The Dissertation Committee for Abigail June Crocker
certifies that this is the approved version of the following dissertation:

Post Disaster Transportation Network Recovery

Committee:

Stephen D. Boyles, Supervisor

Anantaram Balakrishnan

John J. Hasenbein

Erhan Kutanoglu

Post Disaster Transportation Network Recovery

by

Abigail June Crocker, B.S., M.S.

Dissertation

Presented to the Faculty of the Graduate School of

The University of Texas at Austin

in Partial Fulfillment

of the Requirements

for the Degree of

Doctor of Philosophy

The University of Texas at Austin

May 2024

Acknowledgments

I have been extremely fortunate to be supported and mentored by a number of outstanding advisors during my graduate studies at The University of Texas at Austin. First and foremost, I would like to thank Dr. Stephen Boyles, whose continual advice, mentorship, and editing have enabled this dissertation to reach its present form, and whose courses on transportation networks helped set the stage for my graduate studies and research. Thank you, also, to my committee members, Dr. Anant Balakrishnan, Dr. John Hasenbein, and Dr. Erhan Kutanoglu, for their advice and insights that pushed me to continually explore new ideas and new ways of looking at problems. I have been doubly fortunate to enjoy the camaraderie of both my fellow ORIE students and my fellow students in the SPARTA Lab. Graduate studies would not have been nearly as rewarding without your collaboration. Thank you to my husband, Matt, for his continual support and unwavering belief in me, inspiring me to always push to excel, and to my parents for their continual support and encouragement. Finally, I would like to thank Dr. Paul Mlakar, COL Patrick Sullivan, COL Matthew Rogers (Ret.), and MAJ David Picard for supporting this journey from the start and for their advice and mentorship.

The views expressed herein are those of the author and do not necessarily reflect the position of the Department of the Army or the Department of Defense.

Abstract

Post Disaster Transportation Network Recovery

Abigail June Crocker, PhD
The University of Texas at Austin, 2024

SUPERVISOR: Stephen D. Boyles

Transportation network recovery after an extreme hazard or natural disaster is time-sensitive, resource-intensive, and often involves repairing hundreds or thousands of damaged links. Furthermore, optimal sequencing or scheduling of those hundreds or thousands of links for repair is extremely computationally intensive, and intractable by exact solution methods for over about twenty broken links. Previous methods which search the solution space and incorporate travel time into the objective have not been demonstrated on instances exceeding fifty broken links. In this dissertation, I focus on road networks and examine both the single-crew sequencing problem and the multi-crew scheduling problem, greatly expanding the number of broken links which can be handled within a reasonable planning time period. Specifically, my methodology handles up to about 250 broken links in 72 hours for the single-crew problem and up to about 170 broken links in the same time frame for the multi-crew problem, both on the Anaheim and Berlin-Mitte-Center test networks.

For the single-crew problem, I define two problem formulations with the objective of minimizing total travel delay over the repair horizon and explore resulting insights on complexity. Additionally, I define both a lower bound using free flow travel times and a simpler heuristic lower bound. I parameterize a simulated annealing heuristic capable of generating high quality solutions in less than 24 hours

for problems with up to 175–185 broken links on the Anaheim and Berlin-Mitte-Center test networks. Finally, I compare my simulated annealing algorithm and selected heuristic solution methods on the two test networks with 8–48 broken links and quantify trade-offs in terms of solution quality and solution time. These experiments allow for practical recommendations of solution method given instance size, network characteristics, and computational time available.

I extend the formulation to allow for multiple identical work crews, maintaining the objective of minimizing total travel delay over the repair horizon. I establish that this problem is NP-hard by proving that minimizing weighted completion time on identical parallel machines reduces to the multi-crew scheduling problem. I compare heuristic methods, borrowing from machine scheduling literature as well as previously proposed methods for the special case of sequencing repairs for a single work crew as opposed to for multiple crews. Additionally, I propose a simulated annealing parameterization and compare multiple neighborhood definitions and combinations. The resulting simulated annealing algorithm outperforms each other method tested in terms of solution quality, handling up to about 120–130 broken links on the Anaheim and Berlin Mitte-Center networks in 24 hours.

Finally, I develop two decomposition methods which first assign broken links to work crews, and then sequence links for repair within those crews using one of four methods. The first decomposition method is based on an approximation of minimal makespan, and the second is a novel method based on characterising asymmetric interaction coefficients between broken links. These decomposition methods, however, do not achieve sufficient solution quality to recommend their use over simpler sequencing methods. The developed simulated annealing heuristic maintains the highest solution accuracy by a significant margin, though the greedy methods provide quicker solutions.

Table of Contents

List of Tables

List of Figures

Chapter 1: Introduction

When transportation systems are damaged by extreme hazards or natural disasters such as hurricanes, seismic events, floods, or terrorist attacks, efficient long-term recovery actions are essential to re-establishing service. Transportation system damage in these scenarios can affect hundreds or even thousands of links, as evidenced in reports by Zhuang et al. (2009) on the 2008 earthquake in Wenchuan, China, affecting 1,657 bridges and by Lunderville (2012) on over 300 bridges and 2,000 road segments impacted by Hurricane Irene in Vermont. The network degradation after an extreme event can result in elevated congestion on functional routes, economic losses due to lost demand, or even fully disconnected portions of the network with implications for emergency services accessibility. For these reasons, efficiency is key in recovering the network to a fully operational state, highlighting a critical balance between solution time and quality in long-term recovery planning. I have not encountered a previous study which both incorporates travel time in the objective function and utilizes search methods rather than greedy heuristic orderings on instances exceeding fifty broken links. In this dissertation, I examine road networks in particular, and parameterize a search heuristic which handles up to about 250 broken links in 72 hours for the single-crew problem and up to about 170 broken links in the same time frame for the multi-crew problem, both on the Anaheim and Berlin-Mitte-Center test networks.

1.1 Resilience and Recovery

The term resilience originally stems from physics where it describes the capability of a material to regain its original shape after deformation, typically due to compressive stress (Merriam-Webster). However, this concept is applied across widely varied fields with similar but distinct definitions. Specific to transportation,

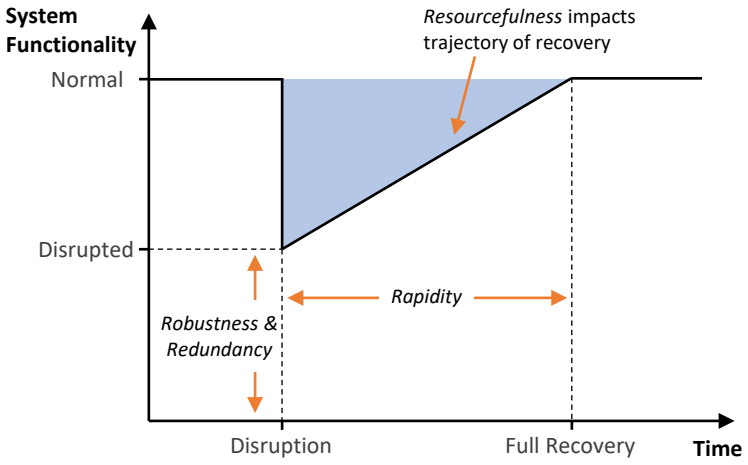

Figure 1.1: Resiliency triangle

Zhou et al. (2019) conducted a structured literature review in 2019 of 101 papers from 2006 to 2017 dealing with resilience in transportation. Their results indicated two key perspectives on resilience in transportation: "the ability to maintain functionality under disruptions" and "time and resources required to restore performance level after disruptions" (Zhou et al., 2019). In earthquake engineering, Bruneau et al. (2003) defines resiliency in three parts as the ability of the system to reduce the probability of a disruption, mitigate the initial loss of functionality due to a disruption, and rapidly return to full functionality after a disruption.

Bruneau et al. (2003) also first introduced the resiliency triangle to quantitatively describe the impact of a disruption and subsequent recovery efforts on a system. Figure 1.1 graphically depicts the lost functionality over time due to a disruption. The initial drop in capability reflects system robustness and redundancy within the system. Rapidity describes the time between the initial disruption and returning to full pre-disruption functionality. The trajectory of recovery – the two dimensional path from decreased functionality immediately post-disruption to fully restored functionality – is determined by resourcefulness including the timely availability of recovery assets and organizational effectiveness to employ them. A more complex graph for many extreme events such as hurricanes would additionally depict the initial drop

in system functionality over some initial time period, rather than an instantaneous event. However, for the purposes of evaluating long-term recovery efforts, I will focus on the time period between full disruption – after the extreme event and immediate clean-up efforts – and full recovery.

Operationally in terms of a roadway network, Zhang et al. (2018b) categorizes comprehensive resiliency in terms of mitigation strategies, emergency response, and long-term recovery. Mitigation strategies encompass actions taken prior to a disruption to increase one or more components of resiliency. These actions seek to harden the network against potential disruptions and increase resiliency through robustness of individual components and redundancy within the network. In terms of the functionality of a transportation network, emergency response might including clearing obstructions such as power lines, fallen trees, and debris, or even patching minor damage to restore immediate functionality. For those network links which cannot be rapidly restored, long-term recovery actions are necessary. Long term recovery actions might include rebuilding a washed-out road, or repairing a bridge or road segment which is structurally unstable and unusable due to an earthquake.

Because there is no general consensus of how to define and break down phases of resiliency and recovery, in this dissertation I distinguish between emergency response and long-term recovery based on the durations involved and the typical objectives to be met. With this framework, though emergency response may extend for weeks or even months in severe scenarios, the focus is still on immediate connectivity of the network and basic functionality of key links (even if still degraded), whereas long-term recovery seeks to return the network to full functionality. In order to model the two phases independently, I structure the problem such that emergency response must be complete before long-term recovery begins. In some cases, the operational strategy can be brought full circle, where mitigation strategies are implemented in tandem with long-term recovery, and certain damaged network components are repaired to a higher standard than their condition prior to disruption and/or the placement of new links does not exactly correspond to the locations of the destroyed links. In

the following chapters, I focus on long-term recovery of road networks to their pre-disruption state, leaving the integration of emergency response, long-term recovery, and future mitigation strategies to future research.

1.2 Framework and Assumptions

In the framework developed by Bruneau et al. (2003), my efforts focus on minimizing the cumulative loss of system functionality, as represented by the shaded triangle in Figure 1.1, from a post-disruption point of view. I differentiate between immediate actions taken to clear roads which are temporarily impeded or constricted and long-term actions taken to restore roads which are damaged such that they are unusable without major repairs or reconstruction. Cheng and Zhang (2022) address the scenario of short-duration road repair where repair times are on the order of 1–12 hours and therefore travel time and routing given damaged links are integrated into the model. In contrast, I focus on long-term recovery where durations are measured in days, and full recovery requires months. I set up the problem such that a subset of network links are initially broken and choose to minimize total travel delay over the repair horizon, with the static traffic assignment problem (TAP) as a lower level problem. I use the user equilibrium formulation, which assumes that all drivers have perfect knowledge of system conditions and seek to minimize their own travel time. Given these assumptions on driver behavior, at user equilibrium, for each origin-destination pair every used path from that origin to that destination has equal and minimal travel time (Wardrop, 1952). Stated differently, no single driver could achieve a lower travel time by shifting routes. In Chapter 2, I focus on sequencing the repair of broken links for a single repair crew. In Chapter 3, I study the scheduling of broken links among multiple identical repair crews. In both chapters, I employ the following assumptions:

- Broken links are unusable until fully repaired.
- Repair times for each link are fixed and known.

- User equilibrium is reached after each repair.
- Repair crews cannot be subdivided.
- Preemption (pausing a link repair to shift to another link before returning to the first link) is not permitted.

In Chapter 3 I additionally assume that:

- Repair crews are identical.
- Each broken link is repaired by a single repair crew.

The chosen problem structure, along with the first general assumption above, results in binary repair states – either a link is fully broken and unusable or at full operational capacity. While this structure could be extended in future research to allow partial service restoration, or even service upgrade beyond original capacity, both extensions would result in a drastic increase in solution space over the current formulation. Therefore, for this research I restrict links to binary repair states. The use of static TAP as the lower level problem in both the single- and multi-crew formulations is contingent on the third general assumption above, that user equilibrium is reached after each repair, and furthermore that equilibrium is reached in a negligible amount of time in comparison to repair durations. Similarly, I treat transitions between repairs as instantaneous, with set up times included in repair durations and not dependent on previous repairs or repair order. These assumptions together represent simplifications of the repair and traffic routing processes, but are reasonable for the time scale under study and useful for examining the underlying problem structure, which may be extended in future research to study the impacts of relaxing these assumptions.

In terms of structure, the resulting network recovery sequencing and scheduling problems share characteristics with scheduling road maintenance and repairs and

expanding an existing road network. In the case of routine or periodic road maintenance and repair, rather than being unusable, links may have lower capacity prior to maintenance or repair and either return to design capacity or improve upon original capacity upon completion. In the case of network expansion, if the set of new links is already fixed, and their repair durations are known and fixed, the links could simply be considered broken until constructed, and methods presented in Chapters 2 and 3 could be applied as appropriate.

1.3 Overview and Contributions

In Chapters 2 and 3, I specifically address balancing the trade-off between solution time and quality when determining which method is "best" for a particular application for the single-crew sequencing problem and the multi-crew scheduling problem, respectively. I develop a simulated annealing method which delivers solutions of comparable quality to those found by a previously proposed beam search (Gokalp et al., 2021) for single-crew instances while achieving much lower run times. I adapt the simulated annealing heuristic for multi-crew problems, maintaining comparable accuracy to the single-crew formulation, while the beam search accuracy degrades due to the change in underlying structure. Additionally, I benchmark proposed solution methods from the literature on equivalent problem instances on the Anaheim and Berlin-Mitte-Center (BMC) networks from the Transportation Networks for Research repository (2022) with 8–100 broken links, and 10–100 instances for each tested parameter combination. The code used for the experiments in this dissertation is available on GitHub (Gokalp and Crocker, 2024).

In Chapter 2, examining the sequencing problem, I compare a shortest path method to find the exact optimal solution, a beam search heuristic, my simulated annealing heuristic, and six greedy heuristics, using instances with 8–48 broken links. The bidirectional beam search provides the highest quality solutions in the majority of single-crew experiments, but the simulated annealing heuristic runs significantly

faster, with minimal to no loss in solution quality. Only the greedy heuristics using Rey and Bar-Gera's (2020) approximation method reliably come within a 50% accuracy gap from the beam search or simulated annealing objective function values. While those methods can more rapidly obtain a reasonable repair sequence, beam search and simulated annealing both obtain much higher quality solutions and can solve instances with up to 55 and 250–260 broken links, respectively, in under 72 hours. Additionally, I explore the extension of the described methods to scenarios with multi-class demand. This extension allows relative weighting of critical logistic movements versus residential and commercial traffic, versus emergency services traffic within the damaged network. My primary contributions in Chapter 2 are: 1) a single-crew simulated annealing heuristic; 2) a comparison of solution methods by run time and solution quality; 3) methods of obtaining lower bounds on objective value; 4) recommendations on appropriate greedy methods for use when severely time-constrained.

In Chapter 3, I extend the problem formulation to allow for multiple identical work crews, rather than a single work crew, increasing applicability and realism. First, I place the multi-crew problem within the class of NP-hard problems, demonstrating that a restricted version of the multi-crew problem is equivalent to minimizing weighted completion time on identical parallel machines from the general scheduling literature. I then adapt the simulated annealing heuristic developed for the sequencing problem for the scheduling problem, additionally assessing multiple novel neighborhood definitions, and adapt the brute force and greedy methods to account for repairs completing in parallel. Additionally, I develop two methods of decomposing links into work crew assignments and subsequently sequencing links for repair within those work crews. The first decomposition method is based on a four-thirds greedy approximation of the minimal makespan assignment, and the second method uses novel asymmetric interaction coefficients between broken links derived from the impact on other link flows of breaking each link. The simulated annealing method consistently achieves the highest solution quality of the methods tested, and

can process up to about 120–130 broken links in 24 hours or 170–180 broken links in 72 hours on the Anaheim or BMC networks. My primary contributions in Chapter 3 are: 1) establishing the multi-crew scheduling problem as NP-hard; 2) a multi-crew simulated annealing heuristic; 3) exploration of the effect of neighborhood definition on simulated annealing performance; 4) adapted sequencing methods for scheduling application; 5) two methods of decomposition into crews and four methods of sequencing links for repair within each crew; 6) comparison of developed methods by run time and solution quality; 7) recommendations on appropriate greedy methods for use when severely time-constrained.

Finally, in Chapter 4, I summarize this dissertation's contributions as well as detailing several potential directions for future research. Additionally, I discuss in greater depth the implications of the set of assumptions used to frame the problem at hand and describe potential relaxations of particular assumptions for future research.

Chapter 2: Balancing Solution Time and Quality in Disaster Recovery Sequencing

2.1 Introduction

Natural disasters significantly disrupt road networks, and sequencing link repairs exactly optimally is intractable for even moderate numbers of damaged links. In this section, I present a simulated annealing heuristic which can offer high quality solutions in less than 24 hours for problems with up to 165 and 175 broken links on the Anaheim and BMC test networks, respectively, corresponding to 18–20% of network links. I generate 1800 random problem instances over the two test networks with 8–48 broken links to compare optimal sequencing, the bidirectional beam search heuristic, simulated annealing heuristic, and six greedy heuristics in order to quantify trade-offs in terms of solution quality and solution time.

2.2 Literature Review

This chapter focuses on methods of balancing run time and solution quality when sequencing the repair of links in a damaged transportation network, using TSTT to represent the functionality of any network state. Multiple authors have looked at this general formulation of the problem and proposed heuristics for finding the optimal repair sequence (Rey and Bar-Gera, 2020; Gokalp et al., 2021), but I have not encountered a review which focuses on comparing available proposed methods on equivalent problem instances and enunciating the trade-offs involved.

Additionally, many authors have studied similar problems with objective functions capturing some combination of functional, topological, and economic performance metrics, often on small-scale case studies. Zhang et al. (2017) measure performance by the average number of reliable independent pathways. Chen and Miller-Hooks (2012) quantify performance by the amount of demand met in an intermodal

freight network, identifying recovery actions to be taken with a stochastic mixed-integer program. Merschman et al. (2020) and Zhang et al. (2018a) combine performance metrics from multiple categories to seek an optimal repair sequence. Several authors use performance metrics related to TSTT, but as part of multi-stage frameworks with varied solution methods (Bocchini and Frangopol, 2012; Vugrin et al., 2014; Ye and Ukkusuri, 2015).

Bocchini and Frangopol (2012) model capacity restoration as a time-continuous process in that the repair rate for a link is determined by the level of funding applied, with multiple link repairs able to proceed in sequence. They then use a genetic algorithm to solve the resultant multi-objective bilevel optimization model to maximize total network resilience, minimize time to reach a certain level of network functionality, and minimize present cost of repairs. The application focus is a network with damaged bridges and specific detours prescribed for each damaged bridge. Vugrin et al. (2014) take into account the cost of recovery operations as well as system impact measured by TSTT, solving their bilevel optimization model using a modified simulated annealing algorithm. They demonstrate their method on a network with nine nodes and 30 links, four of which are designated as broken, but with eight recovery tasks for each broken link, and varying levels of service during recovery.

Ye and Ukkusuri (2015) seek to maximize system resilience, defined as the sum of the recovery ratios of system performance during reconstruction. In contrast to other papers, they use day-to-day traffic assignments, using a logit-based loading model with fixed demand, incorporating drivers' learning rates and perceived costs for each path to model traffic flow evolution during reconstruction. The authors solve this model using tabu search, with the largest network being a modification of the Sioux Falls test network with 24 nodes, only two origin destination pairs, and only 36 links, six of which are designated as broken. Extension of this method to larger networks, or even the unmodified Sioux Falls test network, would require overcoming the challenge of enumerating path flows.

For this study, I choose to use the single performance measure of TSTT, and focus on algorithmic contributions to balance solution time and quality, which may in turn also be useful for solving similar or extended formulations. The performance gains with minimal accuracy loss achieved by the bidirectional beam search heuristic over the exact enumerative approach (within the range where enumeration is tenable) inspired the desire to further explore high quality heuristic solution methods. I establish that the single-crew sequencing problem can be formulated as a shortest path problem from the immediately post-disruption state to the fully repaired state, with the other network nodes representing intermediate repair states and links representing feasible transitions between repair states. This representation allows for some discussion of computational complexity, given the most efficient known shortest path algorithms. For my results, given the structure of the induced network, I rely on the computational complexity for finding the shortest path in a directed acyclic network, as presented by Ahuja et al. (1993). Fully defining this induced network, however, would require computing the state transition (link) costs by solving static TAP for each potential repair state. Indeed, the beam search presented by Gokalp et al. (2021) explores this induced network from both the source and the destination, using estimates of link costs and exploring the most promising branches, rather than computing all necessary link costs *a priori*.

Static TAP is well-researched and efficient algorithms have been developed for finding the solution given a network and demand matrix. TAP was first formulated as a convex program by Beckmann et al. (1956). The first two classes of algorithms developed for solving TAP were link based and path based methods. Link based methods directly reflect the convex nature of Beckmann's formulation by identifying a search direction and then a step size at each iteration until some convergence criterion is met. Examples of link based methods include MSA (Powell and Sheffi, 1982), Frank-Wolfe (Frank and Wolfe, 1956), and Conjugate Frank-Wolfe (Mitradjieva and Lindberg, 2013). These methods are memory efficient, but slow to converge in large networks. In contrast, path based methods are memory intensive, since they store

the set of used paths, but converge much faster than link based methods. Two well-known examples of path based methods are gradient projection (Jayakrishnan et al., 1994), wherein each step is taken in the direction of steepest descent and the new solution is projected onto the set of feasible flows, and projected gradient (Florian et al., 2009), wherein the direction of steepest descent is projected onto the set of feasible flows and then a step is taken in the projected direction.

More recently, bush based methods have been developed to efficiently store path flows while optimizing computational time. For computational experiments, I use Algorithm B, proposed by Dial (2006) and implemented by Boyles (2022). This method stores current flows in a connected and acyclic graph from each zone, and shifts flows from the longest to shortest path within each bush, iterating over the bushes until convergence. Other bush based methods include origin based assignment (OBA) and traffic assignment by paired alternative segments (TAPAS). OBA, authored by Bar-Gera (2002), allows for simultaneous shifts involving many paths. TAPAS is a more recent bush based algorithm which finds path flows satisfying proportionality, proposed by Bar-Gera (2010) and further developed by Xie and Xie (2014).

In seeking to solve the repair sequencing problem, a variety of greedy methods have also been proposed or recommend themselves to potential use, but these sacrifice significant accuracy in pursuit of efficiency. Gokalp et al. (2021) use three greedy methods to benchmark their beam search heuristic. The first method targets the greatest immediate benefit at each stage of repairs. Specifically, at each stage, the link chosen for repair is the one which maximizes the decrease in TSTT from repairing that single link over the repair duration of that link. The second greedy method orders links for repair based on the myopic greatest immediate benefit, or the decrease in TSTT due to repairing that link first over the repair duration of that link. Finally, the importance factor ordering prescribes that links are repaired in decreasing order of pre-disruption flow. All three methods are tested against the beam search heuristic over instances with 8 and 16 broken links on the Sioux Falls network and 8, 16, and

32 broken links on the Anaheim network. For 8, 16, and 32 broken links, the exact enumeration method would need to examine approximately 4.0×10^4, 2.1×10^{13}, and 2.6×10^{35} sequences respectively, while the induced graphs for the shortest path method contain approximately 1.0×10^3, 5.2×10^5, and 6.9×10^{10} links respectively.

Rey and Bar-Gera (2020) assume that links are grouped into projects *a priori* and use a sampling method developed by Rey et al. (2019) which is quadratic in the number of projects to approximate the first order effects of each project. Then, they order projects greedily based on either largest average first order effects, or decreasing Smith's ratio (weight-to-duration) using the average first order effects as the weight parameter. In Rey and Bar-Gera's paper (2020), between three and nine work crews are used, and each instance consists of either nine or ten projects on the BMC network. In these scenarios, the exact enumeration method would need to evaluate between nine (for nine projects and eight work crews) and 6.0×10^5 (for ten projects and three work crews) unique repair sequences. As a comparison case, another greedy algorithm – shortest processing time – is used which simply ranks projects by increasing repair duration.

Hackle et al. (2018) propose simulated annealing (SA) as a heuristic for solving a variation on the repair scheduling problem which seeks to minimize costs (namely the costs of repair as well as costs such as increased travel times and disconnections). Additionally, Hackle et al. (2018) allow for multiple types of repair interventions, with each link having a finite set of possible interventions – each with a commensurate cost, duration, and resource requirement – to bring that link back to full functionality. Simulated annealing is a stochastic local search algorithm which uses the neighborhood of a current solution to propose a new solution. The new solution is accepted if it improves upon the current solution, and accepted with some probability less than one if it is worse than the current solution. This heuristic method was first developed by Kirkpatrick et al. (1983) based on ideas put forward by Metropolis et al. (1953). Franzin and Stützle (2019) extensively describe the parameterization of simulated

annealing algorithms, providing a unifying framework with which to compare various implementations and variations of simulated annealing.

With numerous proposed formulations and varied solution methods with differing small degrees of scalability in the literature, I seek to compare some of the available methods to characterize when one method recommends itself over others. At present, none of the proposed non-greedy methods I have encountered handle realistic instances of multiple hundreds of links within a reasonable planning time period of, say, 72 hours. In order to address these areas, I select the single performance measure of TSTT as a foundation on which to build such a comparison of general methods, which could potentially be adapted for extended or similar formulations, and additionally propose a simulated annealing algorithm which seeks to balance solution time and quality considerations.

2.3 Problem Formulation

The model is a bilevel optimization problem, where the upper level objective is to minimize the total travel delay over the full repair horizon, and the lower level problem is the static traffic assignment problem of computing equilibrium flows for a given repair state. The traffic assignment problem was originally formulated by Beckmann et al. (1956) and is extensively described in books by Patriksson (1994) and Boyles et al. (2023). In defining the upper and lower level problems, I use the same notation as described in Gokalp et al. (2021). Given a network with a set of nodes $\mathcal{N}$, a set of links $\mathcal{A}$, and a set of zones $\mathcal{Z}$, each link a has a travel time T_a, which is a non-decreasing function of the flow on link a only. I assume that the travel demand is fixed, and define d_{rs} as the travel demand from zone r to zone s. The set of simple paths connecting zone r to zone s is defined as Π^{rs} and $\Pi = \cup_{(r,s)\in\mathcal{Z}^2}\Pi^{rs}$ is the set of all such paths.

For this model, if a link is damaged (broken) it cannot be traversed until fully repaired. Let $\mathcal{B} \subset \mathcal{A}$ be the set of broken links, and $N = |\mathcal{B}|$ be the number of broken

links. The time needed to repair a broken link $b \in \mathcal{B}$ is designated D_b and is assumed to be both fixed and known *a priori*. There is no restriction that the network must be strongly connected while links remain broken, and a penalty cost is imposed for lost trips resulting from a broken network. A trip is considered lost if the travel time increases more than Q times from the base case (undamaged network) or if the origin and destination are no longer connected through the network. I represent this by creating artificial links directly connecting origin-destination (OD) pairs, with a constant travel time of Q times the equilibrium travel time in the full network. This formulation avoids numerical stability issues in severely damaged networks caused by using the same large constant for all disconnections. I create these artificial links *after* selecting the set of broken links, and only for OD pairs which have been disconnected or whose new equilibrium travel time is high enough that such a link would be used. In this way, I avoid computational challenges created by the naïve method of adding an artificial link between every OD pair. The set of artificial links is denoted $\mathcal{R}$ in the formulation below. This is a deviation from the problem formulation used by Gokalp et al. (2021), and results in slightly longer solution times in order to extend the formulation's realism and broaden applicability.

I assume that repairs proceed sequentially and that user equilibrium is reached after each repair. More precisely, I assume Wardrop's principle (Wardrop, 1952) is satisfied and the time to reach user equilibrium after completion of a repair is negligible compared to the repair duration. Thus, the time to reach user equilibrium can be approximated as an instantaneous transition. This assumption is common in this type of problem formulation and in representing ordinary conditions in transportation networks. The assumption of user equilibrium allows the use of the static traffic assignment problem, for which extremely efficient algorithms exist, as the lower level problem (Dial, 2006).

