ where w_t is the weight and v_t is the maximum flow value for time period t. Let x_{ijt} represent the flow on arc (i,j) at time t. Define the binary decision variable β_{ijt} to equal 1 if arc (i,j) at time period t is operational, and 0 otherwise. Decision variable $\alpha_{\mu ijt}$ is defined to equal 1 if machine μ finishes processing arc (i,j) at time t, and 0 otherwise. The *Cumulative* maximum flow IP is then as follows:

$$\max \sum_{t=1}^{T} w_t v_t$$

subject to: (IP)

$$\sum_{(\bar{S},j)\in A_0\cup A_0'} x_{\bar{S}jt} - \sum_{(j,\bar{S})\in A_0\cup A_0'} x_{j\bar{S}t} = v_t \qquad \text{for } t = 1,\ldots,T \tag{4}$$

$$\sum_{(i,j)\in A_0\cup A_0'} x_{ijt} - \sum_{(j,i)\in A_0\cup A_0'} x_{jit} = 0 \qquad \text{for } i \in N\backslash\{\bar{S},S'\},\ t = 1,\ldots,T \tag{5}$$

[7]Through network expansion techniques, a network with multiple sources and sinks can be transformed into a network with a single source and single sink.

$$\sum_{(S',j)\in A_0\cup A_0'} x_{S'jt} - \sum_{(j,S')\in A_0\cup A_0'} x_{jS't} = -v_t \qquad \text{for } t = 1,\ldots,T \tag{6}$$

$$0 \le x_{ijt} \le u_{ij} \qquad \text{for } (i,j)\in A_0,\ t = 1,\ldots,T \tag{7}$$

$$0 \le x_{ijt} \le u_{ij}\beta_{ijt} \qquad \text{for } (i,j)\in A_0',\ t = 1,\ldots,T \tag{8}$$

$$\sum_{(i,j)\in A_0'} \sum_{s=t}^{\min\{T,t+p_{ij}-1\}} \alpha_{\mu ijs} \le 1 \qquad \text{for } \mu = 1,\ldots,m,\ t = 1,\ldots,T \tag{9}$$

$$\beta_{ijt} - \beta_{ijt-1} - \sum_{s=1}^{t}\sum_{\mu=1}^{m} \alpha_{\mu ijt} = 0 \qquad \text{for } (i,j)\in A_0',\ t = 1,\ldots,T \tag{10}$$

$$\sum_{t=1}^{p_{ij}-1} \beta_{ijt} = 0 \qquad \text{for } (i,j)\in A_0' \tag{11}$$

$$\sum_{\mu=1}^{m}\sum_{t=1}^{p_{ij}-1} \alpha_{\mu ijt} = 0 \qquad \text{for } (i,j)\in A_0' \tag{12}$$

$$\alpha_{\mu ijt}, \beta_{ijt} \in \{0,1\} \qquad \text{for } (i,j)\in A_0',\ \mu = 1,\ldots,m,\ t = 1,\ldots,T. \tag{13}$$

Constraints (4) - (6) are standard flow balance constraints and constraints (7) are flow capacity constraints. The combination of these four constraints (constraints (4) - (7)) represent the network flow constraints. Constraints (9) - (13) are the *Cumulative* scheduling constraints. Constraints (9) ensure that machine is not processing more than one arc at a time. Constraints (10) transition an arc from non-operational to operational once it has been processed. Constraints (11) and (12) ensure that an arc cannot become operational until enough time has passed to allow for its processing, which improves the quality of the linear programming relaxation. Constraints (8) link the scheduling and network flow decisions by ensuring that flow can only occur on a non-operational arc if it has been processed by a machine.