Because there are N broken links, there are N network stages in a repair sequence as the broken links are repaired. These non-terminal states are indexed by $t \in \{1, \ldots, N\}$. The sequence of repairs is defined by the binary variables y_b^t, where

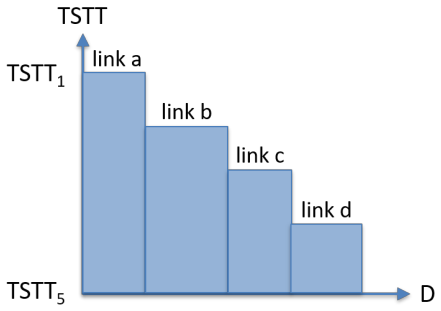

Figure 2.1: The shaded area is total travel delay over the repair horizon

$y_b^t = 1$ iff link b is repaired during stage t. The variable z_b^t equals 1 iff link b was repaired prior to stage t and is therefore useable during stage t. For a given link a in stage t, the flow on that link is x_a^t, and h_π^t represents an equilibrium flow on path π. During stage t, the total system travel time is then $\sum_a x_a^t T_a(x_a^t)$, and stage t lasts for D_b days where b is the link under repair in stage t. The shorthand $TSTT_t$ will be used to compactly denote the TSTT during stage t. Figure 2.1 illustrates the total travel delay over the repair horizon for a possible repair sequence in a network with $N = 4$ broken links. The x-axis in this graph is set at the TSTT before disruption $TSTT_0$, equivalently $TSTT_{N+1}$, after all repairs are complete.

The following bilevel optimization problem expresses minimizing the total travel delay over the repair horizon, where M is a sufficiently large constant:

$$\min_{\mathbf{x},\mathbf{y},\mathbf{z}} \quad \sum_{t=1}^{N} \Big(\sum_{a \in \mathcal{A} \cup \mathcal{R}} x_a^t T_a(x_a^t) - TSTT_0 \Big) \sum_{b \in \mathcal{B}} y_b^t D_b \tag{2.1}$$

$$\text{s.t.} \quad \sum_{t'=1}^{t-1} y_b^{t'} = z_b^t \quad \forall b \in \mathcal{B}, t \in \{1, \ldots, N\} \tag{2.2}$$

$$\sum_{b \in \mathcal{B}} y_b^t = 1 \quad \forall t \in \{1, \ldots, N\} \tag{2.3}$$

$$\sum_{t=1}^{N} y_b^t = 1 \quad \forall b \in \mathcal{B} \tag{2.4}$$

$$y_b^t \in \{0, 1\} \quad \forall b \in \mathcal{B}, t \in \{1, \ldots, N\} \tag{2.5}$$

$$z_b^t \in \{0, 1\} \quad \forall b \in \mathcal{B}, t \in \{1, \ldots, N\}, \tag{2.6}$$

where each $\mathbf{x}_t$ is optimal to:

$$\min_{\mathbf{x}_t, \mathbf{h}_t} \quad \sum_{a \in \mathcal{A} \cup \mathcal{R}} \int_0^{x_a^t} T_a(x) \mathrm{d}x \tag{2.7}$$

$$\text{s.t.} \quad \sum_{\pi \ni a} h_\pi^t = x_a^t \quad \forall a \in \mathcal{A} \cup \mathcal{R} \tag{2.8}$$

$$\sum_{\pi \in \Pi_{rs}} h_\pi^t = d_{rs} \quad \forall (r,s) \in \mathcal{Z}^2 \tag{2.9}$$

$$h_\pi^t \geq 0 \quad \forall \pi \in \Pi \tag{2.10}$$

$$x_b^t \leq M z_b^t \quad \forall b \in \mathcal{B}. \tag{2.11}$$

The upper level objective function, Equation 2.1, minimizes the total travel delay (equivalently, TSTT) over the repair horizon, captured by multiplying the system cost for each stage by the stage duration. The first upper level constraint ensures that each link is available for use only after it is repaired. The second and third upper level constraints ensure that only one link is repaired per stage and each link is repaired at some stage, respectively. The last two upper level constraints define $\mathbf{y}$ and $\mathbf{z}$ as sets of binary variables. The lower level problem consists of Equations 2.7–2.11 and represents the standard user equilibrium formulation of the static traffic assignment problem (TAP), with the addition of the artificial links and a constraint to ensure the flow on broken links is equal to zero.

As formulated, exactly solving the above bilevel optimization problem would require enumerating all possible repair sequences, and evaluating each to determine the optimal sequence, including solving TAP at each repair state encountered. Solving the upper level problem quickly becomes intractable because the number of unique repair sequences is factorial in N, representing all permutations of the broken links $(1, \ldots, N)$. A naïve evaluation method would solve $N! \times N$ instances of TAP, further impacting solution time and tractability. However, by storing repair states and the TSTTs for their respective TAP solutions, the number of TAP evaluations required decreases to 2^N. The repair state at any time t is characterized by a vector $\mathbf{z}^t$ of length N, where each entry is either 1, if the indexed link is repaired, or 0, if the

indexed link is not yet fully repaired. Because each of the N entries has only two possible values, the total number of feasible states is 2^N.

As shown by Gokalp et al. (2021), because links are repaired in sequence, an analogue to Bellman's optimality principle applies: once a subset of repairs is complete, optimizing the repair sequence of the remaining links is independent of the completion order of the first links. This principle holds for any fixed partitioning into subsequences, in that the optimal ordering within each subsequence can be determined independently. The bidirectional beam search heuristic which Gokalp et al. propose and demonstrate within the same paper exploits this principle and is based in dynamic programming.

Until this point, the formulation has treated demand across the network equally, as a single class of demand. However, with a modification to the objective function, a subset of demand can be prioritized, as might be desirable for emergency vehicles and construction crews after a natural disaster. Given a set of demand classes $(1, \ldots, d)$, the modified upper level objective function is presented below, with class weights w_i for $i = 1, \ldots, d$. In this manner, the lower level problems, and therefore route choices given a network configuration, are unaffected. Allowing for multiple classes of demand, the modified upper level objective function is:

$$\min_{\mathbf{x},\mathbf{y},\mathbf{z}} \quad \sum_{t=1}^{N} \Big(\sum_{a \in \mathcal{A} \cup \mathcal{R}} \Big(\sum_{i=1}^{d} w_i x_{a,i}^t \Big) T_a(x_a^t) - TSTT_0 \Big) \sum_{b \in \mathcal{B}} y_b^t D_b,$$

where $x_{a,i}^t$ is the flow on link a from class i during stage t and $\sum_{i=1}^{d} x_{a,i}^t = x_a^t$.

Another technique to prioritize different classes of demand might be to modify parameters of the demand classes themselves. However, this type of modification would cause commensurate changes in individual vehicle behavior, rather than simply influencing the objective function. By elevating, for example, the distance factor for the priority class, an EMS or construction vehicle could be penalized more heavily than a private car for having to take a longer route, in addition to the penalty of increased travel time. However, this weighting technique risks unintentionally

incentivizing undesirable behavior because the underlying route choices would be altered by the modified lower level objective function.

2.4 Alternate Formulation and Complexity

While I choose to formulate the problem at hand as a bilevel optimization problem in order to incorporate calculation of TSTT for each visited state as the lower level problem, with TSTTs treated as known (or calculated as needed) the problem can be reformulated as a single-pair shortest path problem. This reformulation was briefly described by Gokalp et al. (2021), but not tested computationally. Ng and Schonfeld (2023) also present the same shortest path reformulation, testing computationally with four broken links on the Sioux Falls test network. In this reformulation, there are 2^N nodes, corresponding to the 2^N possible repair states ($\mathbf{z}^t$ vectors). The network links are those which connect a repair state $\mathbf{z}^t$ to a valid successor state $\mathbf{z}^{t+1}$. In this way, a link in the network represents repairing one additional link not yet repaired in state $\mathbf{z}^t$. The link cost for a link (k, l) from state k to state l where state l differs from state k by the repair of link b is $(TSTT_k - TSTT_0) \cdot D_b$ where $TSTT_0$ is the TSTT after the disruption and before any links are repaired. The number of links in the network is $N \cdot 2^{N-1}$. Figure 2.2 depicts the induced network for $N = 3$, where the shortest path from $\mathbf{z}^0 = (0, 0, 0)$ with no links repaired to $\mathbf{z}^4 = (1, 1, 1)$ with all links repaired defines the optimal repair order for the three broken links.

Since the induced network is a directed acyclic graph, a topological order exists, and can be used to solve for the shortest path with a time complexity of $O(E)$ where $E = N \cdot 2^{N-1}$ is the number of edges (Ahuja et al., 1993). Therefore, the time complexity of finding the optimal sequence after calculating the link costs (which involves finding the TSTT for 2^N instances of TAP) is $O(N \cdot 2^{N-1})$. While this is the time complexity for solving single-source shortest paths to all nodes, due to the network structure, any link and any node in the network can be on the single-pair shortest path from $\mathbf{z}^0$ to $\mathbf{z}^{N+1}$, and no paths can be excluded from consideration

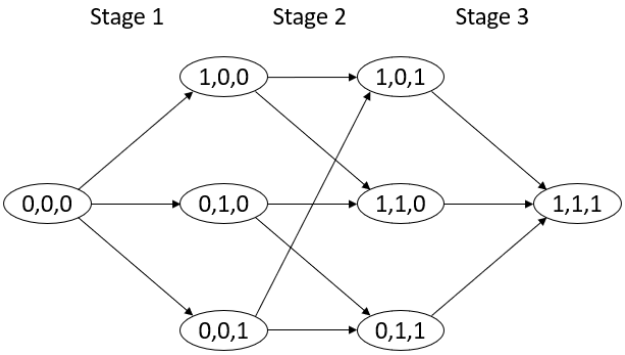

Figure 2.2: Induced network for $N = 3$

a-priori due to only requiring the single-pair shortest path from $\mathbf{z}^0$ to $\mathbf{z}^{N+1}$.

2.5 Toy Examples

A logical next question in regards to complexity is whether the induced network, and therefore the problem complexity, can be reduced by some property of the original network, either by combining or eliminating nodes, thus also eliminating links, or by simply eliminating links. If certain links could be removed *a priori*, without calculating TSTTs, then the worst case complexity of solving the shortest path problem, and therefore the complexity of the equivalent bilevel problem would be reduced. In order to determine whether the problem can be simplified or reduced, I first examine the toy instance depicted in Figure 2.3 with source node s, destination node t, three links from s to t, and total demand d from s to t. The top link a remains functional, while links b and c are selected as the broken links. While in practice the most common link performance function (LPF) is the Bureau of Public Roads (BPR) function

$$t_{ij}(x_{ij}) = t_{ij}^0\Big(1 + \alpha\Big(\frac{x_{ij}}{u_{ij}}\Big)^{\beta}\Big)$$

with $\alpha = 0.15$ and $\beta = 4$ as common defaults, I first examine this toy network using constant, and then linear link performance functions.

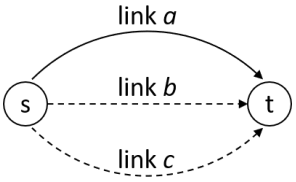

Figure 2.3: Example network with three links, one OD pair

2.5.1 Constant Link Performance Function

In order to obtain a constant link performance function, I set $\alpha = 0$ and $\beta = 1$, resulting in the simplified function

$$t_{ij}(x_{ij}) = t_{ij}^0$$

where travel time is equal to free flow time t_{ij}^0 and no longer depends on link flow. In this case, at each stage of repairs, all traffic will route on the link with the lowest free flow time out of the set of currently usable links. Indexing by link name (a, b, c) for this example, rather than by origin and destination, if $t_a^0 = \min\{t_a^0, t_b^0, t_c^0\}$, then the total travel delay will equal zero, since the fastest path is always available and travel time does not depend on link flow. Mathematically, if $t_a^0 = \min\{t_a^0, t_b^0, t_c^0\}$, the TSTTs before the network disruption, immediately after the disruption, and for each possible intermediate repair state are all equal to $d \cdot t_a^0$ since all demand uses link a. Since the objective function measures travel delay, and there is in fact no delay, the objective value is zero, and the order in which links b and c are repaired is immaterial, regardless of link repair durations and free flow times on links b and c, given that the free flow times are greater than t_a^0.

If instead $t_a^0 > t_b^0 > t_c^0$ and the repair times for links b and c are D_b and D_c, respectively, then Table 2.1 gives the objective function for the two possible repair sequences, with $TSTT_0 = d \cdot t_c^0$ since link c has the lowest free flow time. Setting the two objective functions equal and simplifying, when

$$\frac{t_a^0 - t_b^0}{D_b} = \frac{t_a^0 - t_c^0}{D_c},$$

Repair Sequence	Total Travel Delay
(b, c)	$dt_a^0 D_b + dt_b^0 D_c - dt_c^0(D_b + D_c)$
(c, b)	$dt_a^0 D_c + dt_c^0 D_b - dt_c^0(D_c + D_b)$

Table 2.1: Sequence comparisons, constant LPF, $N = 2$

Repair Sequence	Unit Total Travel Delay
(b, c, d)	$t_a^0 D_b + t_b^0 D_c + t_c^0 D_d - t_c^0(D_b + D_c + D_d)$
(b, d, c)	$t_a^0 D_b + t_b^0 D_d + t_d^0 D_c - t_c^0(D_b + D_c + D_d)$
(c, b, d)	$t_a^0 D_c + t_c^0 D_b + t_c^0 D_d - t_c^0(D_b + D_c + D_d)$
(c, d, b)	$t_a^0 D_c + t_c^0 D_d + t_d^0 D_b - t_c^0(D_b + D_c + D_d)$
(d, b, c)	$t_a^0 D_d + t_d^0 D_b + t_d^0 D_c - t_c^0(D_b + D_c + D_d)$
(d, c, b)	$t_a^0 D_d + t_d^0 D_c + t_d^0 D_b - t_c^0(D_b + D_c + D_d)$

Table 2.2: Sequence comparisons, constant LPF, $N = 3$

the objective functions for the two repair sequences are equal. If instead

$$\frac{t_a^0 - t_b^0}{D_b} > \frac{t_a^0 - t_c^0}{D_c},$$

then sequence (b, c), repairing link b first, achieves a lower objective value than sequence (c, b). Note that the answer is not necessarily to fix the link with the lowest free flow time first, as a long repair duration for this link might offset its benefits. In this case, because link travel times are not dependent on link flows, and all traffic routes on the available link with the lowest free flow time, the optimal repair order is in accordance with a decreasing Smith's ratio (a weight to duration ratio), where the weight parameter is the difference in travel time between the always available link and the link chosen for repair. Once the link with the lowest free flow time has been repaired, however, the remainder of the repair order does not affect the objective value.

With four links total, three of which (b, c, d) are broken in a disruption, the possible repair sequences and their objective values are enumerated in Table 2.2. Since demand is not split between paths at any stage and in constant over the repair horizon, I drop it from the equations below for readability, calculating unit total travel delay. I assign $t_a^0 > t_b^0 > t_c^0 > t_d^0$. Of note, the last two repair sequences have

equivalent objective functions, because once link d is repaired, traffic no longer shifts due to further repairs because link d has the lowest free flow time of the four links. As in the case above with only two broken links, any two sequences can be compared by comparing their objective functions. For example, in order for repairing link d first to be optimal, the following four inequalities must hold, corresponding to comparing either of the last two sequences (which are equivalent in terms of objective value) to each of the first four sequences:

$$t_a^0 D_d + t_d^0 D_b + t_d^0 D_c < t_a^0 D_b + t_b^0 D_c + t_c^0 D_d$$

$$t_a^0 D_d + t_d^0 D_b + t_d^0 D_c < t_a^0 D_b + t_b^0 D_d + t_d^0 D_c$$

$$t_a^0 D_d + t_d^0 D_b + t_d^0 D_c < t_a^0 D_c + t_c^0 D_b + t_c^0 D_d$$

$$t_a^0 D_d + t_d^0 D_b + t_d^0 D_c < t_a^0 D_c + t_c^0 D_d + t_d^0 D_b$$

Using the second inequality, the third term on each side cancels out, and the inequality reduces to

$$\frac{t_a^0 - t_b^0}{D_b} < \frac{t_a^0 - t_d^0}{D_d},$$

indicating that the Smith's ratio for link d must be greater than for link b in order to fix d before b when fixing c last. The fourth inequality provides a corresponding result when fixing link b last. I have now established that for fixing link d first to be optimal, it must have the highest Smith's ratio of the links to be repaired, and the order of links b and c in that case is immaterial. I can similarly establish that links b and c should be sequenced in order of descending Smith's ratio if link d is repaired last, by comparing sequences (b, c, d) and (c, b, d). Therefore, for b to be optimally repaired first, the inequality

$$\frac{t_a^0 - t_b^0}{D_b} > \frac{t_a^0 - t_c^0}{D_c}$$

must hold, as well as

$$\frac{t_a^0 - t_b^0}{D_b} > \frac{t_a^0 - t_d^0}{D_d},$$

from above (because it cannot be optimal for both b and d to be repaired first using a single crew unless $(t_a^0 - t_b^0)/D_b = (t_a^0 - t_d^0)/D_d$). The same logic indicates that link c must have the largest Smith's ratio in order to be optimally repaired first. Once the optimal first link is established, due to the analogue to Bellman's optimality principle discussed in §2.3, if the link with the lowest free flow time has not yet been repaired, the next optimal link can be solved for recursively, applying the logic from the case above with only two broken links. Therefore, the optimal repair order will always be in descending order of the links' Smith's ratios, with the caveat that once the link with the lowest free flow time has been repaired the remaining repair order does not affect the objective value. This result can be easily extended recursively to a case where there are l links from s to t, and $(l-1)$ or fewer of the links are broken.

2.5.2 Linear Link Performance Function

The next simplest link performance function is linear. In the context of a BPR function, a linear function is obtained by setting $\alpha > 0$ and $\beta = 1$ such that the simplified function is

$$t_{ij}(x_{ij}) = t_{ij}^0\Big(1 + \alpha\Big(\frac{x_{ij}}{u_{ij}}\Big)\Big).$$

Once again, I start with the toy instance depicted in Figure 2.3. The top link a remains functional, while links b and c are selected as the broken links. Unless link a dominates (or is dominated by) link b or c in terms of travel time for the demand level d, flow will be split between link a and the first link to be repaired (link b or c) while the final link is being repaired. In order to determine whether link a dominates link b, for example, I would compare the quantities $t_a(d)$ and $t_b(0)$. If $t_a(d) < t_b(0)$, that is the travel time on link a with 100% of the demand from s to t assigned to link a is still less than the travel time on link b with zero demand, then link a dominates link b in terms of travel time, and fixing link b first will not result in reassignment of any demand to link b or a reduction in travel delay.

Travel times $t_{ij}(x_{ij})$ on each link now depend on the flow on that link. Fur-

Sequence	Total Travel Delay
(b, c)	$dt_a(d)D_b + [x_{a1}t_a(x_{a1}) + (d - x_{a1})t_b(d - x_{a1})]D_c - TSTT_0(D_b + D_c)$
(c, b)	$dt_a(d)D_c + [x_{a2}t_a(x_{a2}) + (d - x_{a2})t_c(d - x_{a2})]D_c - TSTT_0(D_b + D_c)$

Table 2.3: Sequence comparisons, linear LPF, $N = 2$

thermore, x_{a1} is the flow on link a after link b is repaired first, or alternately, x_{a2} is the flow on link a after link c is repaired first. The two possible objective functions are represented in Table 2.3. Unless link a dominates (or is dominated by) link b or c for the demand level d, thereby eliminating either x_{a1} or x_{a2}, setting the two equations in Table 2.3 equal to each other does not yield a meaningful relationship without numerically solving for x_{a1} and x_{a2}. In the same toy problem, but with l links from s to t, this would require solving balance equations for flows at each repair state for each possible repair sequence in order to find the optimal repair sequence. Unfortunately, this result indicates that even for linear link performance functions, a single OD pair, and single-link paths, the overall problem complexity does not decrease except in specific cases of link dominance, though the individual TAP calculations are less intensive for linear link performance functions and single-link paths.

2.6 Solution Methods

This section outlines solution methods that will be compared in the experiments below. In light of the complexity results for solving to optimality, I first specify six "greedy" methods which can be evaluated fairly quickly, followed by two iterative methods which examine a larger set of repair sequences. The first iterative method is a refinement of the bidirectional beam search heuristic proposed by Gokalp et al. (2021), and the second is an implementation of the simulated annealing metaheuristic. These methods are not fully independent, since the greedy methods can be used to initialize simulated annealing, or to provide upper bounds to limit branching in the beam search.

2.6.1 Greedy Methods

I implement and compare six greedy methods: shortest processing time, importance factor, lazy greedy, sequential greedy, largest average first-order effect, and largest average Smith's ratio. The shortest processing time (SPT) heuristic is the simplest heuristic proposed in that the only information required is the link repair duration for each link. SPT simply orders the links from shortest repair time to longest repair time, repairs the links in that order, and requires no preprocessing. The importance factor (IF) heuristic is also a simple indexing, however, since links are ordered for repair in descending order of pre-disruption flow. A single TAP must be solved to obtain this information.

The lazy greedy method (LZG) constructs the repair sequence by ordering links by greatest immediate benefit, defined as decrease in TSTT due to repairing the link first in the repair sequence over the link repair duration, solving $O(N)$ TAPs. The sequential greedy method (SQG) constructs the repair sequence myopically at each stage, choosing for repair the link which provides the greatest immediate benefit at that stage of repairs, defined in the same manner as for LZG. At each stage in building the repair sequence, the link is chosen which immediately maximizes this value, and finding a repair sequence using SQG requires solving $O(N^2)$ TAPs.

The final two greedy methods were developed by Rey and Bar-Gera (2020) for sequencing projects of multiple simultaneous link repairs, but can be applied to the current problem by considering projects consisting of a single link repair. These methods first solve TAP for a subset of all possible network states during the repair sequence, and then use this sample to approximate the average first-order effects of repairing each link at any point in time. The sampling method is detailed in Rey et al. (2019) and requires solving a number of TAPs quadratic in the number of projects. Once this sampling is complete, link repairs can be sequenced in order of largest average first-order effects (LAFO), or largest average Smith's ratio (LASR). Smith's ratio is the quotient of approximate first-order effect and repair duration, allowing

LASR to account for repair duration alongside estimated benefit.

2.6.2 Bidirectional Beam Search Heuristic

I extend the bidirectional beam search heuristic of Gokalp et al. (2021) to handle cases where the network is no longer strongly connected while links remain broken. This extension is key to greater realism in scenarios where large portions of the network are strongly impacted by damage, such as has been recorded after extreme hazards or natural disasters. The implementation also improves numerical stability by using a disconnection cost as discussed in §2.3 of Q times the base travel time (in the undamaged network) rather than a single large constant. This extension results in longer solution times for all methods which involve solving TAP, since an artificial link is added for each disconnected origin-destination pair. The following description is a summary of the beam search method with my modifications; the full original algorithm with all details can be found in the original paper by Gokalp et al. (2021).

The base procedure for the bidirectional beam search is the same as that presented by Gokalp et al. (2021), but using a modified network as discussed above, as well as modified initialization and pruning parameters. Algorithm 1 is directly adapted from that paper. To initialize the beam search, I seed the search with a best feasible solution (BFS), taken as either the sequential greedy solution or the importance factor solution, based on which has the lower objective function. Two search fronts are established, with the root node for the forward search (starting with no links repaired) labeled α and the root node for the backward search (starting from the state once all links are repaired) labeled ω. As a shorthand, $TAP(\mathcal{R} \cup \mathcal{A} \setminus \mathcal{B})$ represents solving for user equilibrium and returning the TSTT when all links in $\mathcal{A}$ and $\mathcal{R}$ are usable except for the links in $\mathcal{B}$ (i.e. $TSTT(\alpha)$). Similarly, $TSTT(\omega)$ is set to $TAP(\mathcal{R} \cup \mathcal{A})$.

The sets $Y_f(s)$ and $Y_b(s')$ represent the set of links repaired in state s on the

Algorithm 1 Pseudocode for bidirectional beam search $(\mathcal{B}, D, r)$

Initialization

$Open_f \leftarrow \alpha, Open_b \leftarrow \omega, (BFS_B, BFS) \leftarrow \min(SQG, IF), counter \leftarrow 0$
$g_f(\alpha) \leftarrow 0, g_b(\omega) \leftarrow 0, p_f(\alpha) \leftarrow \varnothing, p_b(\omega) \leftarrow \varnothing, Y_f(\alpha) \leftarrow \varnothing, Y_b(\omega) \leftarrow \varnothing$
$TSTT(\alpha) \leftarrow TAP(\mathcal{R} \cup \mathcal{A} \setminus \mathcal{B}), TSTT(\omega) \leftarrow TAP(\mathcal{R} \cup \mathcal{A})$

Preprocessing

for each $b \in \mathcal{B}$ **do**
    $TSTT \leftarrow TAP(\mathcal{R} \cup (\mathcal{A} \setminus \mathcal{B}) \cup \{b\})$
    $wb_b \leftarrow TSTT(\alpha) - TSTT$
    $TSTT \leftarrow TAP(\mathcal{R} \cup \mathcal{A} \setminus \{b\})$
    $bb_b \leftarrow TSTT - TSTT(\omega)$
    if $bb_b < wb_b$ **then** swap bb_b, wb_b

Search Loop

while $Open_f \neq \varnothing$ **and** $Open_b \neq \varnothing$ **and** $\max(\min_{s \in Open_f} h(s), \min_{s \in Open_b} h(s)) < BFS_B$ **do**
    if $|Open_f| \leq |Open_b|$ **then**
        Expand the forward front
    else
        Expand the backward front
    if $counter \pmod r \equiv 0$ **then**
        for each $s' \in Open_b \cup Open_f$ **do**
            Update bounds
        Use beam search in order to reduce the search space
    $counter \leftarrow counter + 1$

forward front or *not* repaired in state s' on the backward front, respectively. Because Bellman's optimality principle applies, and a single link is added in each successive state, it is sufficient to store the predecessor state label for each successive state generated on either the forward or backward front. These predecessors are designated $p_f(s)$ and $p_b(s')$ for the forward and backward predecessors, respectively. Similarly, $g_f(s)$ and $g_b(s')$ store the cost corresponding to each of these partial sequences. $Open_f$ and $Open_b$ contain the states on each search front which are not yet expanded. Once a state is expanded, it is then removed from the corresponding search front.

The preprocessing procedure finds heuristic bounds on the best and worst benefit for each link $b \in \mathcal{B}$ (highest and lowest marginal impact of repairing b). Due to the structure of the problem, the calculated quantities used for each link b are the marginal impact of repairing b first and the marginal impact of repairing b last. The higher marginal impact of the two is assigned as the best benefit and the lower marginal impact is assigned as the worst benefit. During each iteration of the search loop, a state is selected for expansion from the search front which has more states not yet expanded. If the heuristic best bound from the selected state is greater than the current BFS, that state is pruned, rather than expanded. Otherwise, potential expansions (child nodes) are found, evaluated, and added to the appropriate search front if their heuristic best bound is less than the current BFS. Every $r = 128$ iterations, a greedy procedure is used to obtain a feasible solution and update the BFS if the found feasible solution is better than the previous BFS.

I experiment with seeding the beam search algorithm with various best known feasible solutions to include those found by SQG, IF, SPT, and my SA algorithm, presented in §2.6.3. Based on numerical experimentation, I find the SQG and IF solutions for each instance where I solve using the beam search and seed the beam search with the better solution of the two (lower objective function value), since the approaches operate by distinct mechanisms and neither dominates the other for every instance in terms of solution quality. Although this procedure requires the solution of $O(N^2)$ TAPs in order to find the SQG and IF solutions and evaluate their objective

function values, the TAP solutions found during the SQG procedure are saved to memory and can be used without resolving TAP whenever the beam search encounters those states, regaining some of the initial time spent. My numerical experiments show that this seeding procedure results in minor overall gains in terms of solution time, at equal solution quality.

When I experiment with seeding the beam search heuristic with the results of the SA heuristic, I discover that while the time from seeding to termination often decreases, the decrease does not fully compensate for the time spent finding the SA solution, and solution quality does not increase in most instances. After fixing the method of seeding, I retune when to start pruning nodes in the beam search, resulting in pruning starting once $3N$ iterations complete instead of at iteration 100, so that the pruning rule scales linearly with instance size. Pruning refers to eliminating nodes from the search tree if their estimated lower bound is greater than the current best feasible solution.

In addition, I investigate the ordinal location in the repair sequence which results on average in the best benefit, that is, the highest marginal impact of repairing a specific link. I use an average because for positions other than first or last, the links already repaired will impact the benefit gained by repairing the next link. I investigate values of $N = 8$ and $N = 10$ on the Anaheim and BMC networks. I find the average marginal impact of repairing each broken link at each ordinal position in the repair sequence, and transform the results into count data for the best and worst repair positions over 25 repetitions on each network for each instance size. For example, in the top left quadrant of Figure 2.4, the blue bar of height 0.70 in category 8 on the x-axis indicates that for 70% of the $8 \times 25 = 400$ links tested, the best marginal benefit (largest immediate decrease in TSTT) occurred when the link is repaired eighth out of the eight broken links in the instance.

Approximately 65–80% of the time, the best benefit (highest average marginal impact) which can be achieved by fixing a single link is achieved by fixing that link

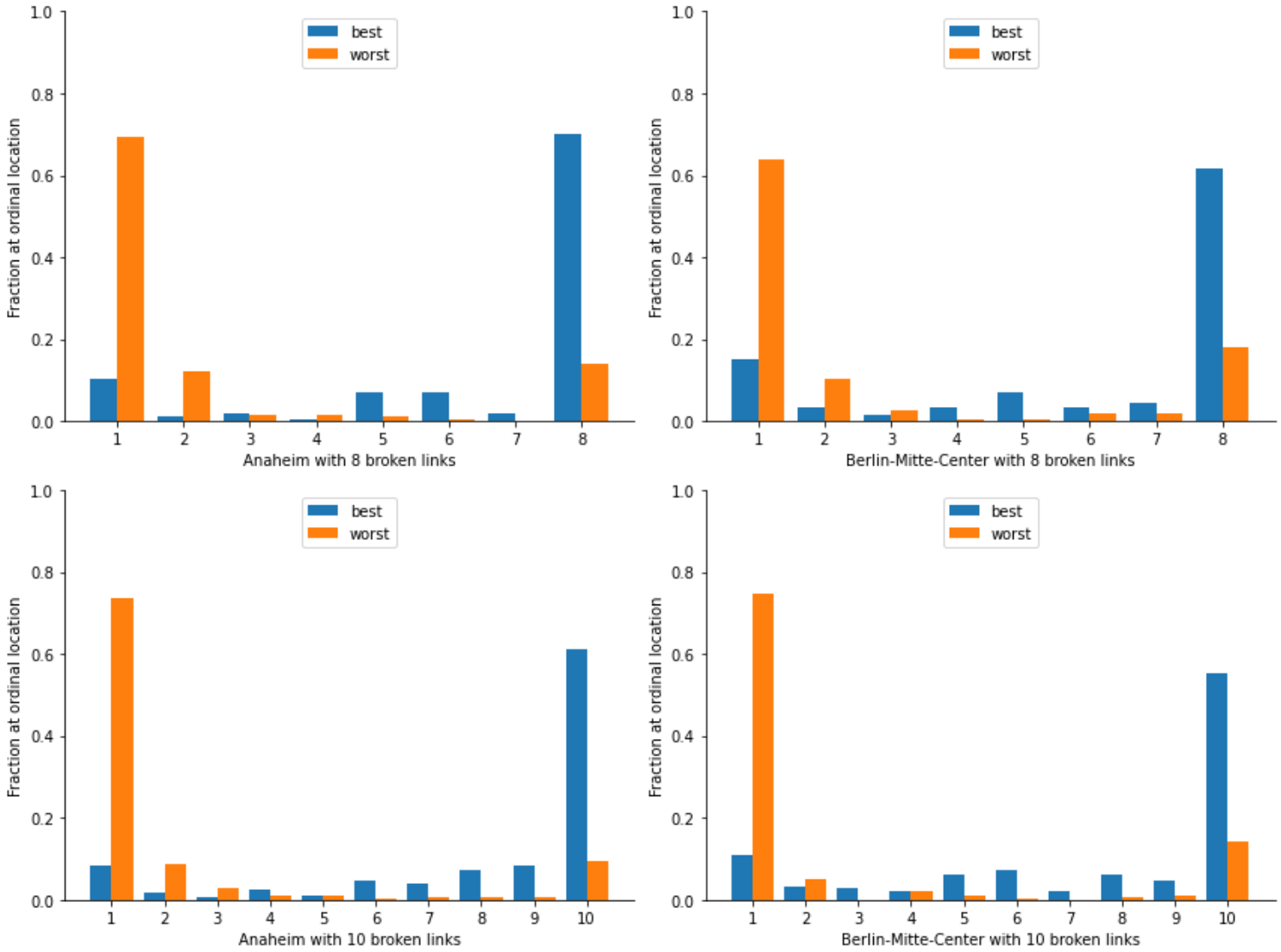

Figure 2.4: Ordinal locations of best and worst average marginal impact; horizontal axis indicates location in repair order (1st through 8th, or 1st through 10th)

either first or last. This statistic is critical to the performance of the beam search algorithm, because the beam search relies on the "best benefit" and "worst benefit" of repairing each link to establish upper and lower bounds for branching and pruning. These values are obtained by finding the marginal impact of repairing each link first and the marginal impact of repairing each link last, with the higher marginal impact of the two for each link assigned as the best benefit and the lower marginal impact assigned as the worst benefit. If the true best or worst benefit were often achieved by repairing a link in the middle of the repair sequence, rather than first or last, these bounds would be invalid, and potentially lead to poor branching and pruning decisions.

Interestingly, the majority of the time (55–70%) the best benefit is achieved by fixing a link last, rather than first or in the middle of the sequence. A potential intuition for this result is that breaking a single link in a fully functional network typically has a greater impact on TSTT than breaking an eighth or tenth link. Specifically, since in my numerical experiments broken links are geographically clustered, the paths disrupted by the eighth or tenth broken link may have already been significantly impacted by one of the other broken links in close proximity, and therefore the additional increase in TSTT is often marginally smaller if a link is broken last versus first.

2.6.3 Simulated Annealing Heuristic

While the beam search developed by Gokalp et al. (2021) and refined above provides high quality solutions in much quicker run times than solving the recovery sequencing problem by brute force, simulated annealing has the potential to provide further significant run time improvements. First, the neighborhood of a given solution must be defined. For the simulated annealing algorithm, I define a 1-neighborhood where a single link in the current recovery sequence (other than the last link in the sequence) is selected and swapped with the link immediately following. In other words, the 1-neighborhood consists of sequences formed by exactly one adjacent swap. For

example, with five broken links, designated link a through e, and a current sequence (a, b, c, d, e), the 1-neighborhood consists of these sequences:

$$(b, a, c, d, e),$$
$$(a, c, b, d, e),$$
$$(a, b, d, c, e),$$
$$(a, b, c, e, d).$$

Table 2.4: 1-neighborhood of sequence (a, b, c, d, e)

With this definition, for a network with N broken links, the size of the 1-neighborhood is always $N - 1$. An analogous definition of a k-neighborhood involves moving a single link k positions *or fewer* forward or backward in the repair order, thus including all similarly defined smaller neighborhoods. Testing 2- and 3-neighborhoods on the Anaheim and BMC networks indicates slightly improved solution quality in some instances, but at a computational time cost outweighing the improvement. Figure 2.5 depicts the solution's percentage improvement relative to the initial BFS over the running time of the algorithm for five instances on the Anaheim network, comparing 1-, 2-, and 3-neighborhoods. The top, middle, and bottom ovals highlight cases where the solutions obtained using 2- and 3-neighborhoods are somewhat superior, slightly inferior, and equivalent, respectively, to the solution obtained using the 1-neighborhood. However, constant to all cases is that the algorithms using 2- and 3-neighborhoods have a higher computational time.

Larger neighborhoods exhibit a larger computational time than the 1-neighborhood. When using the 1-neighborhood, at most four instances of TAP are solved to find the objective value of the new proposed solution due to the analogue to Bellman's principle which underpins the beam search discussed in §2.6.2 (Gokalp et al., 2021). Specifically, let link a be selected to be swapped with b (a is repaired immediately before b in the current solution) and $Y_b(s')$ be the set of links repaired after both link a and b. Then, in order to obtain the objective function value of the new proposed solution, only the following values must be calculated:

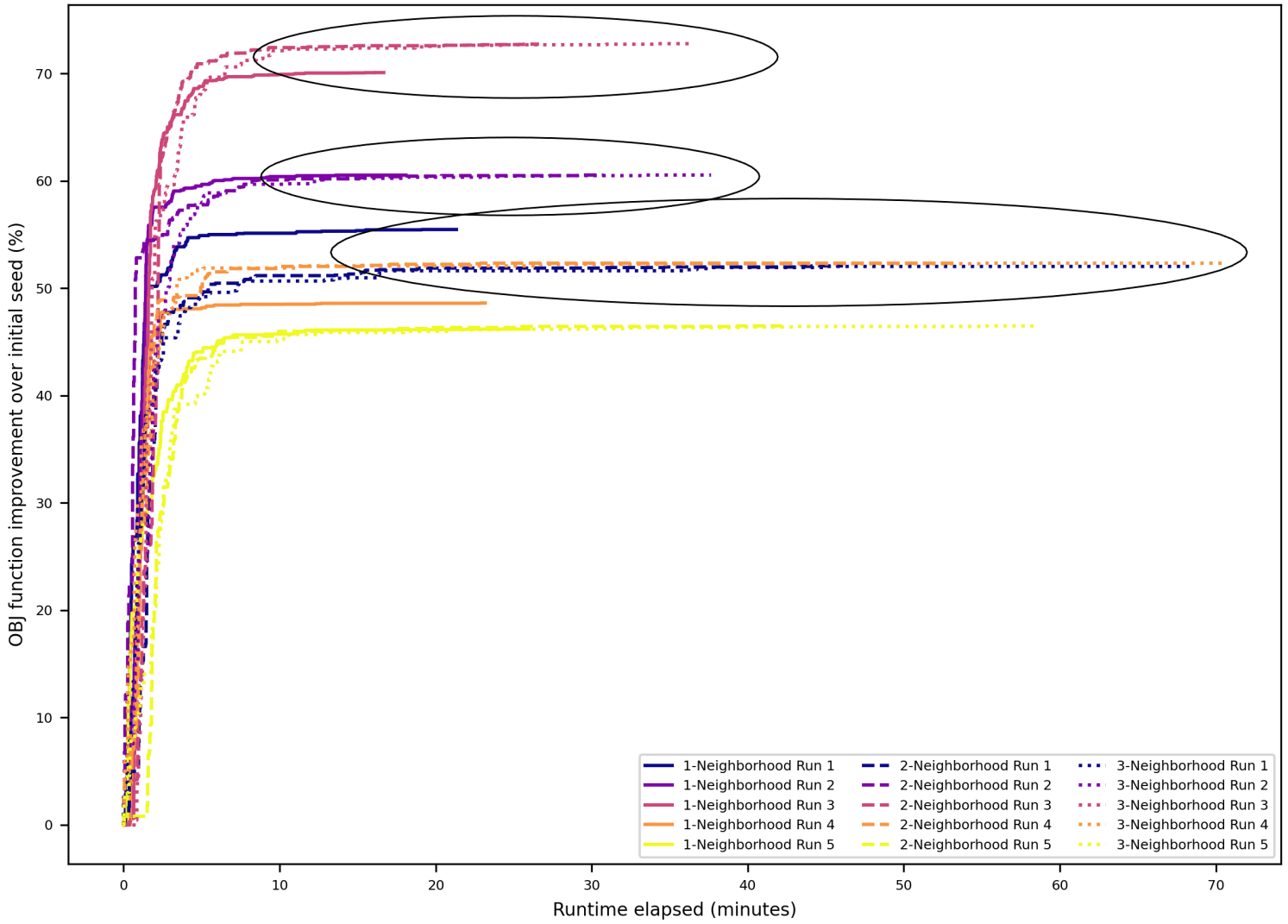

Figure 2.5: Runtime elapsed (minutes) versus OBJ function improvement (%) for Anaheim with 20 broken links; colors represent different runs, line style differentiates choice of neighborhood (solid for 1-neighborhood, dashed for 2-neighborhood, dotted for 3-neighborhood)

$$\begin{aligned} \text{TSTT after both } a \text{ and } b \text{ are repaired} \quad & TAP(\mathcal{R} \cup \mathcal{A} \setminus (a \cup b \cup Y_f(s'))), \\ \text{TSTT after only } a \text{ is repaired} \quad & TAP(\mathcal{R} \cup \mathcal{A} \setminus (a \cup Y_f(s'))), \\ \text{TSTT after only } b \text{ is repaired} \quad & TAP(\mathcal{R} \cup \mathcal{A} \setminus (b \cup Y_f(s'))). \end{aligned}$$

For larger k-neighborhoods, the number of TAPs which may need to be solved to find each new objective value also grows. None of the three methods (1-, 2-, and 3-neighborhoods) dominates the others, but given the generally small and inconsistent performance gaps, and further testing on the Anaheim and BMC networks, I choose to focus on the 1-neighborhood in subsequent computational experiments.

In building the simulated annealing algorithm, the remaining parameter choices to be selected and tuned are: initial temperature, stopping criterion, exploration criterion, acceptance criterion, temperature length, cooling scheme, and temperature restart (Franzin and Stützle, 2019). Based on numerical experimentation, the number of iterations required to reliably obtain high quality solutions varies significantly with instance size (number of broken links, N). Therefore, based on that experimentation, I tune max iterations (stopping criterion) to $1.2N^3$. I choose the initial temperature, T_0, to obtain an initial probability of approximately 10% to accept a move which increases the objective function by 10%. I use a variant on the Lundy-Mees cooling scheme where $T_{i+1} = T_i/(a + b \times T_i)$, with $a = b = 1$ as proposed by Szu and Hartley (1987), with a temperature length of one, indicating temperature updates every iteration. I also explored geometric cooling schemes as defined by Kirkpatrick et al. (1983), varying both geometric parameters and epoch (temperature) length, but these did not result in comparable solution quality to that produced by Szu and Hartley's variant of Lundy-Mees. No temperature restarts are tested within the scope of these experiments, but would be an interesting direction for research to further tune the simulated annealing method.

The exploration criterion used at each step is to randomly select a proposed new solution from the current 1-neighborhood. I also test a slight modification to this default behavior, maintaining a dictionary of the indices of current successive

failures, that is, those since the last accepted movement. In this manner, resources are not wasted by re-evaluating the same solution multiple times without intermediate movement. If all possible moves from a current solution are evaluated and rejected by the acceptance criterion defined below, then either the smallest increase in objective function is taken as the new solution to escape the local minimum, or a random solution from the 1-neighborhood is simply accepted with probability one.

However, both procedures for maintaining a dictionary of successive failures result in a significant decrease in solution quality over the simulated annealing method without maintaining the dictionary of rejected indices. When testing on the Anaheim and BMC networks, with instance sizes from 16 to 24 broken links and testing 10 random instances at each size, the base simulated annealing method finds the best solution of the three methods in every single tested instance. The average objective function accuracy gaps with respect to the best found solution for each network and instance size when maintaining a dictionary of successive failures range between 15% and 29%. The standard deviations of the accuracy gaps range between 9% and 28%. The first procedure (smallest increase in objective function in 1-neighborhood taken as the new solution after rejecting each with the normal acceptance criterion) results in the fastest solution times among tested simulated annealing methods in many instances, but, due to the higher likelihood of becoming stuck in the vicinity of a local minimum, fails to achieve quality solutions. The second procedure avoids the locally quasi-deterministic quality of the first procedure, but demonstrates less consistent decreases in solution time, while still degrading solution quality. For these reasons, neither modification is used in further experiments.

For acceptance criterion, I use the generalized simulated annealing variant proposed by Bohachevsky et al. (1986), with $g = -1$ and $\beta = (1/T)^{2/3}$. The general formula is:

$$p_{GSA} = \begin{cases} 1 & \text{if } \Delta(s', s) \leq 0 \\ \exp(-\beta f(s)^g \Delta(s', s)) & \text{otherwise,} \end{cases}$$

where s is the current solution, $f(s)$ is the current solution's objective function value, s' is the proposed new solution, and $\Delta(s', s)$ is the change in objective function value due to the proposed step. Applying the selected values for g and β, the formula used in this application is

$$p_{GSA} = \begin{cases} 1 & \text{if } \Delta(s', s) \leq 0 \\ \exp\left(\frac{-\Delta(s', s)}{f(s) \times T^{2/3}}\right) & \text{otherwise.} \end{cases}$$

Finally, I explore using multiple, shorter runs of the simulated annealing algorithm, rather than a single run. Instead of a single run with max iterations equal to $1.2N^3$, I conduct four runs with max iterations of $1.2N^{2.5}$ each, selecting the solution with the lowest objective function value from among the four runs. I hypothesized that this method might improve solution quality by exploring on four distinct paths, while saving computation time when the number of broken links is greater than 16. However, solution quality decreases due to the shorter runs, and diversification does not regain the lost solution quality.

Finally, in order to quantify the consistency of the fully parameterized simulated annealing method, I run a series of experiments where I conduct ten runs of simulated annealing on each problem instance, over ten instances each for 16, 20, and 24 broken links on the Anaheim and BMC networks. The box and whisker plots in Figure 2.6 show that while there is variability in the solutions and objective function values obtained for the same problem instance on different simulated annealing runs, the objective function gaps are reasonably small for the majority of instances tested.

Specifically, the percentage gap of the worst solution obtained versus the best solution obtained was no more than 5.5% for two-thirds of instances tested across the three instance sizes and two networks. Under 12% of instances tested exceeded a 10% gap between the worst and best solutions obtained. Figure 2.7 depicts the combined frequencies of the largest observed relative accuracy gap for each instance over all tested configurations.

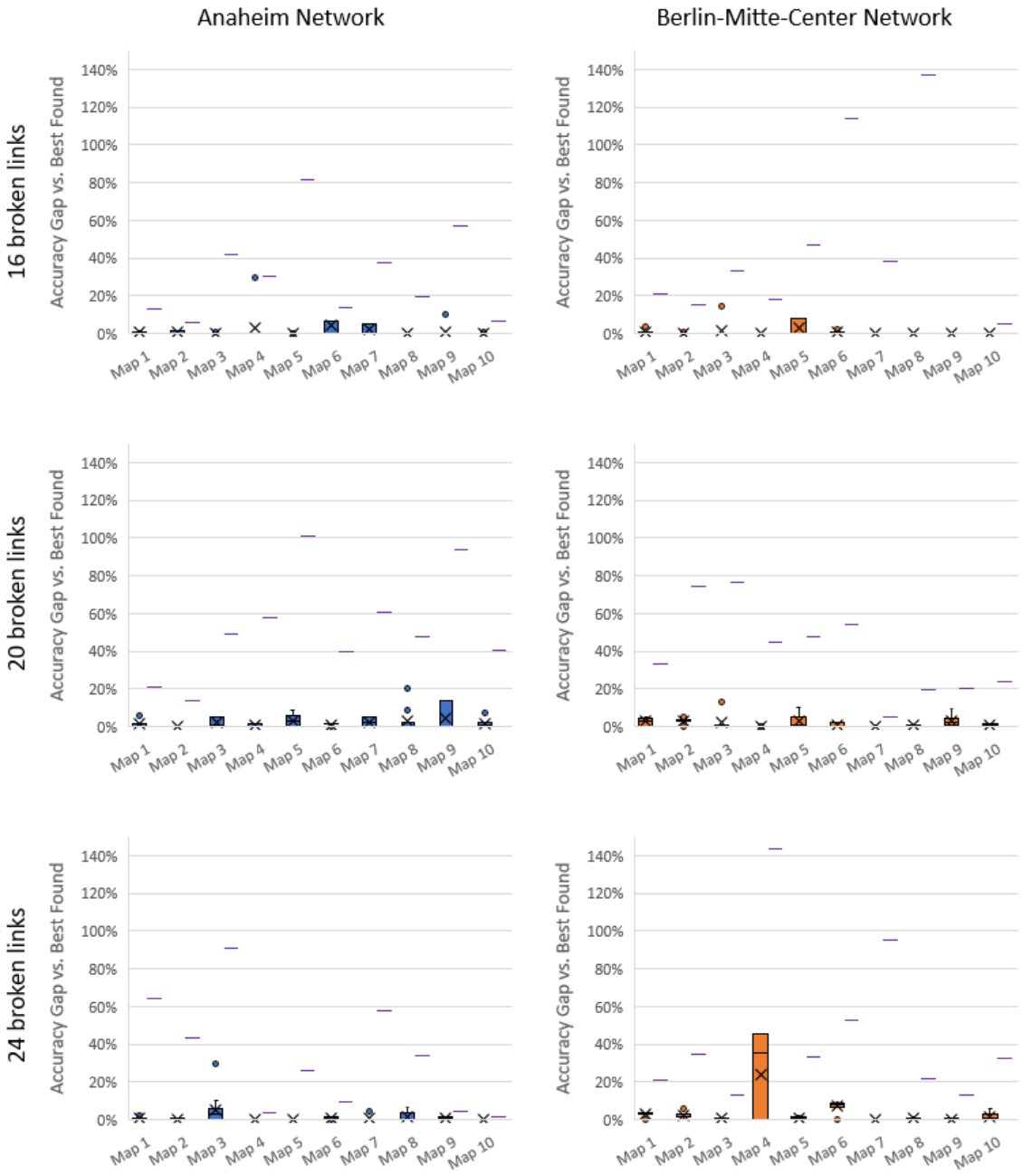

Figure 2.6: OBJ function accuracy gap versus best found OBJ function for ten instances (randomly generated maps) for Anaheim and BMC networks with 16, 20, and 24 broken links; the shaded area is the interquartile range (IQR) containing the median (horizontal line), the X indicates the mean, and outliers are shown as individual dots; standalone horizontal lines above and to the right of each bar indicate the BFS used to initialize SA

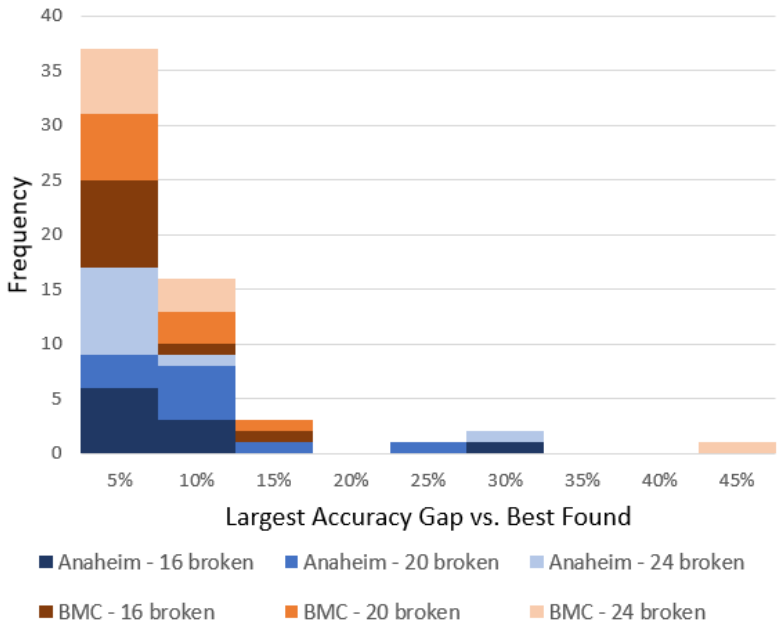

Figure 2.7: Largest accuracy gap versus best found; frequency over 60 instances (maps) tested, 10 random instances each on Anaheim and BMC networks for each of 16, 20, and 24 broken links

2.7 Obtaining Lower Bounds

While a feasible upper bound on travel delay can be established by any of the greedy methods discussed in §2.6.1, depending on the time available, establishing a valid and tight lower bound is challenging. When solving to optimality, there are two areas of significant computational effort: evaluating the TSTT at each of the 2^N possible repair states, and finding the repair sequence with the lowest objective function value given the TSTT at each possible repair state. If the TSTTs are precalculated, the optimization problem can be reformulated as follows:

$$\min_{\mathbf{p},\mathbf{y},\mathbf{z}} \quad \sum_{t=1}^{N} \Big(\sum_{s=0}^{2^N-1} TSTT_s p_s^t - TSTT_0 \Big) \sum_{b=1}^{N} y_b^t D_b \tag{2.12}$$

$$\text{s.t.} \quad \sum_{t'=1}^{t-1} y_b^{t'} = z_b^t \quad \forall b \in \{1,\ldots,N\}, t \in \{1,\ldots,N\} \tag{2.13}$$

$$\sum_{b=1}^{N} y_b^t = 1 \quad \forall t \in \{1,\ldots,N\} \tag{2.14}$$

$$\sum_{t=1}^{N} y_b^t = 1 \quad \forall b \in \{1,\ldots,N\} \tag{2.15}$$

$$p_s^t = \mathbf{1}\Big\{ \sum_{b=1}^{N} 2^{b-1} z_b^t = s \Big\} \quad \forall t \in \{1,\ldots,N\} s \in \{0,\ldots,2^N-1\} \tag{2.16}$$

$$p_s^t \in \{0,1\} \quad \forall t \in \{1,\ldots,N\}, s \in \{0,\ldots,2^N-1\} \tag{2.17}$$

$$y_b^t \in \{0,1\} \quad \forall b \in \{1,\ldots,N\}, t \in \{1,\ldots,N\} \tag{2.18}$$

$$z_b^t \in \{0,1\} \quad \forall b \in \{1,\ldots,N\}, t \in \{1,\ldots,N\} \tag{2.19}$$

where $\mathbf{z}_t$ is the state vector during stage t, and $\mathbf{p}_t$ is the induced state during stage t used to index the precalulated TSTT values. In this formulation, broken links are indexed $1, ..., N$, so that $\mathbf{z}_t$ vectors can be treated as binary representations of integer repair state for the purpose of indexing, as implemented in Equation 2.16. Because the TSTT depends on the full $\mathbf{z}$ state vector, rather than its components due to the nonlinearities inherent in TAP, this updated formulation induces an objective function with $N \cdot 2^N$ quadratic terms. From computational experiments, after presolving the 2^N TAP instances, even solving to optimality by brute force (computing the objective function for each of the possible $N!$ repair sequences, and choosing the lowest objective function) is faster than solving the induced IP. The time gap between the two methods grows with instance size. Furthermore, given precalculated TSTTs, the problem can be reformulated as a shortest path problem, with a computational complexity of $O(N \cdot 2^{N-1})$ rather than $O(N \cdot N!)$ for the enumeration approach ($N!$ possible sequences each of length N).

I can simplify the lower level problem by disregarding the effects of congestion and link capacities by setting travel times equal to free flow times for all links. In this simplification, the lower level problem reduces to shortest path, taking into account fixed toll and distance costs if desired. The Beckmann function in the objective function of the lower level problem, which was the source of nonlinearity, becomes linear when using a constant link performance function. Because travel time T_a no longer depends on x_a, the Beckmann function simplifies to the linear expression:

$$\sum_{a \in \mathcal{A} \cup \mathcal{R}} \int_0^{x_a^t} T_a(x) \mathrm{d}x = \sum_{a \in \mathcal{A} \cup \mathcal{R}} T_a \int_0^{x_a^t} \mathrm{d}x = \sum_{a \in \mathcal{A} \cup \mathcal{R}} T_a x_a^t.$$

The TSTT obtained by an all or nothing assignment, placing each unit of demand on its shortest path with travel times equal to link free flow times, will be less than or equal to the TSTT obtained by TAP for that repair state using the original link performance functions, assuming congestion effects are not permitted to be negative. This reflects the assumption that as more vehicles drive on a link, increasing congestion, travel times on that link also increase. In other words, for this inequality to hold, travel time using the original link performance function may never be lower than free flow time, a feature of any nondecreasing link performance function such as the BPR function. Therefore, the objective function value for any repair order will be the same or lower using the simplification than the original formulation. Of note, the bound obtained may validly be negative, since the objective function minimizes travel delay, rather than TSTT, over the repair horizon. Furthermore, the optimal objective value for the upper level problem using the simplified lower level problem will be lower than or equal to the optimal objective value for the original problem, indicating a valid lower bound on the original objective value. The simplified problem could be formulated and solved as a single level MIP. However, based on my computational experiments for the previous single level formulation, and tailored methods to solve shortest path problems, I instead solve the shortest path problem in

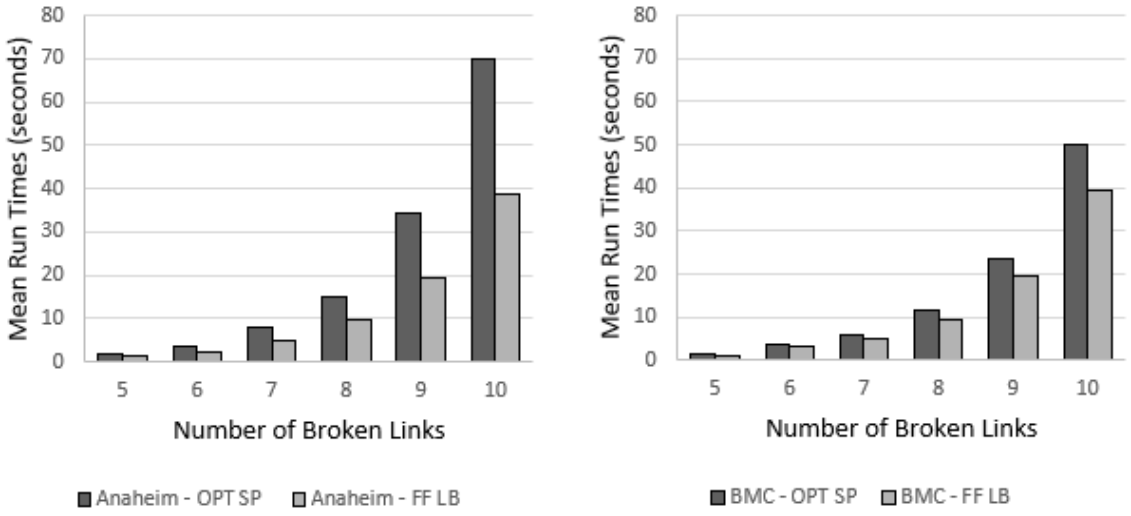

Figure 2.8: Mean run time comparisons on Anaheim and BMC for free flow lower bound vs. optimal shortest path method, 100 random instances at each number of broken links

the modified induced network after computing the TSTT resulting from all or nothing assignment flows for each possible repair state.