$C_{\max}$-Threshold We now present the IP formulation of $Pm|MF|C_{\max} - Threshold$. We define x_{ij} to be the flow on arc (i,j), binary $z_{\mu ij}$ to equal 1 if arc (i,j) is processed by machine μ, and $\bar{T}$ to be the minimum time needed to allow the network to reach or exceed the input desired performance value P. The IP formulation is

$$\min\ \bar{T}$$

subject to: (IP)

$$\sum_{(\bar{S},j)\in A_0\cup A_0'} x_{\bar{S}j} - \sum_{(j,\bar{S})\in A_0\cup A_0'} x_{j\bar{S}} = v \tag{14}$$

$$\sum_{(i,j)\in A_0\cup A_0'} x_{ij} - \sum_{(j,i)\in A_0\cup A_0'} x_{ji} = 0 \qquad \text{for } i \in N\backslash\{\bar{S},S'\} \tag{15}$$

$$\sum_{(S',j)\in A_0\cup A_0'} x_{S'j} - \sum_{(j,S')\in A_0\cup A_0'} x_{jS'} = -v \tag{16}$$

$$0 \le x_{ij} \le u_{ij} \qquad \text{for } (i,j)\in A_0 \tag{17}$$

$$0 \le x_{ij} \le u_{ij}\sum_{\mu=1}^{m} z_{\mu ij} \qquad \text{for } (i,j)\in A_0' \tag{18}$$

$$v \ge P \tag{19}$$

$$\sum_{(i,j)\in A_0'} p_{ij} z_{\mu ij} \le \bar{T} \qquad \text{for } \mu = 1,\ldots,m \tag{20}$$

$$\sum_{\mu=1}^{m} z_{\mu ij} \le 1 \qquad \text{for } (i,j)\in A_0' \tag{21}$$

$$z_{\mu ij} \in \{0,1\} \qquad \text{for } (i,j)\in A_0', \mu = 1,\ldots,m. \tag{22}$$

Constraints (14)-(17) are the network flow constraints, which include flow balance and flow capacity constraints. Constraints (19)-(21) represent the $C_{\max}$-*Threshold* scheduling constraints and ensure that the maximum flow meets the desired performance value, $\bar{T}$ represents the largest time a machine is processing a task, and that only one machine processes each non-operational arc. Constraints (18) link the network flow and scheduling constraints by only allowing flow on a non-operational arc that has been processed by a machine.

Minimum Cost Flow INDS IP Formulation

Cumulative The IP formulation of $Pm|MCF|Cumulative$ is quite similar to the formulation of $Pm|MF|Cumulative$. In fact, all necessary variables in this IP formulation have been previously discussed (see Table 15 for a summary) and many constraints are similar. The IP formulation of this problem is

$$\min \sum_{t=1}^{T} w_t \sum_{(i,j)\in A_0\cup A_0'} c_{ij} x_{ijt}$$

subject to: (IP)

$$\sum_{(i,j)\in A_0\cup A_0'} x_{ijt} - \sum_{(j,i)\in A_0\cup A_0'} x_{jit} = s_i \qquad \text{for } i\in S,\ t = 1,\ldots,T \tag{23}$$

$$\sum_{(i,j)\in A_0\cup A_0'} x_{ijt} - \sum_{(j,i)\in A_0\cup A_0'} x_{jit} = 0 \qquad \text{for } i\in N\backslash\{S\cup D\},\ t = 1,\ldots,T \tag{24}$$

$$\sum_{(i,j)\in A_0\cup A_0'} x_{ijt} - \sum_{(j,i)\in A_0\cup A_0'} x_{jit} = -d_{it} \qquad \text{for } i\in D,\ t = 1,\ldots,T \tag{25}$$

$$0 \le x_{ijt} \le u_{ij} \qquad \text{for } (i,j)\in A_0,\ t = 1,\ldots,T \tag{26}$$

$$0 \leq x_{ijt} \leq u_{ij}\beta_{ijt} \qquad \text{for } (i,j) \in A_0',\ t = 1,\ldots,T \tag{27}$$

$$\sum_{(i,j)\in A_0'} \sum_{s=t}^{\min\{T,t+p_{ij}-1\}} \alpha_{\mu ijs} \leq 1 \qquad \text{for } \mu = 1,\ldots,m,\ t = 1,\ldots,T \tag{28}$$

$$\beta_{ijt} - \beta_{ijt-1} - \sum_{s=1}^{t}\sum_{\mu=1}^{m} \alpha_{\mu ijt} = 0 \qquad \text{for } (i,j) \in A_0',\ t = 1,\ldots,T \tag{29}$$

$$\sum_{t=1}^{p_{ij}-1} \beta_{ijt} = 0 \qquad \text{for } (i,j) \in A_0' \tag{30}$$

$$\sum_{\mu=1}^{m}\sum_{t=1}^{p_{ij}-1} \alpha_{\mu ijt} = 0 \qquad \text{for } (i,j) \in A_0' \tag{31}$$

$$\alpha_{\mu ijt}, \beta_{ijt} \in \{0,1\} \qquad \text{for } (i,j) \in A_0',\ \mu = 1,\ldots,m,\ t = 1,\ldots,T. \tag{32}$$

The objective function seeks to minimize the weighted cost of flow at each time period over the set time horizon T. Constraints (23) - (25) represent the traditional flow balance constraints for supply, transshipment, and demand nodes and constraints (26) are the capacity constraints of operational arcs. Constraints (27)-(32) are identical to the scheduling and linking constraints (8)-(13) found in the maximum flow *Cumulative* IP formulation.

$C_{\max}$-Threshold The IP formulation of $Pm|MCF|C_{\max} - Threshold$ is very similar to the IP formulation of $Pm|MF|C_{\max} - Threshold$ with the main difference being that P represents a threshold that bounds above the cost of the flow. This formulation is

$$\min\ \bar{T}$$

subject to: (IP)

$$\sum_{(i,j)\in A_0\cup A_0'} x_{ij} - \sum_{(j,i)\in A_0\cup A_0'} x_{ji} = s_i \qquad \text{for } i \in S \tag{33}$$

$$\sum_{(i,j)\in A_0\cup A_0'} x_{ij} - \sum_{(j,i)\in A_0\cup A_0'} x_{ji} = 0 \qquad \text{for } i \in N\backslash\{S \cup D\} \tag{34}$$

$$\sum_{(i,j)\in A_0\cup A_0'} x_{ij} - \sum_{(j,i)\in A_0\cup A_0'} x_{ji} = -d_i \qquad \text{for } i \in D \tag{35}$$

$$0 \leq x_{ij} \leq u_{ij} \qquad \text{for } (i,j) \in A_0 \tag{36}$$

$$0 \leq x_{ij} \leq u_{ij}\sum_{\mu=1}^{m} z_{\mu ij} \qquad \text{for } (i,j) \in A_0' \tag{37}$$

$$\sum_{(i,j)\in A_0\cup A_0'} c_{ij}x_{ij} \leq P \tag{38}$$

$$\sum_{(i,j)\in A_0'} p_{ij} z_{\mu ij} \leq \bar{T} \qquad \text{for } \mu = 1, \ldots, m \tag{39}$$

$$\sum_{\mu=1}^{m} z_{\mu ij} \leq 1 \qquad \text{for } (i,j) \in A_0' \tag{40}$$

$$z_{\mu ij} \in \{0,1\} \qquad \text{for } (i,j) \in A_0', \mu = 1, \ldots, m. \tag{41}$$

The objective function seeks to find the minimum time $\bar{T}$ that satisfies the constraints. Constraints (33)-(35) enforce the traditional flow balance constraints. Constraints (37)-(41) are almost identical to the scheduling and linking constraints constraints (18)-(22), found in the maximum flow $C_{\max}$-*Threshold* IP formulation.

Shortest Path INDS IP Formulation

Cumulative* and $C_{\max}$-*Threshold As described in Section 4.3, the shortest path problem can be modeled as a minimum cost flow problem. Therefore, the shortest path INDS IP formulations are identical to the minimum cost flow INDS IP formulations. The only alteration is that the capacity of the arcs can be reduced to the number of demand nodes $|D|$, since each demand node has one unit of demand.