Though the lower bound obtained by relaxing the lower level problem and solving to optimality is valid, computational experiments show that it is not a tight bound, and indeed, in 80% of a total of 1200 experiments over two networks and six instance sizes, this lower bound is negative. On average, for the Anaheim network, finding the free flow bound (FF LB) took about 60% as long as finding the optimal solution by solving the induced shortest path problems (OPT SP), as depicted on the left side of Figure 2.8. For the BMC network, that average was just over 80%, indicating minimal time savings from using the simplified lower level problem. Notably, the mean run times to find the free flow lower bound on the Anaheim and BMC networks were very similar for each instance size tested, while the mean run times to find the optimal solution using the shortest path method were notably longer on the Anaheim network than on the BMC network. This result is reinforced logically, as both networks are of similar size, but the number of trips on the Anaheim network is about nine times higher than the number of trips on the BMC network. The num-

ber of trips in the network does not affect finding free flow times or the number of potential repair sequences – governed only by the number of broken links – but may affect the time required to solve each instance of TAP.

Due to the excessive time involved in the previous calculation method relative to the tightness of the resulting bound, I propose an alternate method of obtaining a heuristic lower bound. While not guaranteed by theory, the proposed method produces a valid lower bound in 90% of experiments conducted over two networks and varied instance sizes. Borrowing from the method used to establish heuristic upper and lower bounds for individual states during the beam search, I make use of the "best benefit" of repairing each link, as defined in §2.6.2, to obtain a heuristic lower bound. Using this definition, the best benefit of repairing a link is set to the greater of the benefit of repairing that link first or last in a repair order. The repair order used to obtain the lower bound is descending order of best benefit. However, because I seek a lower bound, not a feasible solution, I use the following formula to calculate the total travel delay:

$$\sum_{t=1}^{N} \min\left\{TSTT_1 - TSTT_0 - \sum_{t'=0}^{t-1}\sum_{b=1}^{N} y_b^t bb_b, 0\right\} \sum_{b=1}^{N} y_b^t D_b$$

where $TSTT_0$ is the TSTT before network disruption, and $TSTT_1$ is the TSTT during stage 1, which occurs after disruption and before any links complete repairs.

2.8 Comparison of Methods

Numerical experiments are conducted on a desktop computer running Ubuntu with a 16-core 3.4 GHz processor and 32 GB RAM. The test networks used are Anaheim and BMC from the Transportation Networks for Research repository (2022). The Anaheim network has 416 nodes, 914 links, and 38 zones, with just over 100,000 trips. The BMC network has 398 nodes, 871 links, and 36 zones, with nearly 11,500 trips. These networks were chosen for their similar size, in order to test algorithms on multiple networks with similar size but varied structure. All instances of TAP

are solved using Boyles' implementation of Dial's Algorithm B (Boyles et al., 2023; Dial, 2006). In order to obtain randomized problem instances, I first select a network node at random as the epicenter of the extreme event. Then, I sample a set of additional nodes based on proximity to the epicenter, and sample N broken links from those adjacent to sampled nodes, weighted by predisruption flow in order to ensure impactful network disruption. I set the disconnection parameter $Q = 10$, representing adding artificial links for any disconnected OD pair post-disruption, as well as for any OD pair where post-disruption travel times increase at least tenfold from the base case (pre-disruption).

In order to quantify relative solution quality and solution time of optimal (shortest path), beam search, and simulated annealing algorithms, I solve 100 random instances at each of eight through twelve broken links. In order to measure solution quality, I report an accuracy gap for each method, which captures the difference between the solution value found by that method and the best solution value (lowest objective value) found by any of the three methods for a particular instance. At 12 broken links, the mean solution time to optimality using shortest path is about 4.5 minutes on the Anaheim network and 3.4 minutes on the BMC network. Above this threshold, I solve 100 random instances by beam search and by simulated annealing for 13 through 15 broken links. The accuracy gap in these cases is measured against the best solution found (i.e. at least one of the heuristic methods will have a reported accuracy gap of zero for any given individual instance).

Figures 2.9 and 2.10 depict the accuracy gap and run time results on the Anaheim and BMC networks, respectively. For both networks, the accuracy gap means for the BS heuristic remain extremely close to zero for each instance size, with few high outliers. On both test networks, the top of the IQR for the SA heuristic remains below a 1.5% accuracy gap from the best found solution value for each instance size, indicating that for at least 75% of instances at each size, the SA heuristic is within 1.5% of the best found solution value. The mean for the SA heuristic, however, denoted by an X in the figure, is pulled as high as 2.7% (for 15 broken links on the BMC

network) because the distribution is right-skewed, with more high outliers (defined as points which are over $1.5 \times$ IQR larger than the top of the IQR) than the BS heuristic. Indeed, up to 13% of data points are considered outliers for some instance sizes, because the IQR is extremely small ($< 1.5\%$). These observations demonstrate that BS is more consistent than SA, though both achieve high accuracy in the majority of instances tested on both networks. Tabulated summary statistics are found in Appendix A, Tables A.1 and A.5.

On both the Anaheim and BMC networks, as instance size (number of broken links) increases, the mean and median run times increase monotonically for all three methods depicted. Because artificial links are added to the network on an as-needed basis for each random instance as discussed in §2.3, the effective size of the network for solving TAP varies with each individual instance, and not necessarily proportionately to the number of broken links. For this reason, while run time central tendency measures vary monotonically with instance size, it is unsurprising that outliers are evidenced on both networks for all three methods.

Above 15 broken links, I add in the six greedy methods in order to compare their performance against the simulated annealing heuristic for instance sizes of 16, 24, 32, and 48 broken links, instantiating 25 random instances at each instance size. Tabulated summary statistics for these experiments for both run times and accuracy gaps are found in Appendix A, Tables A.3 and A.7. Due to increasing computational time for the beam search heuristic as seen in Figure 2.11, I drop that heuristic for instance sizes over 24 broken links. For both methods, the run time distributions are generally somewhat right-skewed with the potential for a few high outliers. These outliers are non-trivial because in a practical scenario, only the solution to one instance is required, and if that instance is a high outlier compared to benchmarked run times, results will not be available when expected. An advantage to SA in this scenario is that the current BFS can be retrieved at any point in the run time, accepting the risk that a better solution may have been found if the algorithm were allowed to run for longer. In contrast, in order to function as intended, the bidirectional beam

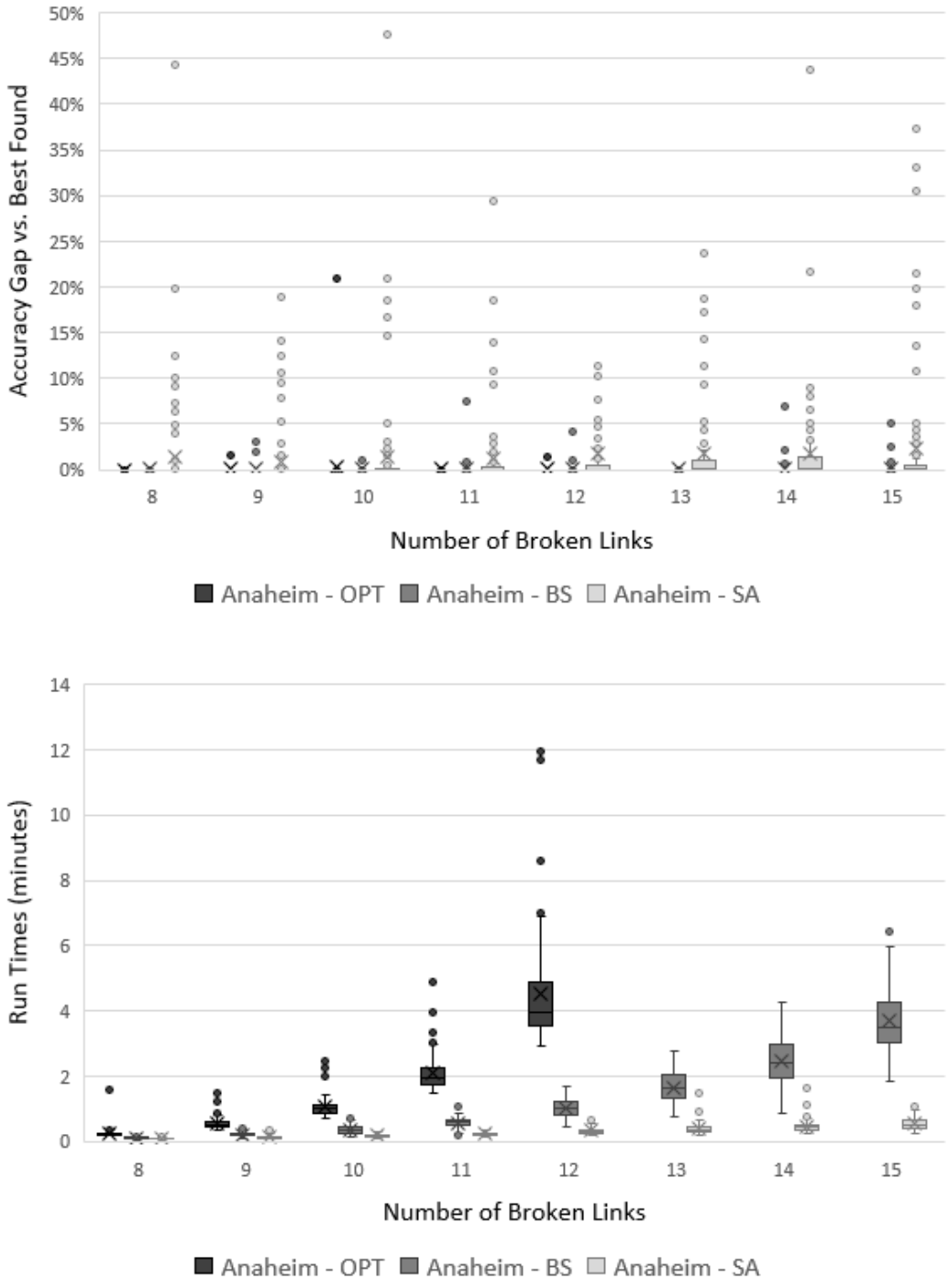

Figure 2.9: Anaheim accuracy gap and run time comparison graphs for 8–15 broken links, 100 random instances at each number of broken links

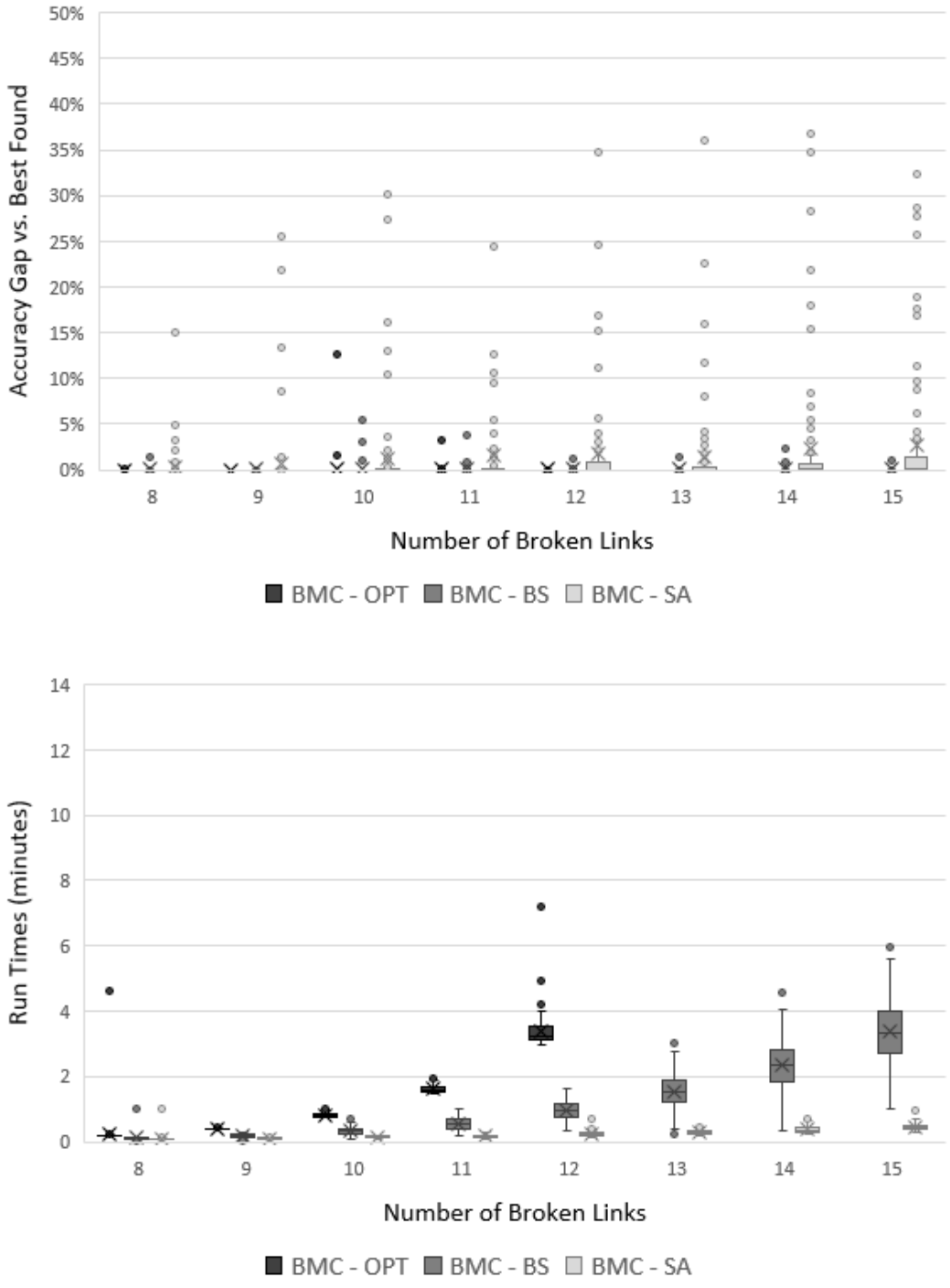

Figure 2.10: BMC accuracy gap and run time comparison graphs for 8–15 broken links, 100 random instances at each number of broken links

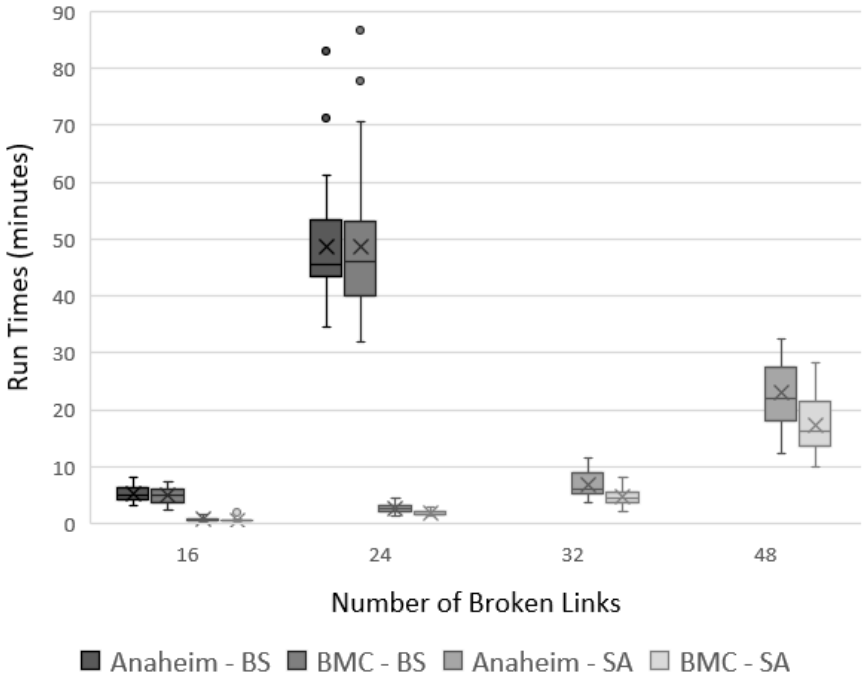

Figure 2.11: Run time comparison graph for 16, 24, 32, and 48 broken links, 25 random instances at each number of broken links

search needs to run to completion in order for the forward and backward fronts to meet in the middle, and ensure that promising open search fronts are explored.

In computing solution times for each method, I carefully ensure that all information used is "paid for" by each algorithm. Since the goal is to compare algorithms on equivalent instances and provide insights on relative strengths, weaknesses, and potential use cases, the only pre-processing not included in solution times is loading the network into the data structure and establishing artificial links. For example, evaluating the objective function value is not technically part of finding the SQG solution, because all a client actually needs if the method is already set in stone is the repair sequence. However, in order to use the SQG solution as the BFS to seed the beam search or simulated annealing methods, these methods require the objective function value and must pay for the time to not only find the solution sequence, but also evaluate it. Additionally, as mentioned when discussing greedy methods, every

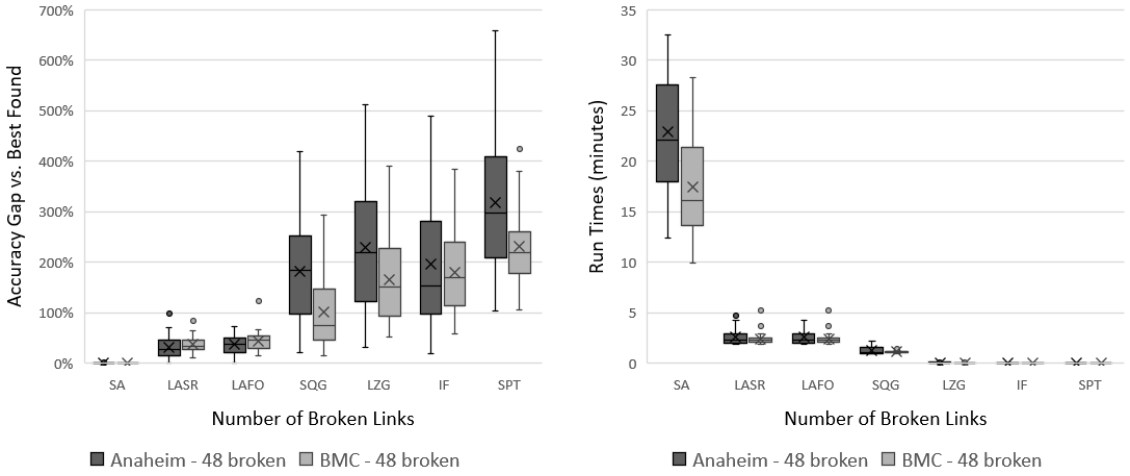

Figure 2.12: Accuracy gap and run time comparison graphs for 48 broken links, 25 random instances

method other than SPT requires knowledge of either network flows or TSTT in the undamaged network, and thus requires solving at least one instance of TAP. Due to this extreme diligence, and use of artificial links to account for network states which are not strongly connected while ensuring numerical stability, the solution times for all methods tend to be higher than reported in other studies.

Figure 2.12 depicts accuracy gap and run time comparisons at 48 broken links on both Anaheim and BMC networks. Of note, the SA heuristic finds the best solution of any method in all 25 instances on the BMC network, but only 23 out of 25 instances on the Anaheim network. In one of those instances, SA misses the best found value by about 2.7%, with the best solution found by LAFO, and LASR achieving an objective value between LAFO and SA. However, in the other instance, the best objective value is found by LAFO, LASR is 22% higher than LAFO, and SA is 43% higher than LAFO. These results demonstrate that SA is certainly the most accurate method in general, but it does not dominate LAFO and LASR in terms of accuracy gap. This is due to the randomness inherent in simulated annealing and the fact that LAFO and LASR use a different mechanism, approximation by sampling, than SQG and IF which are used to seed the SA heuristic. Tabulated summary

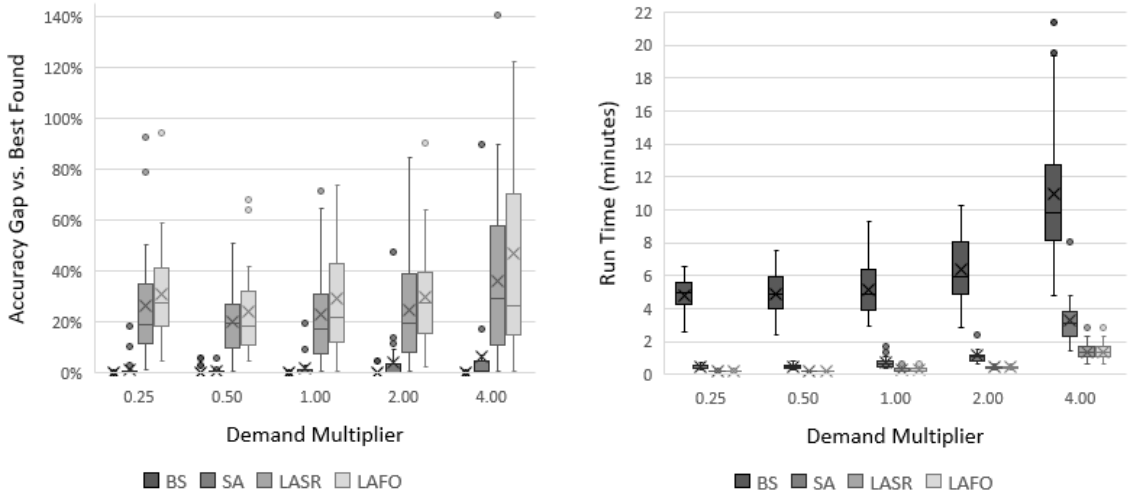

Figure 2.13: Accuracy gap and run time comparison graphs for varied demand multiples; Anaheim network with 16 broken links, ten random instances at each demand multiple

statistics for 48 broken links are found in Appendix A, Tables A.9 and A.11.

In terms of run time, SA evidences the longest and most variable run times. LZG, IF, and SPT run times are still negligible with 48 broken links, though their accuracy does not particularly recommend these methods for use if a more accurate method is tenable given the available time. LAFO and LASR provide a middle ground if the time necessary for SA is unacceptable based on operational constraints, significantly outperforming other greedy methods tested in terms of solution quality. The accuracy loss is certainly not negligible, with mean and median values for the two methods and networks falling between 23% and 44%, and the top of the IQR falling between 45% and 56%. However, at around one-fifth the run time of SA at this instance size, these methods may be operationally feasible when SA is not. Additionally, as noted by Rey and Bar-Gera (2020), LAFO and LASR can both be run, and the better solution accepted, in approximately the same time required to execute each method individually. The only additional time required is essentially the time to evaluate each solution in order to compare the objective values, since the algorithm time is primarily used to complete the average first order effects approximations.

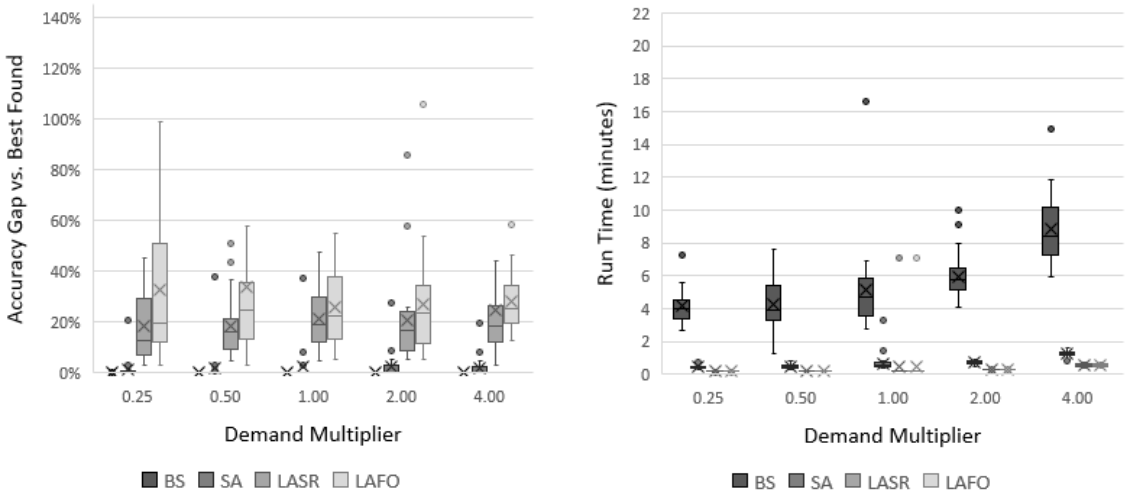

Figure 2.14: Accuracy gap and run time comparison graphs for varied demand multiples; BMC network with 16 broken links, ten random instances at each demand multiple

In addition to testing on multiple networks of similar size, I investigate the impact of inflated and deflated demand across both networks on run times and solution accuracy gaps. The results are depicted in Figures 2.13 and 2.14 for Anaheim and BMC, respectively, and tabulated summary statistics are found in Appendix A, Tables A.13 – A.28. Mean and median run times increase for all four methods investigated when demand is increased above baseline values. For the beam search and simulated annealing heuristics, neither increasing nor decreasing the demand multiplier significantly affects their accuracy gap distribution. Accuracy gaps for LAFO and LASR methods, however, exhibit inconsistent effects due to either increasing or decreasing the demand multiplier. Interestingly, when demand is increased four-fold, the median values remain close to those for the original parameters, but the means increase significantly, in this case reflecting that the upper half of the distribution is much more spread out, while the lower half does not change as significantly.

The definition of an "acceptable" solution time will vary with application, however, a time allowance of 24–72 hours is not unreasonable when considering long-term recovery planning. Starting with the same data as Figures 2.9 through 2.11, I

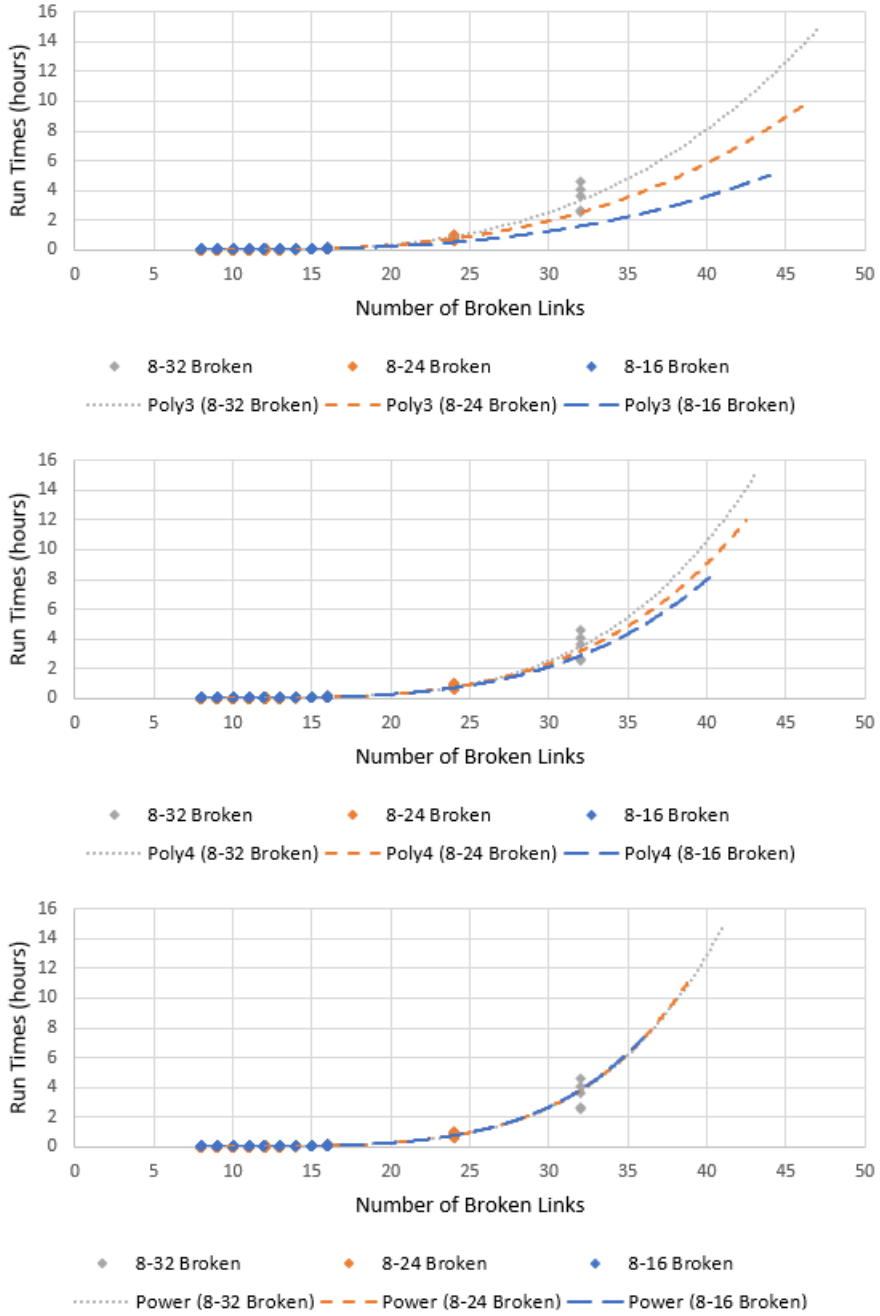

Figure 2.15: Run time curve fits for beam search on Anaheim network, testing out of sample performance

used least squares regression, excluding outliers as defined in the discussion on Figures 2.9 and 2.10, to fit run time curves for BS and SA to predict approximately how many links each algorithm can handle in 24 and 72 hours. For the beam search, I explored both fitting polynomials (third and fourth degree) and fitting a power function. I first fit a curve for each of the three functions using the beam search run time results on Anaheim for 8–16 broken links, then added in the data for 24 broken links and refit each function, then added five random instances at 32 broken links to further test out of sample performance, and refit each function a final time. The results of these experiments are shown in Figure 2.15. This procedure was critical to obtaining reliable estimates for the number of broken links feasible to process in 24 and 72 hours, since those values are out of the range of the testing conducted. When using a third degree polynomial fit, as seen in the top graph of Figure 2.15, the fit lines obtained by only considering instances up to 16 and even 24 broken links clearly miss the observed mean for instances with 32 broken links. The divergence between fit lines obtained with the restricted and expanded sample spaces decreases when using a fourth order polynomial, as observed in the middle graph of Figure 2.11, but is still easily noticeable within the testing range. In contrast, the three fit lines obtained for the power function are almost exactly superimposed on one another in the bottom graph of Figure 2.11 and pass through the center of the cluster at 32 broken links. This indicates that the fit obtained using data only up to 16 broken links provides the same predictive performance as the fit obtained using data up to 32 broken links when predicting the mean, and the prediction is accurate, at least to 32 broken links. This consistency provides an indicator that using the power function will result in reliable out of sample performance in terms of higher numbers of broken links.

Next, I conduct similar experiments with the collected simulated annealing run time data for Anaheim network. For SA, a third degree polynomial fit with intercept fixed at zero is a natural choice since the stopping criterion used is max iterations of $1.2N^3$, but I also explore second and fourth degree polynomials and the power function. I first fit a curve for each functions using the simulated annealing run time

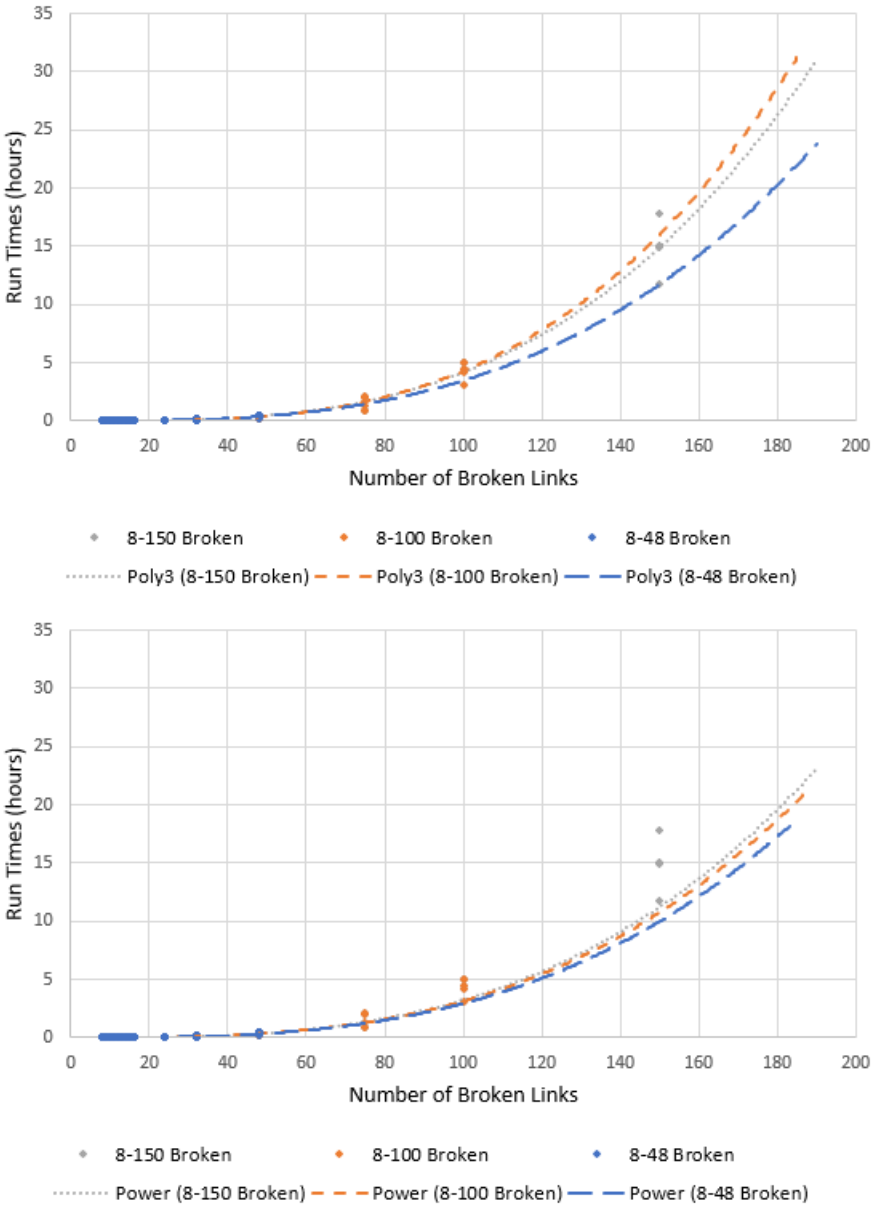

Figure 2.16: Run time curve fits for simulated annealing on Anaheim network, testing out of sample performance

results on Anaheim for 8–48 broken links, then added five random instances at 75 and 100 broken links and refit each function, then added five random instances at 150 broken links to further test out of sample performance, and refit each function a final time. Among the polynomial fits, the third degree polynomial clearly offered the best fits, and Figure 2.16 contrasts fitting to a third degree polynomial with fitting to a power function. While the power fit is more consistent in out of sample predictions, it systematically estimates too low of run times, with all three predictions falling below all realized values for 150 broken links. From the top graph in Figure 2.16, it is apparent that while a third degree polynomial results in a reasonable estimator, its accuracy does decay farther from the sampled area. Due to the underestimation evidenced by the power fit, I choose to use the third degree polynomial fit for approximating run times for higher numbers of broken links.

Using the selected functions, Figure 2.17 depicts run time curve fits for SA on Anaheim and BMC networks using the data described above to include the additional experiments at 32 broken links for the beam search and 75, 100, and 150 broken links for simulated annealing on both networks. Given a time allowance of 24 hours, the beam search heuristic is able to solve instances with up to about 45 broken links on the Anaheim or BMC network. In the same time period, the simulated annealing algorithm can solve instances with up to 175–185 broken links. Assuming a time allowance of 72 hours, the beam search method could only solve instances with up to about 55 broken links, while the simulated annealing algorithm tackles instances of up to 250–260 broken links, or about 24–29% of network links. Based on the consistency demonstrated by each method, I would recommend using the beam search heuristic over the simulated annealing heuristic if the time investment required for the beam search will not impact the start of reconstruction operations. If broken links can be identified very early into initial recovery operations, then significant time may be available for optimization in a real world scenario before long-term reconstruction assets can be deployed due to initial recovery operations and restricted access to affected areas.

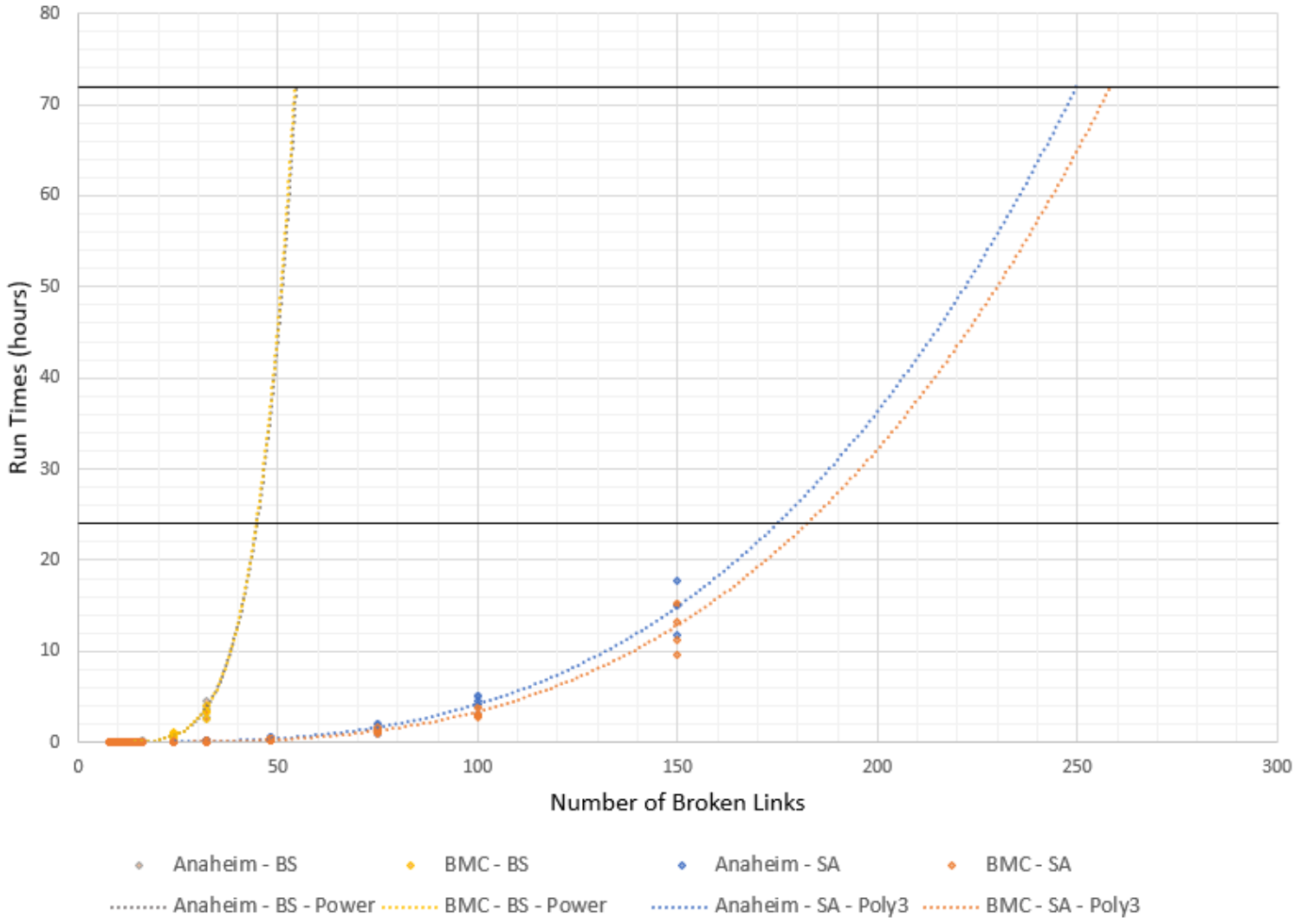

Figure 2.17: Run time curve fits for simulated annealing and beam search on Anaheim and BMC networks

2.9 Conclusions

In this chapter, I present a simulated annealing heuristic for the link repair sequencing problem in a damaged road network. Additionally, I compare an exact (shortest path) method, bidirectional beam search heuristic, simulated annealing heuristic, and six greedy heuristics, computationally testing on Anaheim and Berlin-Mitte-Center networks. My experiments indicate that the bidirectional beam search provides the highest quality solutions in the majority of experiments, but the simulated annealing heuristic runs significantly faster, with minimal loss in solution quality. Across the board, none of the greedy heuristics except for LASR and LAFO using Rey and Bar-Gera's approximation method reliably came within a 50% accuracy gap from the beam search or simulated annealing objective function. While LASR and LAFO are fast methods of obtaining a reasonable repair sequence, beam search and simulated annealing both obtain much higher quality solutions and can solve instances with up to 55 and 220–250 broken links, respectively, in under 72 hours.

An area for future research would be to explore which greedy heuristic most reliably designates the optimal (or near optimal) first link for repair. While greedy heuristics performed significantly sub-optimally when finding the full optimal repair sequence, if a greedy heuristic could find the optimal first link to repair with some high degree of accuracy, say 95%, the repair assets could begin work almost immediately on that link, and the rest of the sequence could be optimized using a heuristic with a much higher solution quality, but longer solution time, while the first repair is completed.

Chapter 3: Scheduling Disaster Recovery using Multiple Identical Crews

3.1 Introduction

This chapter focuses on the assignment and ordering of broken links in a road network for repair using multiple identical work crews. In the single-crew formulation without preemption, link repairs are started and completed in a strict sequence, and an analogue to Bellman's optimality principle applies such that once a subset of repairs are complete, optimizing the repair sequence of the remaining links is independent of the completion order of the first links (Gokalp et al., 2021). Given the urgency of regaining network functionality, if multiple repair assets are available, then the network can be brought to full operational performance more rapidly. However, the modification from single-crew to uniform multi-crew (where all crews have identical properties) is not trivial in terms of solution methods. The analogue to Bellman's optimality principle is lost, since the total system travel time (TSTT) at a repair state is multiplied by the duration in that repair state in the objective function, which is no longer independent of the repair sequence (since link repairs can now start partway through other link repairs).

In this chapter, I continue to use the same assumptions as in the previous chapter including that broken links are unusable until fully repaired and repair durations and known and fixed, but the assumptions differ in the key point that repairs of multiple links are allowed to proceed in parallel, with each work crew repairing a different link, rather than in strict sequence. The objective is to minimize the total travel delay over the repair horizon, assuming that traffic reaches user equilibrium in a negligible amount of time after each repair is completed. I establish that this problem is NP-hard by restriction by proving that minimizing weighted completion time on identical parallel machines reduces to the multi-crew scheduling problem. I adapt both the existing sequential greedy method and the sequential simulated annealing

method for the multi-crew formulation, and explore alternative neighborhoods for the simulated annealing algorithm. For the other greedy methods identified in Chapter 2, I apply post-processing to the single-crew sequence in order to assign each link in the sequence to the first available crew, sequentially. Additionally, I develop a novel method of solving specifically the multi-crew scheduling problem, decomposing the set of broken links $\mathcal{B}$ into clusters assigned to each repair asset, and subsequently optimizing only within each repair crew.

The remainder of this chapter is broken into six sections. §3.2 reviews previous and related literature. In §3.3, I define the problem formulation and the solution space. §3.4 compares this problem to a standard problem from machine scheduling literature to establish the multi-crew problem as NP-hard. §3.5 presents and discusses solution methods, including adapted sequencing methods from the single-crew problem, novel simulated neighborhoods, and decomposition methods. In §3.6, I report computational results, and the chapter concludes with key findings and directions for future research.

3.2 Literature Review

This chapter addresses methods of balancing run time and solution quality when scheduling links for repair by multiple identical crews in a damaged transportation network, with TSTT representing the functionality of any network state. Multiple authors have looked at this general formulation of the problem and proposed heuristics for finding the optimal repair sequence (Rey and Bar-Gera, 2020; Gokalp et al., 2021), but I have not encountered a review which focuses on comparing available proposed methods on equivalent problem instances, directly obtains heuristic solutions to the multi-crew problem, and enunciates the trade-offs involved. I turn to the general scheduling literature in order to frame the problem within that field and examine whether generic scheduling techniques can be adapted to solve this particular problem, as well as adapting sequencing heuristics for the multi-crew formulation.

Choosing the single performance measure of TSTT, I focus on algorithmic contributions in order to balance solution time and quality for the multi-crew problem. These in turn may also be useful for solving similar or extended formulations. In Chapter 2, I review various choices of performance measures and formulations spanning maximum number of independent pathways, amount of demand met, and combinations of functional, topological, and economic performance metrics (Chen and Miller-Hooks, 2012; Merschman et al., 2020; Zhang et al., 2017, 2018a). Furthermore, several authors have used performance metrics related to TSTT, but as a portion of a multi-stage framework (Bocchini and Frangopol, 2012; Vugrin et al., 2014; Ye and Ukkusuri, 2015).

A variety of greedy methods have been proposed or recommend themselves to potential use in solving the repair sequencing problem, and I extend their use to the multi-crew formulation in order to compare against other proposed and developed heuristics. Gokalp et al. (2021) use three greedy methods to benchmark their beam search heuristic for the sequencing problem: SQG, LZG, and IF. I adapt the SQG method to account for links completing repairs in a different order than repairs are started due to using multiple repair crews. The LZG and IF methods require no adaptation, since the sequencing is completed myopically, and I post-process the resulting sequence to obtain a multi-crew schedule by assigning each link in the sequence to the earliest available crew, as done by Rey and Bar-Gera (2020), and described by Anderson et al. (2003) in a multiple identical machine scheduling context.

Rey and Bar-Gera (2020) assume that links are grouped into projects *a priori* and use a sampling method presented by Rey et al. (2019) to approximate the first order effects of each project. Then, projects are ordered greedily based on the LAFO or LASR heuristic using the approximated average first order effects. As a comparison case, they use the simple and time-effective SPT heuristic. In their paper, each resultant ordering (sequence) is post-processed as described above to obtain a schedule for multiple crews.

Turning to general scheduling literature, Anderson et al. (2003) specifically address the application of local search heuristics to machine scheduling problems. This application is particularly pertinent to the development of promising simulated annealing neighborhoods for the present problem since it shares characteristics with traditional scheduling problems from the literature. Complexity results for related problems in machine scheduling (Bruno et al., 1974; Lenstra et al., 1977; Garey and Johnson, 1990) as well as heuristics and polynomial time approximation schemes (PTAS) for related parallel machine problems (Kawaguchi and Kyan, 1986; Smith, 1956; Skutella and Woeginger, 2000) additionally inform my search for tailored heuristic solutions.

3.3 Problem Formulation

I use the same objective function to be minimized as in the single-crew formulation, which is total travel delay over the repair horizon. In this formulation, as in the sequencing formulation, TSTT is used to represent the functionality of any network state. Consider a network $G = (\mathcal{N}, \mathcal{A})$, with a set of broken links $\mathcal{B}$. With only one repair asset (work crew) available, exactly determining the optimal repair sequence requires examining $|\mathcal{B}|!$ possible sequences. I demonstrate in the proofs of Propositions 3.1 and 3.2 that for the multi-crew formulation the degrees of freedom are reduced such that with K identical crews, solving to optimality requires examining only $|\mathcal{B}|!/K!$ unique repair sequences, rather than $|\mathcal{B}|!$. However, for realistically-sized transportation networks, solving to optimality still becomes quickly intractable.

The model is once again a bilevel optimization problem, where the upper level objective is to minimize the total travel delay over the full repair horizon, and the lower level problem is the static TAP of computing equilibrium flows for a given repair state. I use the same notation as described in Chapter 2. I maintain the assumptions that if a link is broken (damaged) it cannot be traversed until fully repaired, the link repair durations are fixed and known *a priori*, and user equilibrium is reached after

each repair. I assume that each crew works on a single link repair at a time until its completion, without preemption and without idle time. Because there are N broken links, there are once again N network stages in a repair schedule as the N broken links complete repairs. These non-terminal stages are indexed by $t \in \{1, \ldots, N\}$. Since there are multiple crews, it is possible for two repairs to finish simultaneously. In this case, the duration of one stage would simply be zero, and two repair schedules would have identical objective values.

The sequence of repair completions is defined by the binary variables y_b^t, where $y_b^t = 1$ iff link b completes repairs during stage t. The variable z_b^t equals 1 iff link b completed repairs prior to stage t and is therefore usable during stage t. Crew assignments are defined by the binary variables c_b^k where $c_b^k = 1$ iff link b is repaired by crew k. This formulation does not explicitly define the sequence of what order in which to *start* repairs, but that sequence can be recovered from the crew assignments and repair completion sequence defined by $\mathbf{y}$. For a given link a in stage t, the flow on that link is x_a^t, and h_π^t represents an equilibrium flow on path π. During stage t, the total system travel time is then $\sum_a x_a^t T_a(x_a^t)$, and stage t lasts for $F_t - F_{t-1}$ days where F_t is the completion time of the tth link to complete repairs, and $F_0 = 0$. The shorthand $TSTT_t$ will again be used to compactly denote the TSTT during stage t, with $TSTT_0$ being the TSTT before disruption, equivalently $TSTT_{N+1}$, after all repairs are complete.

The following bilevel optimization expresses minimizing the total travel delay over the repair horizon, where M is a sufficiently large constant:

$$\min_{\mathbf{x},\mathbf{y},\mathbf{z},\mathbf{c}} \quad \sum_{t=1}^{N} \Big(\sum_{a \in \mathcal{A} \cup \mathcal{R}} x_a^t T_a(x_a^t) - TSTT_0 \Big) [F_t - F_{t-1}] \tag{3.1}$$

$$\text{s.t.} \quad F_t = \sum_{k=1}^{K} \sum_{b \in \mathcal{B}} c_b^k y_b^t \Big[\sum_{t'=1}^{t} \sum_{b' \in \mathcal{B}} c_{b'}^k y_{b'}^{t'} D_{b'} \Big] \quad \forall t \in \{1, \ldots, N\} \tag{3.2}$$

$$F_0 = 0 \tag{3.3}$$

$$\sum_{k=1}^{K} c_b^k = 1 \quad \forall b \in \mathcal{B} \tag{3.4}$$

$$\sum_{t'=1}^{t-1} y_b^{t'} = z_b^t \quad \forall b \in \mathcal{B}, t \in \{1, \ldots, N\} \tag{3.5}$$

$$\sum_{b \in \mathcal{B}} y_b^t = 1 \quad \forall t \in \{1, \ldots, N\} \tag{3.6}$$

$$\sum_{t=1}^{N} y_b^t = 1 \quad \forall b \in \mathcal{B} \tag{3.7}$$

$$y_b^t \in \{0, 1\} \quad \forall b \in \mathcal{B}, t \in \{1, \ldots, N\} \tag{3.8}$$

$$z_b^t \in \{0, 1\} \quad \forall b \in \mathcal{B}, t \in \{1, \ldots, N\} \tag{3.9}$$

$$c_b^k \in \{0, 1\} \quad \forall b \in \mathcal{B}, k \in \{1, \ldots, K\} \tag{3.10}$$

where each $\mathbf{x}_t$ is optimal to:

$$\min_{\mathbf{x}_t,\mathbf{h}_t} \quad \sum_{a \in \mathcal{A} \cup \mathcal{R}} \int_0^{x_a^t} T_a(x) \mathrm{d}x \tag{3.11}$$

$$\text{s.t.} \quad \sum_{\pi \ni a} h_\pi^t = x_a^t \quad \forall a \in \mathcal{A} \cup \mathcal{R} \tag{3.12}$$

$$\sum_{\pi \in \Pi_{rs}} h_\pi^t = d_{rs} \quad \forall (r, s) \in \mathcal{Z}^2 \tag{3.13}$$

$$h_\pi^t \geq 0 \quad \forall \pi \in \Pi \tag{3.14}$$

$$x_b^t \leq M z_b^t \quad \forall b \in \mathcal{B} \tag{3.15}$$

The upper level objective 3.1 is to minimize the total travel delay (equivalently, total TSTT) over the repair horizon, captured by multiplying the system cost for each stage by the stage duration. The first upper level constraint defines the completion

time of the tth link to complete, and the second constraint defines the boundary condition of F_0. The third upper level constraint ensures that each link is only assigned to one of the K crews. The fourth upper level constraint ensures that each link is available for use only after it is repaired. The fifth and sixth upper level constraints ensure that only one link is repaired per stage and each link is repaired at some stage, respectively. The last three upper level constraints define $\mathbf{y}$, $\mathbf{z}$, and $\mathbf{c}$ as sets of binary variables. The lower level problem (Equations 3.11 – 3.15) is the standard user equilibrium formulation of the static TAP, with the addition of the artificial links and a constraint to ensure the flow on broken links is equal to zero.

As formulated, exactly solving the above bilevel optimization problem would require enumerating all possible unique repair sequences, and evaluating each to determine the optimal sequence, including solving TAP at each repair state encountered. The repair state at any time t is characterized by a vector $\mathbf{z}^t$ of length N, where each entry is either 1, if the indexed link is repaired, or 0, if the indexed link is not yet fully repaired. Minimizing total TSTT over the repair horizon is analogous to minimizing weighted completion time in an identical parallel machines scheduling problem, with the complicating factor of state-dependent weights. The computational time to calculate each of the 2^N state-dependent weights (TSTTs) is the time required to solve TAP for each state for the full network, with set of nodes $\mathcal{N}$, set of links $\mathcal{A}$, and set of zones $\mathcal{Z}$. The computational time to evaluate the objective function for each unique repair sequence, once weights are established, depends only on the number of broken links, $|\mathcal{B}| = N$. The number of unique repair sequences (start orders) is demonstrated in Propositions 3.1 and 3.2.

Proposition 3.1. *The single-sequence repair start order fully defines a K-crew non-delay schedule.*

Proof. A non-delay schedule is one in which no crew is allowed to be idle while a job is waiting to be processed (Pinedo, 2016). Given the order in which each link begins to be repaired, the first K links immediately begin repair, one by each of the K crews,

at time 0. If these K links did not begin to be repaired at time 0, either the repair start order would not be respected or the resulting schedule would not be non-delay.

After time 0, in order to both adhere to the repair start order and maintain a non-delay schedule, as soon as a crew completes a link repair, it immediately begins to repair the next link in the repair start order which is not yet underway. □

Proposition 3.2. *Using the single-sequence representation defined above, there are $N!/K!$ potential non-delay repair schedules for N broken links and K identical crews.*

Proof. By Proposition 3.1, each single-sequence repair start order fully defines a K-crew non-delay schedule. There are $N!$ ways to order N unique items. Therefore, there are $N!$ repair start orders. However, because the crews are identical, the label assigned to a particular crew is arbitrary. Therefore, the first K links in any repair start order, which all start repair at time 0, can be sorted by increasing repair duration without affecting the repair schedule.

More precisely, the link in the $K + 1$ position in the repair start order will always be assigned to the same crew as the link among the first K links with the shortest repair duration in order to maintain the non-delay property and respect the repair start order. Therefore, for a given set of K links which comprise the first K elements of a repair start order and begin repairs at time 0, there are $K!$ possible orderings of those K links which result in the same non-delay repair schedule (since crew labels are arbitrary). Therefore, while there are $N!$ single-sequence repair start orders, there are only $N!/K!$ potential non-delay schedules. □

3.4 NP-Hardness

Now that I have laid out the problem formulation and discussed some of the computational challenges of an exact solution method, I formally establish that finding that optimal repair schedule is NP-hard, even without the additional calculations required to find the 2^N TSTTs. In order to establish that the problem is NP-hard,

I start with a standard problem within scheduling – minimizing the sum of weighted completion times on identical parallel machines, where the number of machines is a problem input. To represent this problem compactly, I refer to it as $P||\sum w_b C_b$ using the three field classification $\alpha|\beta|\gamma$ common in scheduling literature and introduced by Graham et al. (1979), with β being an empty set in this instance. In the first field, $\alpha = P$ indicates the machines used are identical. The second field is unused, but would indicate further restrictions on the problem, such as precedence constraints or job release times. Finally, $\gamma = \sum w_b C_b$ identifies the objective function to be minimized, which is the sum of weighted completion times over all jobs $b \in \mathcal{B}$.