Minimum Spanning Tree INDS IP Formulation

Cumulative The IP formulation for $Pm|MST|Cumulative$ is similar to the other IP formulations of the *Cumulative* objective with the exception that capturing the network performance becomes more difficult. Due to the absence of a compact formulation for the minimum spanning tree problem, we have adopted a multi-commodity flow problem as used by Wong [30]. This formulation has $|N| - 1$ commodities. Node 0 will be the supply node for all commodities and then every other node has a demand equal to 1 for their own commodity. The integer decision variable x_{ijt}^{ℓ} represents the flow on the undirected arc (i,j) of the ℓ-th commodity at time t and ranges between -1 and 1, where a negative flow corresponds to sending flow from j to i. The binary decision variable ω_{ijt} is equal to 1 if arc (i,j) is in the MST at time t. The *Cumulative* minimum spanning tree INDS IP formulation is then as follows:

$$\min \sum_{t=1}^{T} w_t \sum_{(i,j)\in A_0\cup A_0'} c_{ij}\omega_{ijt}$$

subject to: (IP)

$$\sum_{(i,j)\in A_0\cup A_0'} x_{ijt}^{\ell} - \sum_{(j,i)\in A_0\cup A_0'} x_{jit}^{\ell} = 1 \qquad \text{for } i = 0,\ t = 1, \ldots, T,\ \ell = 1, \ldots, |N| - 1 \tag{42}$$

$$\sum_{(i,j)\in A_0\cup A_0'} x_{ijt}^{\ell} - \sum_{(j,i)\in A_0\cup A_0'} x_{jit}^{\ell} = 0 \qquad \text{for } i \in N\backslash\{0\},\, t=1,\ldots,T,\, \ell=1,\ldots,|N|-1, \ell \neq i \quad (43)$$

$$\sum_{(i,j)\in A_0\cup A_0'} x_{ijt}^{\ell} - \sum_{(j,i)\in A_0\cup A_0'} x_{jit}^{\ell} = -1 \qquad \text{for } i \in N\backslash\{0\},\, t=1,\ldots,T,\, \ell=1,\ldots,|N|-1, \ell = i \quad (44)$$

$$x_{ijt}^{\ell} \leq \omega_{ijt} \qquad \text{for } (i,j) \in A_0 \cup A_0',\, t=1,\ldots,T,\, \ell=1,\ldots,|N|-1 \quad (45)$$

$$-x_{ijt}^{\ell} \leq \omega_{ijt} \qquad \text{for } (i,j) \in A_0 \cup A_0',\, t=1,\ldots,T,\, \ell=1,\ldots,|N|-1 \quad (46)$$

$$\omega_{ijt} \leq \beta_{ijt} \qquad \text{for } (i,j) \in A_0 \cup A_0',\, t=1,\ldots,T, \quad (47)$$

$$\sum_{(i,j)\in A_0'} \sum_{s=t}^{\min\{T,t+p_{ij}-1\}} \alpha_{\mu ijs} \leq 1 \qquad \text{for } \mu=1,\ldots,m,\, t=1,\ldots,T \quad (48)$$

$$\beta_{ijt} - \beta_{ijt-1} - \sum_{s=1}^{t}\sum_{\mu=1}^{m} \alpha_{\mu ijt} = 0 \qquad \text{for } (i,j) \in A_0',\, t=1,\ldots,T \quad (49)$$

$$\sum_{t=1}^{p_{ij}-1} \beta_{ijt} = 0 \qquad \text{for } (i,j) \in A_0' \quad (50)$$

$$\sum_{\mu=1}^{m}\sum_{t=1}^{p_{ij}-1} \alpha_{\mu ijt} = 0 \qquad \text{for } (i,j) \in A_0' \quad (51)$$

$$\alpha_{\mu ijt}, \beta_{ijt} \in \{0,1\} \qquad \text{for } (i,j) \in A_0',\, \mu=1,\ldots,m,\, t=1,\ldots,T \quad (52)$$

$$\omega_{ijt} \in \{0,1\} \qquad \text{for } (i,j) \in A_0 \cup A_0', t=1,\ldots,T. \quad (53)$$