Theorem 3.3. *$P||\sum w_b C_b$ reduces to multi-crew scheduling where state-dependent TSTTs are already calculated a-priori.*

Proof. Suppose that N jobs ($b = 1, ..., N$) need to be scheduled on K identical machines ($i = 1, ..., K$) where K is an input parameter, and each job has a duration p_b and weight w_b. The objective function to be minimized is $\gamma = \sum w_b C_b$, the sum of weighted completion times. I will show that $P||\sum w_b C_b$ is a special case of the multi-crew scheduling problem defined in §3.3, proving Theorem 3.3 by restriction.

In the multi-crew problem, the objective function is to minimize

$$\sum_{t=1}^{N} \Big(\sum_{a \in \mathcal{A} \cup \mathcal{R}} x_a^t T_a(x_a^t) - TSTT_0 \Big) [F_t - F_{t-1}] = \sum_{t=1}^{N} \big(TSTT_{\mathbf{z}^t} - TSTT_{N+1} \big) [F_t - F_{t-1}],$$

where $TSTT_{\mathbf{z}^t}$ is the TSTT corresponding to repair state $\mathbf{z}^t$, and $TSTT_0$ is the TSTT before disruption, which is equal to $TSTT_{N+1}$, the TSTT after all links are repaired. I assume for this proof that the TSTT values for all 2^N possible repair states are calculated *a-priori*. This formulation of the objective function sums over the travel delays during time periods $1, ..., N$ depicted as vertical bars in the left side of Figure 3.1, but can be reformulated to instead sum over horizontal bars, as in the right side of Figure 3.1. The reformulated equivalent objective function to be minimized is

$$\sum_{t=1}^{N} \big(TSTT_{\mathbf{z}^t} - TSTT_{\mathbf{z}^{t+1}} \big) F_t.$$

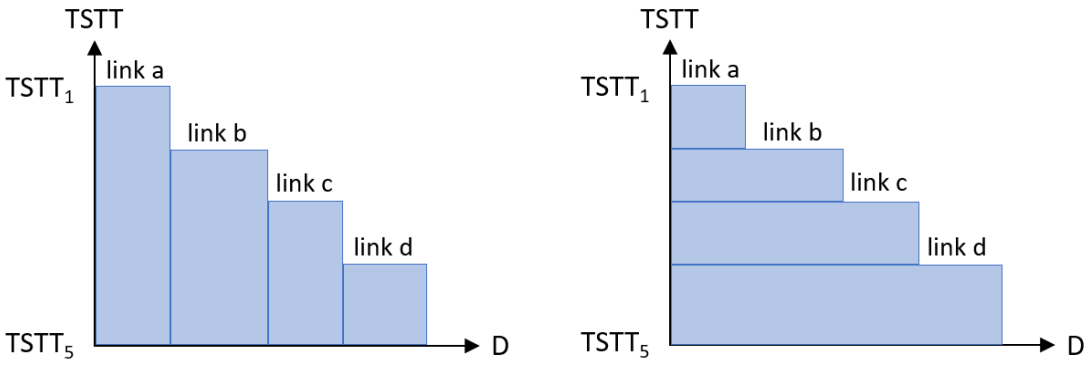

Figure 3.1: The shaded area is total travel delay over the repair horizon

From §3.3, F_t is the completion time of the tth link to be repaired. There is a one-to-one mapping $f : t \to b$ between the time indices in the multi-crew problem and the job indices in $P||\sum w_b C_b$ for any given repair schedule. Therefore, I assign $C_b = F_{f(t)}$. If I restrict the multi-crew problem such that the improvement in TSTT from repairing a link b (equivalently completing job b) is not dependent on repair state, but fixed at a constant $\Delta TSTT_b$, then $\left(TSTT_{\mathbf{z}^t} - TSTT_{\mathbf{z}^{t+1}}\right) = \Delta TSTT_{f(t)}$. I assign $w_b = \Delta TSTT_{f(t)}$. Combining, the objective function for the restricted problem is now equal to the $P||\sum w_b C_b$ objective function:

$$\sum_{t=1}^{N} \Delta TSTT_{f(t)} F_t = \sum_{b=1}^{N} w_b C_b.$$

Therefore, by restriction, $P||\sum w_b C_b$ reduces to multi-crew scheduling where state-dependent TSTTs are already calculated *a-priori*. □

Because $P||\sum w_b C_b$ is strongly NP-hard when the number of machines (crews) is an input parameter (Garey and Johnson, 1990), and the above reduction is completed in polynomial time for any instance, Corollary 3.4 immediately follows.

Corollary 3.4. *Multi-crew scheduling where state-dependent TSTTs are already calculated a-priori is strongly NP-hard.*

3.5 Solution Methods

Next, I examine solution methods. I begin with a short discussion of heuristic methods from machine scheduling, and adapt methods from the single-crew sequencing problem for application to the multi-crew scheduling problem. Subsequently, I propose and evaluate novel simulated annealing neighborhoods specifically for use with multiple crews. Finally, I propose two methods of decomposing links into crews *a priori*, as well as four methods of sequencing links within each crew.

3.5.1 Heuristic Methods from Machine Scheduling

In order to provide a starting point for solving the multi-crew scheduling problem, I first examine heuristic methods applied to $P||\sum w_b C_b$ in the literature. A basic heuristic is to apply the weighted shortest processing time (WSPT) rule, also known as Smith's ratio, which optimally solves $1||\sum w_b C_b$ by ordering jobs by non-increasing w_b/D_b ratios (Smith, 1956). Sahni (1976) and Kawaguchi and Kyan (1986) extend this concept to provide a PTAS for $Pm||\sum w_b C_b$ and a performance ratio guarantee for $P||\sum w_b C_b$, respectively. Skutella and Woeginger (2000) further establishes a PTAS for $P||\sum w_b C_b$. However, since the weights in the multi-crew scheduling problem are state-dependent, rather than only link or job dependent, these methods do not translate to performance guarantees for the multi-crew scheduling problem, since any uniform weight w_b used for a link repair would be an approximation. In fact, one of the heuristics proposed by Rey and Bar-Gera (2020) approximates this weight (the first-order effect of repairing a particular link) by sampling, and orders projects greedily by the resulting Smith's ratio, in effect applying the WSPT rule using approximate weights. I do not adapt the significantly more complex PTAS established for $P||\sum w_b C_b$ to the multi-crew scheduling problem.

3.5.2 Adapting Existing Sequencing Methods

Additionally, I adapt existing methods applied to the single-crew sequencing problem, as the single-crew problem is a special case of the multi-crew problem. However, not all methods can be effectively generalized from single- to multi-crew because Bellman's optimality principle is lost in the generalization. The bidirectional beam search heuristic developed by Gokalp et al. (2021) is one such method which fundamentally relies upon this principle, and does not translate readily to the multi-crew problem. Strictly greedy methods, however, which order links for repair based on a performance measure calculated *a-priori*, rather than sequentially as the repair order is constructed, transfer more readily to the multi-crew problem. Therefore, I consider and evaluate five strictly greedy sequencing methods from the literature. The sequential greedy method can also be adapted for multi-crew by taking into account repair start and completion times, rather than simply repair order. The simulated annealing heuristic parameterized for the single-crew problem, while not a greedy method, can be modified to solve the multi-crew method, taking into account the more complex structure, and examining the effect of the neighborhood definition.

3.5.2.1 Post-processing Greedy Sequencing Methods

The SPT heuristic is the simplest heuristic proposed in that the only information required is the link repair duration for each link. SPT simply orders the links from shortest repair time to longest repair time and repairs the links in that order. The IF heuristic is also a simple indexing, however, since links are ordered for repair in descending order of pre-disruption flow, a single TAP must be solved to obtain the link flows required. LZG repairs links in order of the greatest immediate benefit, defined by decrease in TSTT due to repairing a link while all others remain broken divided by that link's repair duration.

The approximation methods proposed by Rey and Bar-Gera (2020) first solve TAP for a subset of all possible network states during the repair sequence, and then

use this sample to approximate the average first-order effects of repairing each link at any point in time. The sampling method is detailed in Rey et al. (2019) and requires solving a number of TAPs quadratic in the number of projects. In Rey and Bar-Gera (2020), the objects sequenced are projects, rather than links, but the method can be applied to projects of a single link each, in order to compare to other methods on equal footing. The first approximation method ranks projects and greedily sequences them in descending order of largest average first-order effects. The second approximation method takes into account repair duration, as well as first-order effects, ranking projects based on descending largest approximated Smith's ratio, and then sequencing greedily. Smith's ratio, in this case, is implemented as the ratio of approximated first-order effects over repair duration.

For each of the above methods, once obtaining the sequence, I assign one of the first K links in the sequence to each one of the K crews. Then, I assign each subsequent remaining link in the sequence to the crew with the shortest already assigned work duration. I refer to this assignment procedure as post-processing.

3.5.2.2 Multi-crew Sequential Greedy Method

For the single-crew problem, SQG constructs the repair sequence myopically, by choosing the link to repair which provides the greatest immediate benefit at each stage, defined by decrease in TSTT due to repairing the link over the link repair duration. At each stage in building the repair sequence, the link is chosen which maximizes this value. At any point in the repair order in the single-crew setting, the baseline TSTT is the TSTT after repairing all previous links already chosen for repair. The comparison TSTT for each link yet to be repaired is the TSTT given that all previous links and the newly selected link are repaired. The next link is chosen for repair based on the greatest decrease from the current baseline.

In the multi-crew setting, the first K links (the first link for each of the K crews) are chosen based on the greatest immediate benefit, with the baseline being

no repaired links, and the comparison being a single link repaired. This selection method is a simplification, since naturally only one of the first K links can actually complete first, unless there are multiple identical repair durations among them. The underlying simplifying (though inaccurate) assumption is that the next link chosen for repair will complete before any other links already in progress (being worked on by other crews). While inaccurate, this simplification is necessary for the heuristic in order to avoid significantly increasing the number of required TAPs. Subsequently, at each stage, the crew with the shortest combined work duration assigned so far is chosen for the next link assignment, and the repair state at that time is established by examining completion times of links already assigned to crews. This TSTT forms the baseline for comparison, and each remaining eligible link's immediate benefit is calculated using the assumption that it would complete before any additional links from other crews complete.

3.5.2.3 Adapting Simulated Annealing for Multi-crew

In adapting the single-crew simulated annealing method for the multi-crew application, the primary areas of focus are the definition of a neighborhood and the size of the solution space. In representing a candidate solution, I continue to use a vector of length N, which is a sequence of links for repair representing their start (though not necessarily completion) order. In order to calculate the multi-crew objective function, the links are assigned in sequence order to the first open crew, in the same manner as post-processing a greedy solution as discussed above. In this data representation, the first K links in any sequence are interchangeable without affecting the objective function. Therefore, the solution space for a problem with N broken links decreases as K increases. For this reason, while I used a stopping criterion of max iterations equal to $1.2N^3$ for the single-crew heuristic, based on this insight as well as computational experiments, I use a stopping criterion of max iterations equal to $1.5(N-K+1)^3$ for the multi-crew heuristic.

To obtain a performance baseline for exploration of other neighborhoods, I

first maintain the definition of a 1-neighborhood used in the single-crew heuristic. The impact of swapping a single link in the repair sequence (other than the last link in the sequence) with the link immediately following, however, changes significantly. In the multi-crew setting, swapping the start order of two links adjacent in the repair start order will generally result in one or both links swapping to a different crew. Additionally, if the swapped links have different repair durations, the start and completion times as well as potentially repair crew assignments of later links (links occurring in the repair order after those swapped) will additionally shift. Therefore, while the neighborhood definition is the same, the impact of a single swap on the repair schedule is significantly greater in the multi-crew setting.

I hold constant the parameter choices of initial temperature, exploration criterion, acceptance criterion, temperature length, cooling scheme, and temperature restart from the single-crew parameterization in Chapter 2. I choose the initial temperature, T_0, to obtain an initial probability of approximately 10% to accept a move which increases the objective function by 10%. I use a variant on the Lundy-Mees cooling scheme where $T_{i+1} = T_i/(a + b \times T_i)$, with $a = b = 1$ as proposed by Szu and Hartley (1987), with a temperature length of one, indicating temperature updates every iteration, and no temperature restarts. The exploration criterion used at each step is to randomly select a proposed new solution from the current 1-neighborhood. For acceptance criterion, I use the generalized simulated annealing variant proposed by Bohachevsky et al. (1986), with $g = -1$ and $\beta = (1/T)^{2/3}$, resulting in the formula

$$p_{GSA} = \begin{cases} 1 & \text{if } \Delta(s', s) \leq 0 \\ \exp\big(-\frac{\Delta(s',s)}{f(s)\times T^{2/3}}\big) & \text{otherwise.} \end{cases}$$

Finally, since the multi-crew performance of the greedy algorithms does not directly mirror their performance in the single-crew setting, I reinvestigate the set of greedy solutions to use in selecting the best feasible solution (BFS) with which to seed the simulated annealing algorithm. The selected greedy methods are evaluated prior to initializing the simulated annealing algorithm, and the solution with the

lowest objective value is used as the BFS to seed the local search. The individual candidate methods under consideration were SPT, IF, LZG, SQG, and LAFO/LASR. I consider LAFO/LASR together, selecting the better solution of the two, because the vast majority of the computational effort is spent in finding first-order effect estimates, after which both LAFO and LASR solutions can be evaluated with minimal additional effort. Based on initial testing, I discard SPT and LZG due to poor solution accuracy and choose the combinations IF/SQG and IF/LAFO/LASR for computational testing as seeds for the simulation annealing algorithm. In all computational testing, I ensure that all used information is "paid for" in terms of run time. For example, in order to seed the local search with IF/SQG, the run time for the local search starts at the sum of the time required to find the IF and SQG solutions, plus the time to evaluate both objective functions in order to determine which sequence to use as the seed. While IF/LAFO/LASR requires more time initially than IF/SQG, the increased seed quality results in overall shorter run times for the simulated annealing algorithm, including the time to find the seed solution, while maintaining or increasing final solution quality. Based on these results, I seed all additional multi-crew simulated annealing experiments using IF/LAFO/LASR.

3.5.3 Novel Simulated Annealing Neighborhoods

In order to explore additional neighborhoods – and even combinations of neighborhoods – for use in simulated annealing, I first define a new structure to store the current repair schedule. Instead of storing a single sequence which is then post-processed to K crews, I store K repair sequences, one for each crew. This structure allows for deliberate swaps of links within a crew without necessarily shifting any links between crews, which I refer to as a within-crew update. Alternatively, I can choose to swap two links in, say, the same ordinal position on two different crews, e.g. swap the second link in crew a's sequence with the second link in crew b's sequence. I refer to a proposed movement using this method as a crew-swap update. These two methods can be mixed in a single simulated annealing strategy to balance

diversification (changing which links are assigned to which crews) and intensification (optimizing the order of links within a crew, given crew assignments) within the local search.

I compare three strategies which combine these two update types and benchmark them against the performance of the 1-neighborhood adapted from the sequencing problem described in §3.5.2.3. The first strategy tested alternates within-crew and crew-swap updates, regardless of whether each proposed update results in updating the current sequence. On odd numbered iterations, a within-crew update is proposed and on even iterations a crew-swap update is proposed. The second strategy only proposes a crew-swap update every $\lceil N/K \rceil$ iterations, where N is the number of broken links and K is the number of crews, and all other proposed updates are within-crew updates. Finally, the third strategy only proposes a crew-swap update whenever a within-crew update failed (was not accepted) on the previous iteration, otherwise relying on within-crew updates. In order to directly compare the effects of the neighborhood combinations, I do not modify any other parameter choices from the single-crew algorithm.

To assess solution time and quality, I conduct five simulated annealing runs for each of the four strategies for each generated map of broken links. In this manner, I can assess not only performance, but consistency of performance, due to the randomness inherent in any given simulated annealing run. From Figure 3.2, I observe that none of the proposed strategies outperform the 1-neighborhood using post-processing in terms of average solution quality or consistency. Since all four methods evidence roughly comparable run times, I maintain the 1-neighborhood strategy described in §3.5.2.3 for subsequent experiments. A possible intuition for these results is that since total travel delay is determined by link completion order and times, irrespective of crew breakdown, the global repair start sequence is a better proxy than the crew sequences.

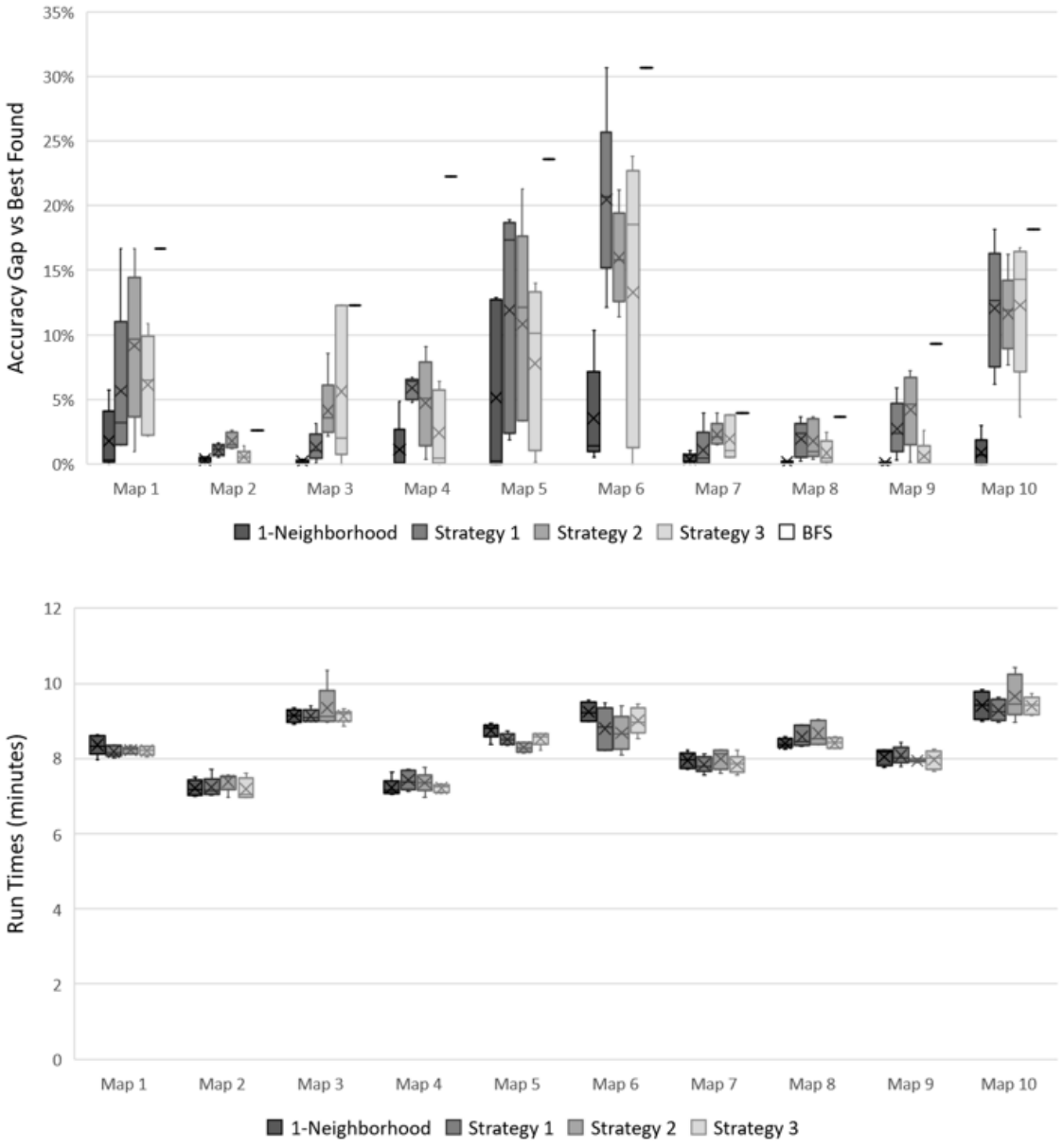

Figure 3.2: Comparison of SA neighborhood strategies; 20 broken links with three crews on Anaheim network; the shaded area is the interquartile range (IQR) containing the median (horizontal line), the X indicates the mean; BFS lines indicate BFS used to initialize SA

3.5.4 Decomposition Methods

In addition to adapting sequencing methods for the multi-crew problem, I explore first decomposing the set of broken links into sets assigned to each crew, and then ordering the links for repair within each crew. The motivation behind this approach is to increase the computational tractability of large scale problems with several available work crews. I propose two methods of decomposing the set of broken links into crew assignments, and four within-crew sequencing methods.

3.5.4.1 Minimum Makespan Decomposition

The simplest decomposition method I propose is to assign broken links to crews by approximating a minimum makespan (MM) assignment. I approximate this objective for decomposition rather than the desired weighted completion objective because minimizing makespan requires knowledge of only repair durations, rather than state-dependent weights. Exactly solving minimum makespan on identical parallel machines is an NP-hard problem because 2-PARTITION, which is NP-hard, reduces to minimum makespan on two identical parallel machines (Bruno et al., 1974; Karp, 1972). Therefore, I use a largest processing time (LPT) first greedy assignment, shown by Graham (1969) to obtain a 4/3 approximation of the optimal minimum makespan objective. Specifically, where $C_{max}(LPT)$ is the makespan using LPT assignment and $C_{max}(OPT)$ is the optimal minimum makespan assignment, Graham shows that

$$\frac{C_{max}(LPT)}{C_{max}(OPT)} \leq \frac{4}{3} - \frac{1}{3K}$$

where K is the number of available crews (machines).

3.5.4.2 Interaction Coefficient Decomposition

For a more sophisticated decomposition, I define pairwise interaction coefficients (IC) between broken links. Each interaction coefficient $IC[a, b]$ captures the fraction by which flow on link b increases (positive coefficient) or decreases (negative

coefficient) due to breaking only link a (with all other broken links already repaired). If the flow on link b increases due to breaking link a, then the links function in parallel to some extent given the network demand and structure. Similarly, if the flow on link b decreases due to breaking link a, then the links function in sequence to some extent given the network demand and structure. I hypothesize that two arcs in sequence which are required to repair the same paths should be assigned to different crews (to be potentially worked simultaneously), whereas two arcs which run in parallel, such that a shortest path chooses between the two, should be assigned to the same crew (such that only one should be prioritized for repair) (Figure 3.3).

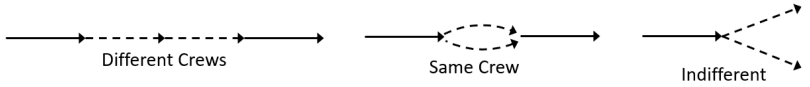

Figure 3.3: Broken arcs (dashed) in sequence, parallel, and unrelated

Algorithm 2 presents the crew assignment method once interaction coefficients are calculated for each pair of broken links, while Algorithm 3 gives the method of calculating interaction coefficients. Interaction coefficients need not be symmetric since the fractional impact of breaking link a on the flow on link b need not be equal to the fractional impact of breaking link b on the flow on link a, especially if the initial flow values are significantly different. Given the $N \times N$ interaction coefficient matrix, for each available crew I choose the highest interaction coefficient $IC[i, j]$ corresponding to unassigned broken links and assign the two corresponding links i and j to that crew. In order to avoid initial assignments which are likely to be suboptimal, I use a safety duration of $\frac{1}{K} \sum_{b \in \mathcal{B}} D_b$. If the selected pair of links would exceed the safety duration when assigned to the same crew, only the link with the longer duration is assigned during the first phase. Once each crew is assigned a link or pair of links, the second assignment phase distributes the remaining links to crews one by one, always assigning to the crew with the shortest currently assigned work duration. The link chosen for assignment at each step is the link for which the sum

of interaction coefficients between that link and links already in the chosen crew is maximized.

Algorithm 2 Interaction Coefficient Crew Assignments

Input: $\mathcal{B}$ (set of broken links), *duration* (dictionary with keys: broken links and values: repair durations), K (integer, number of crews)
crew-time $\leftarrow [0] \cdot K$, *which-crew* $\leftarrow$ dict(), *safety* $=$ sum(*duration* for $b \in \mathcal{B})/K$
$\Delta \leftarrow$ ***Initialize***
Set Δ' as a deep copy of Δ
toassign $=$ set$(b \in \mathcal{B})$
Assignment
for k in range(K) **do** ▷ *Assign first link(s) to each crew*
    $b, b' \leftarrow$ links corresponding to indices of argmax(Δ')
    if $duration[b] + duration[b'] > safety$ **then**
        if $duration[b] > duration[b']$ **then** $add \leftarrow [b]$
        else $add \leftarrow [b']$
    else $add \leftarrow [b, b']$
    for a in add **do**
        which-crew$[a] \leftarrow k$
        crew-time$[k] \leftarrow$ *crew-time*$[k] + duration[a]$
        Remove a from *toassign*
        Set row and column corresponding to a in Δ' to -1
while *toassign* $\neq \varnothing$ **do** ▷ *Assign remaining links to crews*
    $k \leftarrow$ index of min(*crew-time*)
    temp $\leftarrow \varnothing$
    for $b \in$ *toassign* **do**
        $t_1 \leftarrow$ sum$(\Delta[b, j]$ where j is assigned to crew $k)$
        $t_2 \leftarrow$ sum$(\Delta[i, b]$ where i is assigned to crew $k)$
        Append $\max(t_1, t_2)$ to *temp*
    $a \leftarrow$ link in *toassign* corresponding to max value of *temp*
    which-crew$[a] \leftarrow k$
    crew-time$[k] \leftarrow$ *crew-time*$[k] + duration[a]$
    Remove a from *toassign*
return *which-crew*

3.5.4.3 Optimal Within Crew Sequencing

Once broken links are assigned to crews, I then sequence the links within their crew. Of note, within crew sequences are not fully independent, since the overall

Algorithm 3 Interaction Coefficient Crew Assignments, ***Initialize***

Input: $\mathcal{B}$ (set of broken links), *duration* (dictionary with keys: broken links and values: repair durations), K (integer, number of crews)
Initialize Δ as $N \times N$ array of zeros
Solve $TAP(\mathcal{A} \cup \mathcal{R})$ (pre-disruption network state)
Initialize dictionary *initflow* with keys: $b \in \mathcal{B}$ and values: flows from $TAP(\mathcal{A} \cup \mathcal{R})$
for $b \in \mathcal{B}$ **do**
    Solve $TAP(\mathcal{A} \cup \mathcal{R} \setminus b)$ and store flows for $b \in \mathcal{B}$ in dictionary *tempflow*
    for $b' \in \mathcal{B}$ **do**
        if $initflow[b'] = 0$ **then**
            if $tempflow[b'] > 1$ **then** $\Delta[b, b'] \leftarrow 1$
            else $\Delta[b, b'] \leftarrow 0$
        else
            $\Delta[b, b'] \leftarrow (tempflow[b'] - initflow[b'])/initflow[b']$
return Δ

network repair state determines the incremental benefit of repairing any link, and that network repair state may change between the start and completion of a single link repair, due to operating multiple crews. However, in order to develop a heuristic with a reasonable time-complexity, I find "locally" optimal repair sequences for each crew, rather than "globally" optimal sequences. Specifically, I find the optimal repair sequence for each crew's assigned broken links, as if all other links (assigned to other crews) were already repaired. To demonstrate the impact of this simplification, optimally sequencing by brute force enumeration without decomposition requires evaluating approximately $N!/K!$ sequences, since the first K links in the global sequence are effectively interchangable as discussed in §3.5.2.3. Finding the globally optimal sequences for each crew given crew assignments using Algorithm 4 requires evaluating approximately $\left(\frac{N}{K}!\right)^K$ sequences. In contrast, finding the locally optimal sequences for all K crews independently using Algorithm 5 requires evaluating $\lceil \frac{N}{K} \rceil ! \times K$ sequences.

3.5.4.4 Importance Factor Within Crew Sequencing

In contrast to attempting to find the optimal within crew sequencing, either globally or locally, I alternatively use an importance factor to sequence links within

Algorithm 4 Globally Optimal (Brute Force) Within Crew Sequencing

Input: $\mathcal{B}$, *duration*, K, *which-crew* (dict w/ key: broken link, val: crew assignment)
Initialize list *damaged* of length K
for $b \in$ *which-crew* **do**
    $k \leftarrow$ *which-crew*$[b]$
    Append b to *damaged*$[k]$
Initialize list *subsequences* of length K
for k in range(K) **do**
    subsequences$[k] \leftarrow$ iterator of all permutations of *damaged*$[k]$
allsequences $\leftarrow$ cartesian product of *subsequences*$[k]$ for k in range(K)
mincost $\leftarrow \infty$, *minseq* $\leftarrow$ None
for *seq* in *allsequences* **do**
    cost $\leftarrow$ evaluate *seq* objective function value
    if *cost* $<$ *mincost* **then**
        mincost $\leftarrow$ *cost*
        minseq $\leftarrow$ *seq*
return *minseq*

Algorithm 5 Locally Optimal (Brute Force) Within Crew Sequencing

Input: $\mathcal{B}$, *duration*, K, *which-crew* (dict w/ key: broken link, val: crew assignment)
Initialize list *damaged* of length K
for $b \in$ *which-crew* **do**
    $k \leftarrow$ *which-crew*$[b]$
    Append b to *damaged*$[k]$
Initialize lists *subsequences*, *mincost*, and *minseq* of length K
for k in range(K) **do**
    subsequences$[k] \leftarrow$ iterator of all permutations of *damaged*$[k]$
    mincost$[k] \leftarrow \infty$, *minseq*$[k] \leftarrow$ None
    for *seq* in *subsequences*$[k]$ **do**
        cost $\leftarrow$ evaluate *seq* objective value (with all links not in *damaged*$[k]$ functional)
        if *cost* $<$ *mincost*$[k]$ **then**
            mincost$[k] \leftarrow$ *cost*
            minseq$[k] \leftarrow$ *seq*
return *minseq*

crews. This simple strategy orders links for repair within their already assigned crews in decreasing order of pre-disruption flows. Algorithm 6 overviews this method of post-crew assignment sequencing.

Algorithm 6 Importance Factor Within Crew Sequencing

Input: $\mathcal{B}$, *duration*, K, *which-crew* (dict w/ key: broken link, val: crew assignment)
Solve $TAP(\mathcal{A} \cup \mathcal{R})$ (pre-disruption network state)
Initialize dictionary *initflow* with keys: $b \in \mathcal{B}$ and values: flows from $TAP(\mathcal{A} \cup \mathcal{R})$
Sort $\mathcal{B}$ in order of decreasing pre-disruption flow (using *initflow*), store as *order*
for $b \in order$ **do**
    $k \leftarrow$ *which-crew*$[b]$; append b to $seq[k]$
return *seq*

3.5.4.5 Sequential Greedy Within Crew Sequencing

The final method I use to order links within crews is a sequential greedy method, referred to as MM (greedy) or CC (greedy) in figures. Essentially, the order for all crews is solved for concurrently. At each repair stage, from the crew with the shortest set of repairs so far, I next repair the link from that crew with the greatest immediate benefit, as depicted in Algorithm 7. Sequential greedy nor local optimal dominates the other in time complexity for every combination of N and K. Local optimal sequencing is highly sensitive to the ratio N/K (average number of links assigned to each crew) as well as the maximum number of links assigned to a single crew due to evaluating approximately $\lceil \frac{N}{K} \rceil ! \times K$ sequences. In contrast, sequential greedy evaluates approximately $(N/K)^2 \times K = N^2/K$ sequences.

3.6 Results

Numerical experiments are conducted on a desktop computer running Ubuntu with a quad-core 3.3 GHz processor and 8 GB RAM. The test networks used are again Anaheim and Berlin-Mitte-Center from the Transportation Networks research repository (2022). All instances of TAP are solved using Boyles' implementation of

Algorithm 7 Sequential Greedy Within Crew Sequencing

Input: $\mathcal{B}$, *duration*, K, *which-crew* (dict w/ key: broken link, val: crew assignment)
Initialize list *damaged* of length K; *eligible* $\leftarrow \emptyset$
for $b \in$ *which-crew* **do**
    $k \leftarrow$ *which-crew*$[b]$; append b to *damaged*$[k]$; append b to *eligible*
seq $\leftarrow [0] \cdot K$, *crew-order* $\leftarrow \emptyset$, *crew-time* $\leftarrow [0] \cdot K$, *tstts* $\leftarrow$ dict(), *bb* $\leftarrow$ dict()
$after \leftarrow$ TSTT from solving $TAP(\mathcal{A} \cup \mathcal{R} \setminus \mathcal{B})$
for $b \in$ *eligible* **do** ▷ *Find best immediate benefit (bb) of repairing each link first*
    temp $\leftarrow$ TSTT from solving $TAP(\mathcal{A} \cup \mathcal{R} \setminus (\mathcal{B} \setminus b))$
    $bb[b] \leftarrow after - temp$, $tstts[b] \leftarrow temp$
Initialize list *order* of length K
for k in range(K) **do**
    order$[k] \leftarrow$ *damaged*$[k]$ sorted by decreasing $bb[b]/duration[b]$
temp $\leftarrow$ [first element of *order*$[k]$ for k in range(K)]
for b in sorted(*temp*) **do** ▷ *Assign first link to each crew*
    $k \leftarrow$ *which-crew*$[b]$; *crew-time*$[k] \leftarrow$ *crew-time*$[k] +$ *duration*$[b]$
    Append b to *crew-order* and to *seq*$[k]$; remove b from *eligible* and from *damaged*
$after \leftarrow tstts$[first link in *crew-order*]
while *eligible* $\neq \emptyset$ **do** ▷ *Assign remaining links to crews*
    $k \leftarrow$ index of min(*crew-time*), *tstts* $\leftarrow$ dict(), *bb* $\leftarrow$ dict()
    if *damaged*$[k] = \emptyset$ **then** ▷ *If crew k links are assigned, move longest unassigned link to k*
        mover $\leftarrow$ argmax(*duration*$[b]$ for $b \in$ *eligible*)
        Remove *mover* from *damaged*[*which-crew*[*mover*]]
        which-crew[*mover*] $= k$, append *mover* to *damaged*$[k]$
    for $b \in$ *damaged*$[k]$ **do** ▷ *Find current bb of repairing each link in crew k not yet repaired*
        temp $\leftarrow$ TSTT from solving $TAP(\mathcal{A} \cup \mathcal{R} \setminus [\mathcal{B} \setminus$ (*crew-order*$[:1-K] \cup b)])$
        $bb[b] \leftarrow after - temp$, $tstts[b] \leftarrow temp$
    order $\leftarrow$ *damaged*$[k]$ sorted by decreasing $bb[b]/duration[b]$
    $b \leftarrow$ first element in *order*; $k \leftarrow$ *which-crew*$[b]$; *crew-time*$[k] \leftarrow$ *crew-time*$[k] +$ *duration*$[b]$
    if *crew-time*$[k] =$ max(*crew-time*) **then**
        Append b to *crew-order* and to *seq*$[k]$; $after \leftarrow tstts[b]$
    else
        $i \leftarrow K + 1 -$ index of sorted(*crew-time*)$[k]$
        Insert b into *crew-order* at the ith to last position; append b to *seq*$[k]$
        $after \leftarrow$ TSTT from solving $TAP(\mathcal{A} \cup \mathcal{R} \setminus [\mathcal{B} \setminus$ *crew-order*$[:1-K]])$
    Remove b from *eligible* and from *damaged*
return *seq*

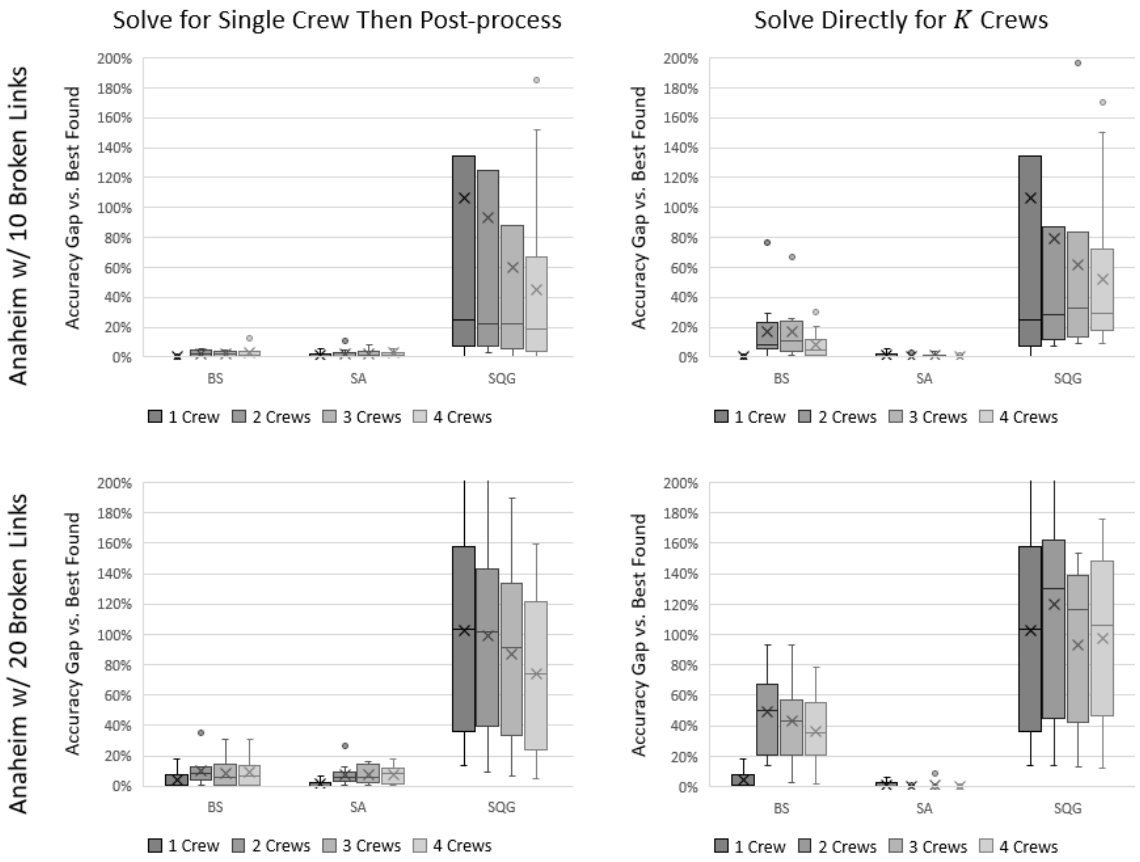

Figure 3.4: Comparison of accuracy gap versus best found OBJ function for post-processing vs. directly solving for K-crews for ten instances on the Anaheim network with 10 and 20 broken links; shaded area is the interquartile range IQR containing the median (horizontal line), the X indicates the mean, and outliers are shown as individual dots

Dial's Algorithm B (Boyles et al., 2023; Dial, 2006). Randomized problem instances are sampled in the same manner as in Chapter 2. I once again set the disconnection parameter $Q = 10$, representing adding artificial links for any disconnected OD pair post-disruption, as well as for any OD pair where post-disruption travel times increase at least tenfold from the base case (pre-disruption).

For the strictly greedy methods – LASR, LAFO, LZG, IF, and SPT – post-processing versus solving directly for K-crews produces the same results because the greedy ordering is a simple sequence, and none of the ordering criteria depend on

how many crews are used. For the beam search, simulated annealing, and sequential greedy heuristics, however, solving for a single-crew sequence and subsequently post-processing often gives a different multi-crew schedule than solving the multi-crew problem directly. Therefore, for each of these three methods, I compare the objective value accuracy gap (from the best found objective value) on the Anaheim network with 10 and 20 broken links, for ten instances each, in order to compare results obtained by post-processing versus solving the K-crews problem directly. For each of the 20 instances (ten with 10 broken links and ten with 20 broken links), I solve the problem for one, two, three, and four crews, using the same randomly instantiated broken links and link repair durations, additionally post-processing the single-crew solution for two, three, and four crews. The results are presented as box and whisker plots in Figure 3.4 and corresponding tabulated summary statistics are found in Appendix B, Tables B.1, B.2, and B.3.

For the beam search heuristic, solving directly for K-crews by using the multi-crew objective function whenever it is evaluated within the heuristic leads to higher accuracy gaps than post-processing. This result follows logically since the theory supporting the use of a beam search relies upon an analogue to Bellman's optimality principle only present in the single-crew formulation. However, in post-processing for multi-crew, the high accuracy evidenced by the beam search for the sequencing problem is diminished. Simulated annealing demonstrates a very similar solution quality degradation in post-processing to beam search, however, SA exhibits the opposite behavior to beam search when solving directly for K-crews. When solving directly for K-crews, using the multi-crew simulated annealing heuristic described in §3.5.2.3, the accuracy obtained is the highest of any method tested, and comparable to simulated annealing's single-crew accuracy. The direct multi-crew formulation of the sequential greedy method performs similarly to post-processing the single-crew sequential greedy sequence. Combined with the fact that both methods of implementing SQG are outperformed by LASR and LAFO (which have similar average run times and run time variation to SQG), SQG does not recommend itself to use

when multiple crews are available.

In order to test the decomposition methods, I compare them to SA, LAFO, LASR, and IF. I choose LAFO, LASR and IF from among the greedy methods, because these three methods outperform SQG, LZG, and SPT for multi-crew in terms of solution accuracy. I test the selected methods with 12 and 20 broken links on the Anaheim network, with $K = 2, 3, 4$ for $N = 12$ and $K = 3, 4, 5, 6$ for $N = 20$. The accuracy gap and run time comparisons are presented as box and whisker plots in Figures 3.5 and 3.6 for instances with 12 and 20 broken links, respectively. I include finding the globally optimal within crew sequences only for 12 broken links due to computational time.

Examining run times in Figure 3.5, finding the globally optimal within crew sequences is more time intensive than SA, though as K increases, the run times for the two methods grow more similar. This observation is significant for two reasons. However, as N increases, run times to find the globally optimal within crew sequence drastically exceed the run times for simulated annealing, regardless of the number of crews. These observations are significant for two reasons. Firstly, with the present methods of decomposition, spending the time to find globally optimal within crew sequences is not advised, since SA has significantly better solution accuracy, and faster or comparable run times. Secondly, the run time to find the globally optimal within crew sequence is highly dependent on the relationship between K and N. In Figure 3.6, there is a similar decrease in run times for locally optimal within crew sequencing as K increases, which is more pronounced when using IC decomposition. Because MM uses the longest processing time first assignment method, it is reasonable that MM will more often have near-equal numbers of links in each crew, whereas IC may result in more uneven link count distribution among crews.

From Figures 3.5 and 3.6, it is clear that SA exhibits the highest solution quality among methods tested, though it also has the longest run times, with the exception of globally optimal within crew sequencing. For this reason, when time allows,

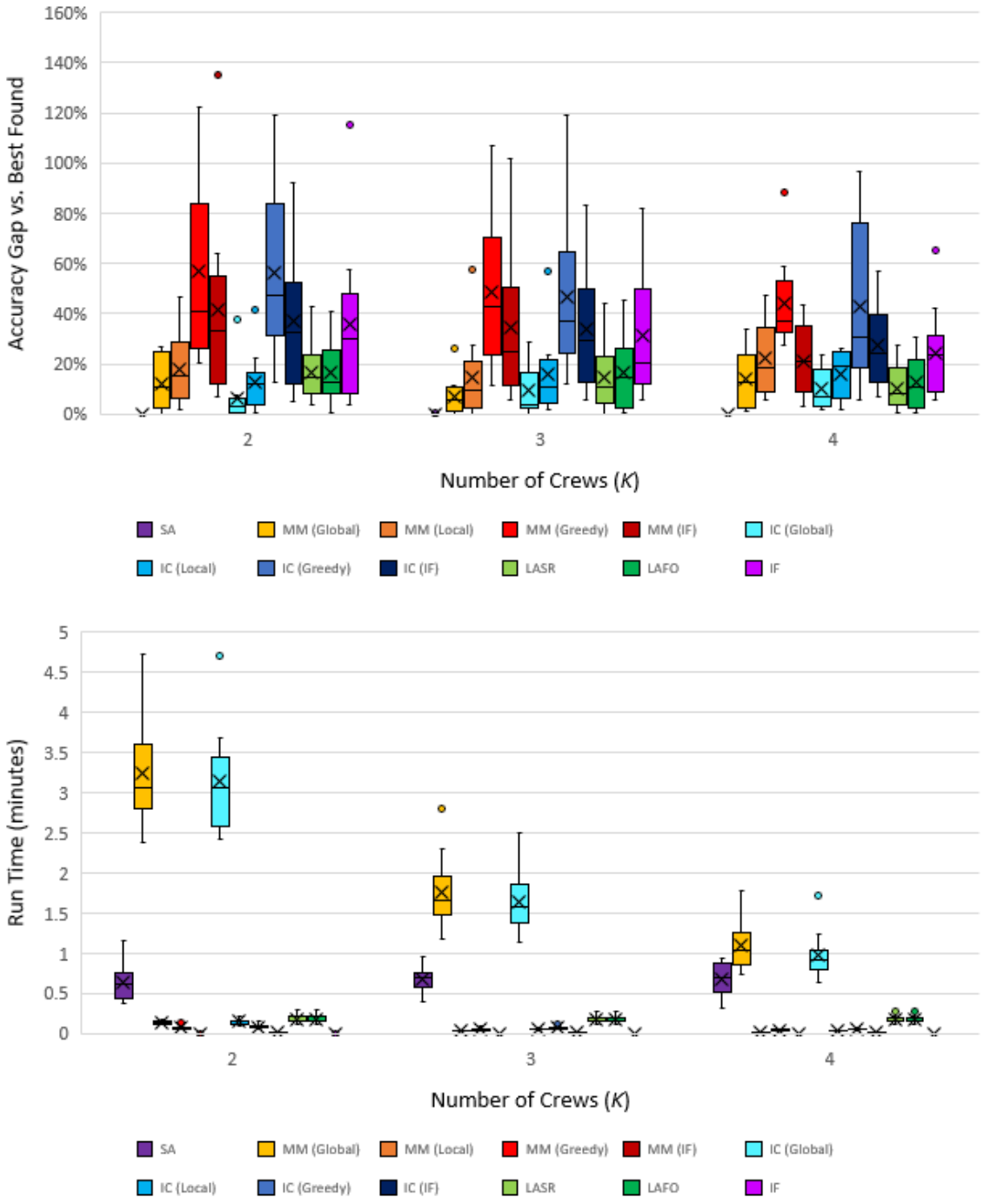

Figure 3.5: Run time and accuracy gap comparisons for multi-crew methods on Anaheim network with 12 broken links, 10 instances at each K value

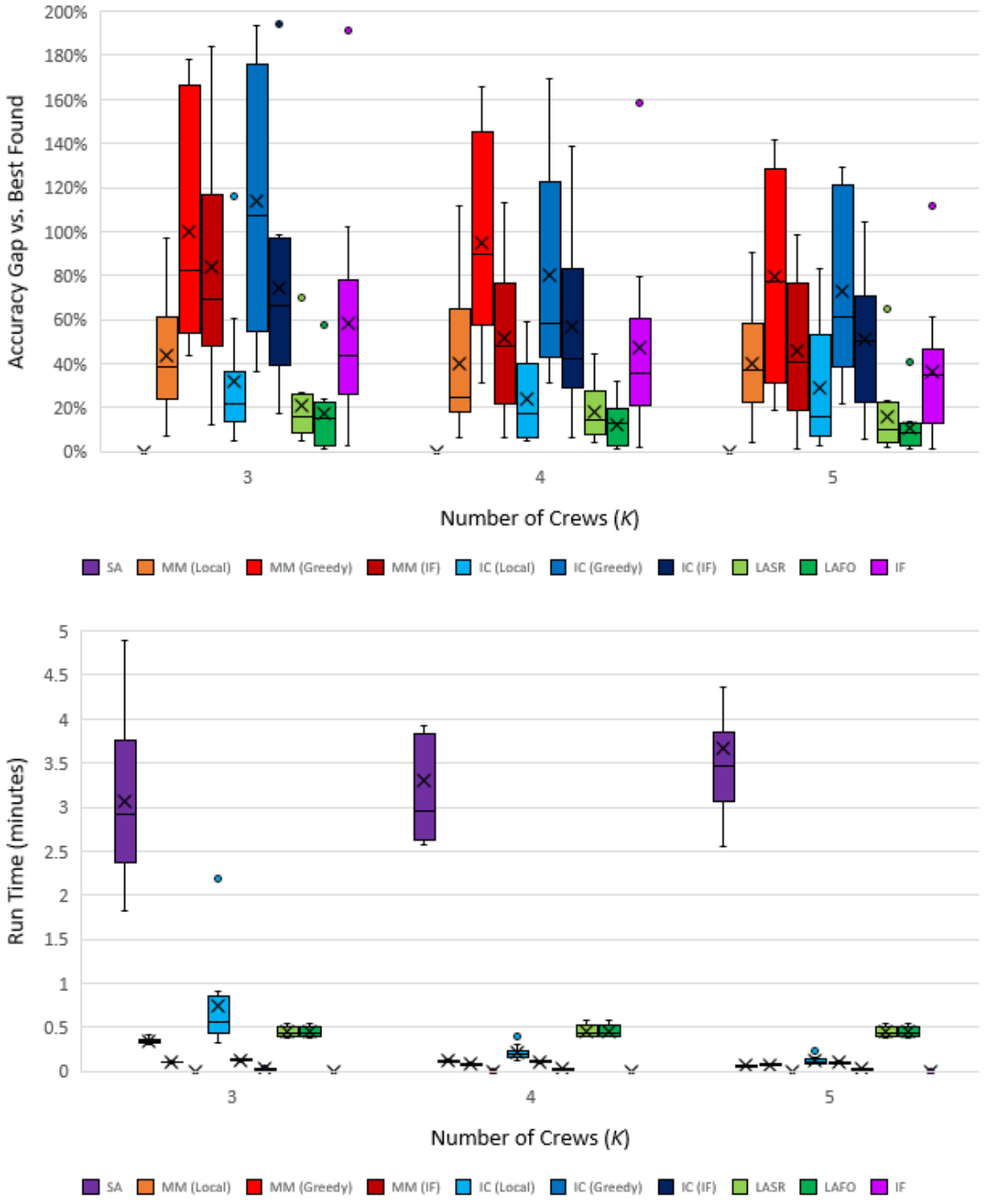

Figure 3.6: Run time and accuracy gap comparisons for multi-crew methods on Anaheim network with 20 broken links, 10 instances at each K value

I would recommend using SA for multi-crew scheduling. If time does not allow for the use of SA, then I would recommend the use of LAFO/LASR, taking advantage of the solution between the two with the lower objective function, since the incremental cost in computational time is minimal to obtain both solutions. While locally optimal within crew sequencing using IC decomposition rivalled the solution quality of LAFO and LASR with only 12 broken links, particularly at lower numbers of crews, this performance is not maintained when the number of broken links increased, within the scope of my testing. Neither greedy nor importance factor within crew sequencing recommend themselves to use, due to poor solution quality. At the observed solution qualities, if the most rapid scheduling solution is desired and LAFO and LASR are computationally intractable within operational constraints, I would point to the simple method of importance factor scheduling using post-processing. This method requires the solution of a single TAP, while outperforming the two other options tested for rapid solutions, LZG and SPT.

Finally, I generate 10 instances each for 10, 20, 30, 40, 50, 60, 70, and 80 broken links on the Anaheim network to investigate performance as the number of broken links increases. I set the number of crews available to $K = \lfloor \sqrt{N} \rfloor$ at each instance size. First, I examine the relationship between solution accuracy gap (from best found objective function value) and makespan increase from the best found (minimal) makespan using the data generated by solving each of the 800 instances with seven heuristics – SA, LASR, LAFO, SGQ, LZG, IF, and SPT. As shown in Figure 3.7, the two gap measures do not exhibit a clear correlation. The points along the vertical axis represent solutions where the solution quality varies widely while the makespan is at the best known value for the instance solved. These points are somewhat unsurprising since a makespan is determined completely by which links are assigned to which crews, whereas the objective function value depends heavily on the completion times of each link – determined by not only crew assignments, but also repair order. The density of points along or immediately adjacent to the horizontal axis, however, is more enlightening. This density of points, specifically between 0%

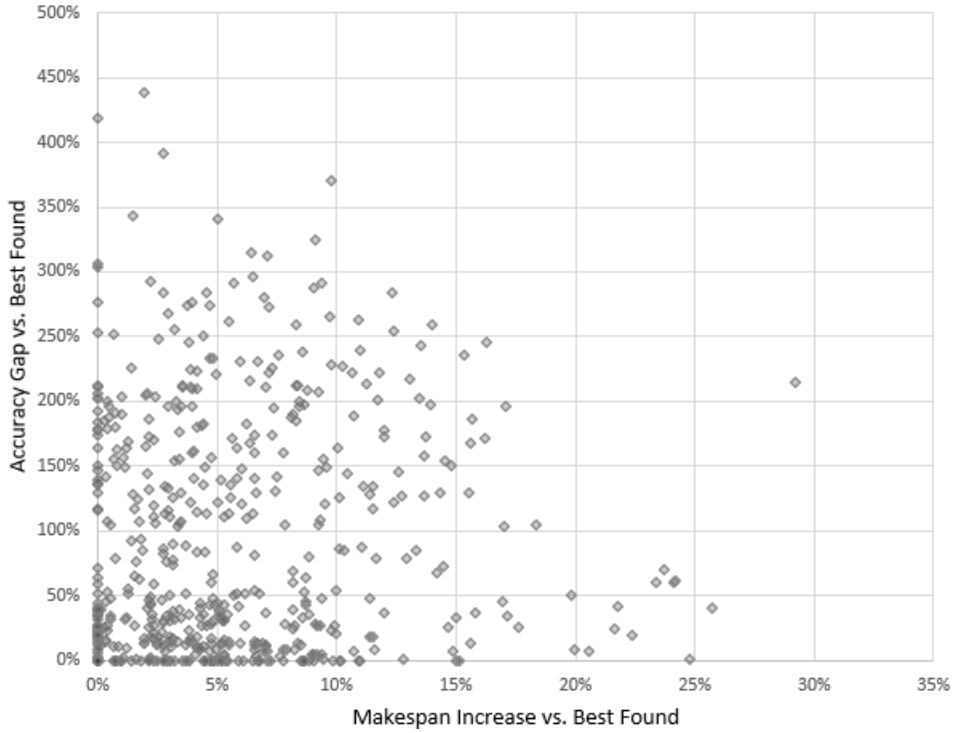

Figure 3.7: Comparison of accuracy gap versus best found OBJ function and makespan increase versus best found makespan for 800 instances (randomly generated maps) and seven heuristics for Anaheim network with 10-80 broken links

and 10% on the horizontal axis, shows that many solutions achieve or nearly achieve the best found objective value while having a makespan that is up to 10% higher than the minimal makespan found. In other words, a near optimal makespan does not seem to be either a reliable predictor of a low objective value or reliably predicted by a low objective value.

Using the same data from above as well as five random instances at 90 and 100 broken links, and additionally conducting the same experiments for 10–100 broken links on the BMC network, I generate Figure 3.8. Corresponding tabulated summary statistics are found in Appendix B, Tables B.4 – B.9. The top graph in Figure 3.8 depicts the distribution of accuracy gaps on Anaheim and BMC networks for SA and LAFO/LASR at each instance size. LAFO/LASR refers to approximating first order effects, evaluating both the LAFO and LASR objective values, and adopting the solution with the lower objective value. The bottom graph depicts the run time distributions for the same experiments. Figure 3.9 displays run time curve fits for SA on Anaheim and BMC networks, using the same data as figure 3.8, but with outliers removed. As in §2.8, I use a third degree polynomial fit with intercept fixed at zero. In 24 hours, SA can solve instances with up to about 120–130 broken links, and handles instances with about 170–180 broken links in 72 hours.

3.7 Conclusions

In this chapter, I define the multi-crew scheduling problem with the objective of minimizing total travel delay over the repair horizon and establish that the problem is NP-hard by restriction by reducing minimal weighted completion time on parallel machines to this problem. I adapt my simulated annealing heuristic for multi-crew scheduling problems. In addition, I adapt a sequential greedy method for use in the multi-crew setting, and post-process five other greedy methods for comparison using multiple identical crews. Finally, I present two methods of decomposing links into crews and subsequently sequencing within those crews, using one of four methods.