Constraints (42)-(44) represent the multi-commodity flow balance constraints. Constraints (45) and (46) constrain the flow on an arc by the decision whether it is in the MST at this time period. Constraints (47) link the scheduling decisions with the MST decisions, i.e., the arc (i,j) must be operational ($\beta_{ijt}=1$) if it is selected for inclusion in the MST. Constraints (48) - (52) represent the *Cumulative* scheduling constraints.

$C_{\max}$-Threshold The IP formulation of $Pm|MST|C_{\max} - Threshold$ will not require the variables representing the network performance to be indexed by t, i.e., we have a single set of MST decision variables. Recalling that $z_{\mu ij}$ equals 1 if arc (i,j) is assigned to machine μ, the IP formulation is

$$\min\ \bar{T}$$

subject to: (IP)

$$\sum_{(i,j)\in A_0\cup A_0'} x_{ij}^{\ell} - \sum_{(j,i)\in A_0\cup A_0'} x_{ji}^{\ell} = 1 \qquad \text{for } i=0,\, \ell=1,\ldots,|N|-1 \quad (54)$$

$$\sum_{(i,j)\in A_0\cup A_0'} x_{ij}^{\ell} - \sum_{(j,i)\in A_0\cup A_0'} x_{ji}^{\ell} = 0 \qquad \text{for } i \in N\backslash\{0\},\, \ell=1,\ldots,|N|-1, \ell \neq i \quad (55)$$

$$\sum_{(i,j)\in A_0\cup A_0'} x_{ij}^{\ell} - \sum_{(j,i)\in A_0\cup A_0'} x_{ji}^{\ell} = -1 \qquad \text{for } i \in N\backslash\{0\},\ \ell = 1,\ldots,|N|-1, \ell = i \tag{56}$$

$$x_{ij}^{\ell} \leq \omega_{ij} \qquad \text{for } (i,j) \in A_0 \cup A_0',\ \ell = 1,\ldots,|N|-1 \tag{57}$$

$$-x_{ij}^{\ell} \leq \omega_{ij} \qquad \text{for } (i,j) \in A_0 \cup A_0',\ \ell = 1,\ldots,|N|-1 \tag{58}$$

$$\omega_{ij} \leq \sum_{\mu=1}^{m} z_{\mu ij} \qquad \text{for } (i,j) \in A_0 \cup A_0' \tag{59}$$

$$\sum_{(i,j)\in A_0\cup A_0'} c_{ij}\omega_{ij} \leq P \tag{60}$$

$$\sum_{(i,j)\in A_0'} p_{ij} z_{\mu ij} \leq \bar{T} \qquad \text{for } \mu = 1,\ldots,m \tag{61}$$

$$\sum_{\mu=1}^{m} z_{\mu ij} \leq 1 \qquad \text{for } (i,j) \in A_0' \tag{62}$$

$$z_{\mu ij} \in \{0,1\} \qquad \text{for } (i,j) \in A_0', \mu = 1,\ldots,m. \tag{63}$$

$$\omega_{ij} \in \{0,1\} \qquad \text{for } (i,j) \in A_0 \cup A_0' \tag{64}$$

The network constraints (54)-(58) representing the MST decisions remain the same as in the *Cumulative* formulations except for the index t. The linking constraints (59) state that an arc cannot be in the MST unless it has been assigned to a machine for processing. The constraint (60) ensures that the MST value remains below the desired threshold performance value. Constraints (61)-(63) are the $C_{\max}$-*Threshold* scheduling constraints.