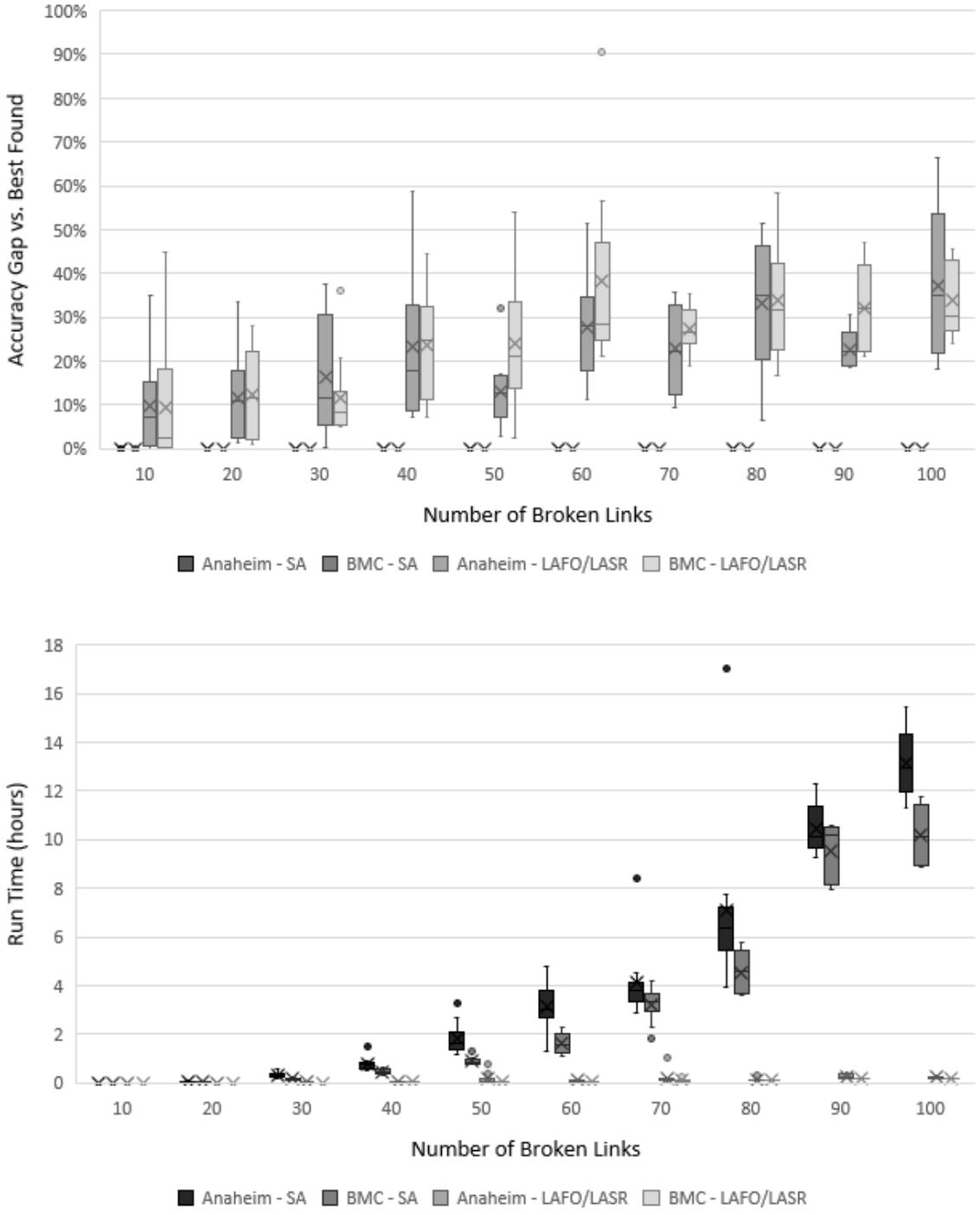

Figure 3.8: Run time and accuracy gap comparisons for simulated annealing and LAFO/LASR on Anaheim network with 10–100 broken links, ten instances at each value 10–80 and five instances at 90 and 100

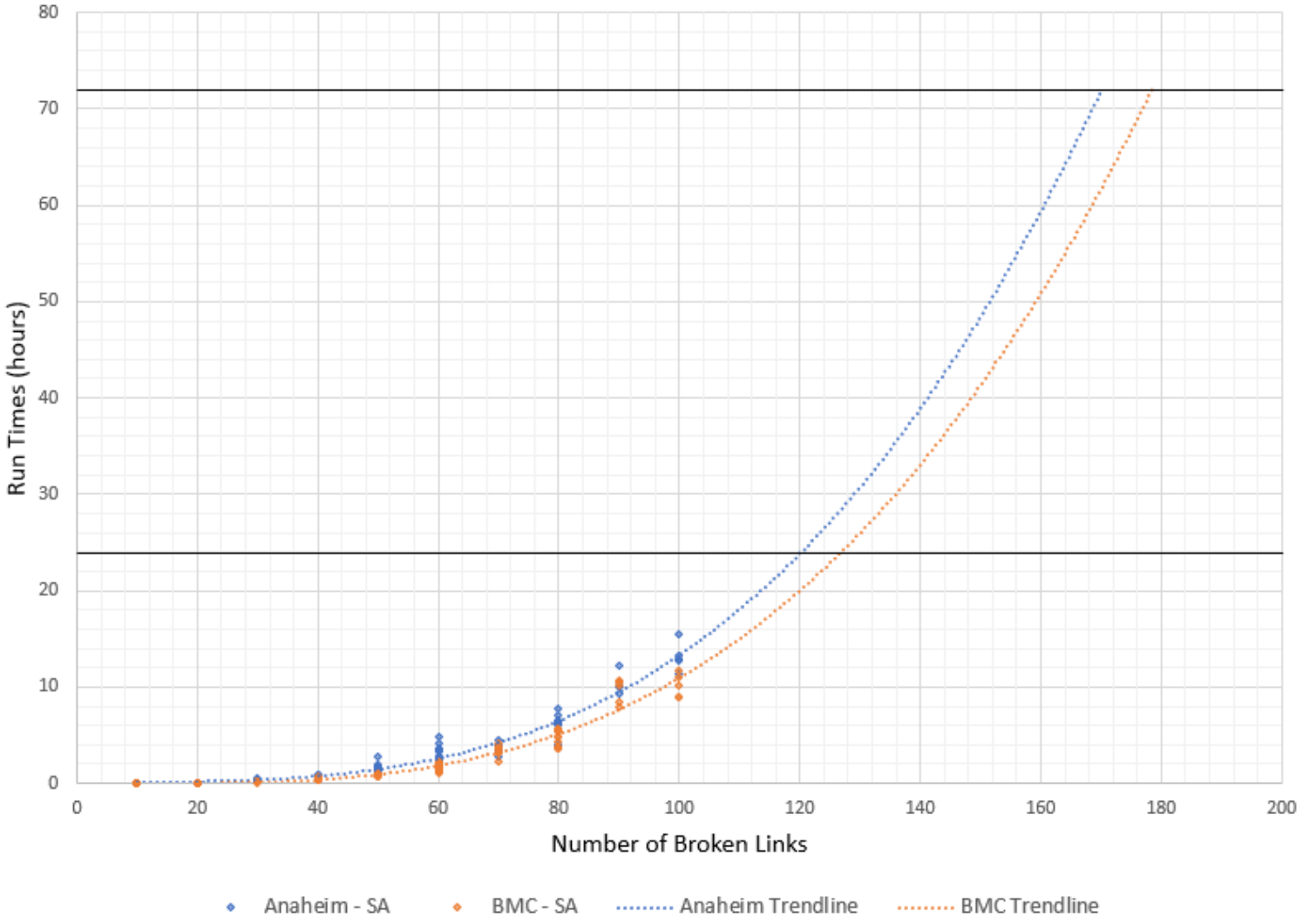

Figure 3.9: Run time curve fits for simulated annealing on Anaheim and BMC networks

Ultimately, however, these decomposition methods do not evidence sufficient solution quality relative to their run times.

I compare the benefits of directly solving the K-crews problem versus post-processing for those methods which are not strictly greedy methods. These experiments demonstrate the importance of using the multi-crew SA heuristic when more than one crew is available, rather than simply post-processing the sequencing result. In addition, the results illustrate the importance of knowing how many crews will be available upfront, and points to an interesting question for further research. If the number of crews which will be available is unknown, variable over the repair horizon, or otherwise stochastic, how does the observed total travel delay over the repair horizon compare to the initially calculated value?

Finally, the multi-crew accuracy gap and run time comparisons indicate that the multi-crew SA heuristic produces the highest quality solutions, and recommends itself to use if operational constraints allow. The combination method LAFO/LASR demonstrates higher solution quality than any other greedy method tested, in addition to outperforming the computationally feasible decomposition methods in solution quality testing at a higher number of broken links. Among the other greedy methods, if time-constraints prohibit the use of LAFO/LASR, then IF with post-processing would be the recommended method to obtain a very rapid, though not as advantageous, solution. An area for future research would be to explore the performance of an SA algorithm where run time is used as the stopping criterion – i.e. stop after 24 hours, rather than a certain number of iterations.

Chapter 4: Conclusion

This dissertation proposed and benchmarked heuristic solution methods for the disaster recovery sequencing and scheduling problems. Chapter 2 focused on sequencing, demonstrating a simulated annealing method which delivers very high solution quality and handles significantly larger problem instances than any previous search method which accounts for travel times in the objective function. In Chapter 3, I formally established the multi-crew scheduling problem as NP-hard. I adapted the simulated annealing method for the multi-crew problem formulation and demonstrated comparably high solution quality to the single-crew formulation of the same heuristic. Additionally, I developed two methods of decomposition into crews, prior to sequencing within those crews, one of which is a novel method of quantifying the extent to which a link is in parallel or sequence with another link.

4.1 Summary and Implications

Chapter 2 addressed the single-crew sequencing problem. I presented two formulations of the problem, the first a bilevel optimization problem with static TAP as the lower level problem, and the second a shortest path formulation, treating the TSTTs for each encountered state as known (or calculated as needed). I used the second formulation to explore complexity, demonstrating the need for heuristic, rather than exact solution methods. I proposed a simulated annealing heuristic which achieves high quality solutions for problem instances with up to 165–175 broken links on the Anaheim and Berlin-Mitte-Center networks in 24 hours, and instances with up to 250–260 broken links on the same networks in 72 hours. Because extreme hazards and natural disasters often affect hundreds or thousands of links, these statistics are critical when considering this heuristic's application to real world scenarios. While other methods and heuristics exist to solve the sequencing problem, simulated an-

nealing offers the most compelling balance between solution time and quality given the ongoing negative effects of a damaged network.

In Chapter 3 I extended the problem to a multi-crew formulation which I demonstrated to be NP-hard, further establishing the need for heuristic solutions. I proposed a multi-crew adaptation of the simulated annealing heuristic developed in Chapter 2 and demonstrated the merits of solving the multi-crew problem directly as opposed to post-processing a solution to the sequencing problem. Additionally, I developed two methods of decomposing links into sets for assignment to crews *a priori*, and four methods of subsequently ordering links within crews. One of these decomposition methods, using interaction coefficients, is a novel approach to quantifying the degree to which a link is in parallel or sequence with another link given demand between an arbitrary number of origin-destination pairs. Ultimately, these decomposition methods did not achieve sufficient solution quality relative to their run times to recommend their use over other methods, but the method of assigning interaction coefficients may have additional applications not yet explored. The multi-crew simulated annealing heuristic achieved the highest solution quality of the methods tested, with only LAFO/LASR consistently coming within even a 50% accuracy gap of the solution value found by simulated annealing. In 24 hours, simulated annealing handles instances with up to 120–130 broken links on Anaheim and Berlin-Mitte-Center, extending to instances with 170–180 broken links in 72 hours.

4.2 Future Work

The simulated annealing algorithms presented for single- and multi-crew represent significant strides towards handling realistically-sized instances on small to medium networks. However, there is significant space for future work in developing additional algorithms which can further increase the size of problems solvable to high levels of accuracy. Notably, establishing a tight lower bound or developing another method which could rival simulated annealing's solution accuracy with reasonable

run times would allow for more precise characterization of the accuracy achieved by simulated annealing at higher numbers of broken links. Additionally, a parallelized implementation optimized for computational speed may significantly extend the size of instance which can be handled by the presented simulated annealing algorithm in reasonable planning time periods.

Additional directions for future research include relaxing key assumptions or introducing stochasticity, potentially in the number of repair crews available or in the project durations. Strategically introducing stochasticity to the deterministic models presented in Chapters 2 and 3 has the potential to more accurately reflect realistic scenarios and provide additional insights for sequencing and scheduling road repair projects after extensive network damage. Stochastic repair times recommend themselves to study especially after an extreme event when the full extent of underlying road damage may not be known until construction begins. Investigating stochastic repair times may also lead to incorporating preemption, potentially with setup times in order to avoid unrealistic job hopping behavior.

For a different line of inquiry, the structural assumption that there are a fixed number of available identical crews over the entire repair horizon could be loosened. This relaxation could be accomplished deterministically, potentially with some crews becoming available later than time zero, some crews only having availability until some later fixed time, or both. Comparing these relaxations to the base case could provide insight on the importance of having a large number of repair crews immediately available as well as the impact of tapering off the number of crews later in the repair horizon. Alternately, allowing for stochasticity in the number of available repair crews naturally leads to the question of whether a sequence, schedule, or scheduling policy is most effective given uncertainty about future numbers of available crews. Another related extension would be to allow multiple work crews to be assigned to a single link simultaneously, increasing the repair rate (though likely not linearly). This extension could provide insight on the relative benefits of concentrating or diffusing repair efforts with multiple available crews.

Appendix A: Single-crew Summary Statistics

A.1 Simulated Annealing Run Time Summary Statistics

The below tables correspond to run time graphs in Figures 2.9, 2.10, and 2.11.

Statistic	Number of Broken Links							
	8	9	10	11	12	13	14	15
Mean	0.098	0.145	0.185	0.228	0.332	0.384	0.466	0.564
Q1	0.080	0.108	0.144	0.179	0.242	0.287	0.362	0.411
Median	0.089	0.126	0.174	0.217	0.293	0.350	0.432	0.524
Q3	0.107	0.165	0.199	0.252	0.369	0.440	0.506	0.646
Low Outliers	0%	0%	0%	0%	0%	0%	0%	0%
High Outliers	6%	6%	9%	7%	7%	4%	7%	3%

Table A.1: SA run time (minutes) summary statistics for 8–15 broken links on Anaheim, 100 random instances at each number of broken links

Statistic	Number of Broken Links							
	8	9	10	11	12	13	14	15
Mean	0.088	0.104	0.142	0.179	0.239	0.301	0.375	0.462
Q1	0.067	0.091	0.124	0.149	0.200	0.250	0.299	0.372
Median	0.078	0.104	0.140	0.171	0.227	0.291	0.348	0.438
Q3	0.088	0.114	0.159	0.200	0.266	0.323	0.444	0.508
Low Outliers	0%	0%	1%	0%	0%	0%	0%	0%
High Outliers	2%	2%	1%	0%	5%	7%	1%	3%

Table A.2: SA run time (minutes) summary statistics for 8–15 broken links on Berlin-Mitte-Center, 100 random instances at each number of broken links

Statistic	Number of Broken Links			
	16	24	32	48
Mean	0.736	2.693	6.901	22.934
Q1	0.503	2.091	5.140	17.979
Median	0.600	2.591	5.989	22.089
Q3	0.923	3.122	8.858	27.590
Low Outliers	0%	0%	0%	0%
High Outliers	0%	0%	0%	0%

Table A.3: SA run time (minutes) summary statistics for 16, 24, 32, and 48 broken links on Anaheim, 25 random instances at each number of broken links

Statistic	Number of Broken Links			
	16	24	32	48
Mean	0.590	1.837	4.742	17.392
Q1	0.445	1.478	3.741	13.605
Median	0.516	1.718	4.556	16.147
Q3	0.616	2.006	5.625	21.325
Low Outliers	0%	0%	0%	0%
High Outliers	4%	8%	0%	0%

Table A.4: SA run time (minutes) summary statistics for 16, 24, 32, and 48 broken links on Berlin-Mitte-Center, 25 random instances at each number of broken links

A.2 Simulated Annealing Accuracy Gap Summary Statistics

The below tables correspond to accuracy gap graphs in Figures 2.9, 2.10, and 2.11.

Statistic	Number of Broken Links							
	8	9	10	11	12	13	14	15
Mean	1.4%	0.9%	1.5%	1.2%	1.7%	1.7%	1.8%	2.4%
Q1	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
Median	0.0%	0.0%	0.0%	0.0%	0.0%	0.1%	0.0%	0.0%
Q3	0.0%	0.0%	0.1%	0.3%	0.5%	1.1%	1.3%	0.5%
Low Outliers	0%	0%	0%	0%	0%	0%	0%	0%
High Outliers	18%	18%	17%	16%	18%	17%	15%	20%

Table A.5: SA accuracy gap summary statistics for 8–15 broken links on Anaheim, 100 random instances at each number of broken links

Statistic	Number of Broken Links							
	8	9	10	11	12	13	14	15
Mean	0.3%	0.7%	1.2%	1.5%	1.7%	1.3%	2.3%	2.7%
Q1	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
Median	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.1%	0.1%
Q3	0.0%	0.0%	0.1%	0.2%	0.8%	0.3%	0.7%	1.3%
Low Outliers	0%	1%	0%	0%	0%	0%	0%	0%
High Outliers	15%	11%	17%	21%	18%	19%	21%	17%

Table A.6: SA accuracy gap summary statistics for 8–15 broken links on Berlin-Mitte-Center, 100 random instances at each number of broken links

Statistic	Number of Broken Links			
	16	24	32	48
Mean	2.9%	1.6%	0.0%	0.3%
Q1	0.0%	0.0%	0.0%	0.0%
Median	0.0%	0.0%	0.0%	0.0%
Q3	1.9%	0.0%	0.0%	0.0%
Low Outliers	0%	0%	0%	0%
High Outliers	20%	20%	0%	12%

Table A.7: SA accuracy gap summary statistics for 16, 24, 32, and 48 broken links on Anaheim, 25 random instances at each number of broken links

Statistic	Number of Broken Links			
	16	24	32	48
Mean	0.9%	6.3%	0.0%	0.0%
Q1	0.0%	0.0%	0.0%	0.0%
Median	0.0%	0.0%	0.0%	0.0%
Q3	0.4%	3.7%	0.0%	0.0%
Low Outliers	0%	0%	0%	0%
High Outliers	12%	16%	0%	0%

Table A.8: SA accuracy gap summary statistics for 16, 24, 32, and 48 broken links on Berlin-Mitte-Center, 25 random instances at each number of broken links

A.3 Comparison Summary Statistics for 48 broken links

The below tables correspond to Figure 2.12.

Statistic	Method						
	SA	LASR	LAFO	SQG	LZG	IF	SPT
Mean	22.934	2.561	2.561	1.279	0.054	0.001	0.000
Q1	17.979	1.997	1.997	0.986	0.041	0.001	0.000
Median	22.089	2.278	2.278	1.087	0.048	0.001	0.000
Q3	27.590	2.929	2.929	1.505	0.061	0.001	0.000
Low Outliers	0%	0%	0%	0%	0%	0%	0%
High Outliers	0%	4%	4%	0%	8%	0%	8%

Table A.9: Methods comparison run time (minutes) summary statistics for 48 broken links on Anaheim, 25 random instances

Statistic	Method						
	SA	LASR	LAFO	SQG	LZG	IF	SPT
Mean	17.392	2.423	2.423	1.120	0.048	0.001	0.000
Q1	13.605	2.064	2.064	1.021	0.044	0.001	0.000
Median	16.147	2.238	2.238	1.128	0.046	0.001	0.000
Q3	21.325	2.501	2.501	1.182	0.050	0.001	0.000
Low Outliers	0%	0%	0%	0%	0%	0%	0%
High Outliers	0%	8%	8%	4%	4%	8%	0%

Table A.10: Methods comparison run time (minutes) summary statistics for 48 broken links on Berlin-Mitte-Center, 25 random instances

Statistic	Method						
	SA	LASR	LAFO	SQG	LZG	IF	SPT
Mean	0.3%	30.9%	37.3%	181.1%	228.4%	196.4%	318.0%
Q1	0.0%	14.7%	21.2%	97.0%	121.8%	97.3%	208.8%
Median	0.0%	26.3%	37.7%	183.8%	217.9%	153.5%	296.5%
Q3	0.0%	46.6%	49.0%	252.1%	320.1%	279.9%	409.1%
Low Outliers	0%	0%	0%	0%	0%	0%	0%
High Outliers	12%	4%	4%	0%	0%	0%	0%

Table A.11: Methods comparison accuracy gap summary statistics for 48 broken links on Anaheim, 25 random instances

Statistic	Method						
	SA	LASR	LAFO	SQG	LZG	IF	SPT
Mean	0.0%	37.2%	44.3%	100.5%	166.1%	180.2%	231.1%
Q1	0.0%	26.3%	29.1%	45.6%	92.9%	113.2%	176.7%
Median	0.0%	33.8%	45.2%	74.2%	151.1%	169.3%	219.5%
Q3	0.0%	45.5%	53.5%	146.9%	227.5%	239.3%	259.6%
Low Outliers	0%	0%	0%	0%	0%	0%	0%
High Outliers	0%	4%	4%	0%	0%	0%	4%

Table A.12: Methods comparison accuracy gap summary statistics for 48 broken links on Berlin-Mitte-Center, 25 random instances

A.4 Varied Demand Multiples Summary Statistics

The below tables correspond to Figures 2.13 and 2.14.

A.4.1 Anaheim – Run Time Comparisons

Statistic	Demand Multiplier				
	0.25	0.50	1.00	2.00	4.00
Mean	4.806	4.847	5.123	6.376	11.002
Q1	4.258	4.036	3.940	4.911	8.156
Median	4.974	4.865	4.909	5.908	9.813
Q3	5.619	5.936	6.340	8.056	12.694
Low Outliers	0%	0%	0%	0%	0%
High Outliers	0%	0%	0%	0%	8%

Table A.13: BS run time (minutes) summary statistics for varied demand multipliers on Anaheim, 25 random instances at each multiplier

Statistic	Demand Multiplier				
	0.25	0.50	1.00	2.00	4.00
Mean	0.443	0.488	0.737	1.139	3.323
Q1	0.346	0.384	0.500	0.852	2.329
Median	0.409	0.450	0.594	1.083	3.129
Q3	0.509	0.555	0.804	1.192	3.792
Low Outliers	0%	0%	0%	0%	0%
High Outliers	0%	0%	12%	12%	4%

Table A.14: SA run time (minutes) summary statistics for varied demand multipliers on Anaheim, 25 random instances at each multiplier

Statistic	Demand Multiplier				
	0.25	0.50	1.00	2.00	4.00
Mean	0.160	0.185	0.303	0.423	1.363
Q1	0.157	0.164	0.206	0.353	1.037
Median	0.158	0.169	0.225	0.424	1.313
Q3	0.158	0.176	0.370	0.488	1.667
Low Outliers	0%	0%	0%	0%	0%
High Outliers	8%	12%	8%	0%	4%

Table A.15: LASR run time (minutes) summary statistics for varied demand multipliers on Anaheim, 25 random instances at each multiplier

Statistic	Demand Multiplier				
	0.25	0.50	1.00	2.00	4.00
Mean	0.160	0.185	0.303	0.423	1.363
Q1	0.157	0.164	0.206	0.353	1.037
Median	0.158	0.169	0.225	0.424	1.313
Q3	0.158	0.176	0.370	0.488	1.667
Low Outliers	0%	0%	0%	0%	0%
High Outliers	8%	12%	8%	0%	4%

Table A.16: LAFO run time (minutes) summary statistics for varied demand multipliers on Anaheim, 25 random instances at each multiplier

A.4.2 Anaheim – Accuracy Gap Comparisons

Statistic	Demand Multiplier				
	0.25	0.50	1.00	2.00	4.00
Mean	0.1%	0.4%	0.0%	0.3%	0.1%
Q1	0.0%	0.0%	0.0%	0.0%	0.0%
Median	0.0%	0.0%	0.0%	0.0%	0.0%
Q3	0.0%	0.0%	0.0%	0.0%	0.0%
Low Outliers	0%	0%	0%	0%	0%
High Outliers	8%	20%	12%	20%	16%

Table A.17: BS accuracy gap summary statistics for varied demand multipliers on Anaheim, 25 random instances at each multiplier

Statistic	Demand Multiplier				
	0.25	0.50	1.00	2.00	4.00
Mean	1.5%	0.8%	1.7%	4.1%	6.3%
Q1	0.0%	0.0%	0.0%	0.0%	0.0%
Median	0.1%	0.0%	0.1%	0.2%	0.9%
Q3	0.7%	0.6%	1.4%	3.8%	4.5%
Low Outliers	0%	0%	0%	0%	0%
High Outliers	12%	16%	12%	12%	12%

Table A.18: SA accuracy gap summary statistics for varied demand multipliers on Anaheim, 25 random instances at each multiplier

Statistic	Demand Multiplier				
	0.25	0.50	1.00	2.00	4.00
Mean	26.3%	20.0%	23.1%	25.0%	36.2%
Q1	11.7%	10.0%	7.7%	8.2%	11.2%
Median	19.3%	19.4%	17.2%	19.4%	29.1%
Q3	34.9%	27.0%	31.0%	39.1%	57.7%
Low Outliers	0%	0%	0%	0%	0%
High Outliers	8%	0%	4%	0%	4%

Table A.19: LASR accuracy gap summary statistics for varied demand multipliers on Anaheim, 25 random instances at each multiplier

Statistic	Demand Multiplier				
	0.25	0.50	1.00	2.00	4.00
Mean	31.0%	24.0%	29.4%	29.8%	47.1%
Q1	18.2%	11.1%	11.9%	15.7%	14.7%
Median	27.7%	18.7%	21.8%	27.5%	26.6%
Q3	41.5%	32.1%	42.8%	39.5%	70.6%
Low Outliers	0%	0%	0%	0%	0%
High Outliers	4%	8%	0%	4%	4%

Table A.20: LAFO accuracy gap summary statistics for varied demand multipliers on Anaheim, 25 random instances at each multiplier

A.4.3 Berlin-Mitte-Center – Run Time Comparisons

Statistic	Demand Multiplier				
	0.25	0.50	1.00	2.00	4.00
Mean	4.156	4.300	5.127	5.958	8.892
Q1	3.387	3.275	3.543	5.136	7.275
Median	3.955	3.903	4.692	5.765	8.379
Q3	4.538	5.397	5.853	6.502	10.190
Low Outliers	0%	0%	0%	0%	0%
High Outliers	8%	0%	4%	8%	4%

Table A.21: BS run time (minutes) summary statistics for varied demand multipliers on BMC, 25 random instances at each multiplier

Statistic	Demand Multiplier				
	0.25	0.50	1.00	2.00	4.00
Mean	0.445	0.452	0.679	0.691	1.233
Q1	0.358	0.359	0.433	0.620	1.152
Median	0.430	0.457	0.523	0.682	1.221
Q3	0.493	0.567	0.735	0.789	1.408
Low Outliers	0%	0%	0%	0%	0%
High Outliers	8%	0%	8%	0%	0%

Table A.22: SA run time (minutes) summary statistics for varied demand multipliers on BMC, 25 random instances at each multiplier

Statistic	Demand Multiplier				
	0.25	0.50	1.00	2.00	4.00
Mean	0.175	0.175	0.475	0.284	0.544
Q1	0.161	0.172	0.193	0.274	0.472
Median	0.163	0.175	0.197	0.285	0.527
Q3	0.167	0.179	0.211	0.291	0.599
Low Outliers	4%	0%	0%	1%	0%
High Outliers	12%	0%	4%	8%	0%

Table A.23: LASR run time (minutes) summary statistics for varied demand multipliers on BMC, 25 random instances at each multiplier

Statistic	Demand Multiplier				
	0.25	0.50	1.00	2.00	4.00
Mean	0.175	0.175	0.475	0.284	0.544
Q1	0.161	0.172	0.193	0.274	0.472
Median	0.163	0.175	0.197	0.285	0.527
Q3	0.167	0.179	0.211	0.291	0.599
Low Outliers	4%	0%	0%	1%	0%
High Outliers	12%	0%	4%	8%	0%

Table A.24: LAFO run time (minutes) summary statistics for varied demand multipliers on BMC, 25 random instances at each multiplier

A.4.4 Berlin-Mitte-Center – Accuracy Gap Comparisons

Statistic	Demand Multiplier				
	0.25	0.50	1.00	2.00	4.00
Mean	0.0%	0.1%	0.1%	0.1%	0.1%
Q1	0.0%	0.0%	0.0%	0.0%	0.0%
Median	0.0%	0.0%	0.0%	0.0%	0.0%
Q3	0.0%	0.0%	0.0%	0.0%	0.0%
Low Outliers	0%	0%	0%	0%	0%
High Outliers	20%	20%	16%	16%	20%

Table A.25: BS accuracy gap summary statistics for varied demand multipliers on BMC, 25 random instances at each multiplier

Statistic	Demand Multiplier				
	0.25	0.50	1.00	2.00	4.00
Mean	1.4%	2.1%	2.6%	2.6%	1.9%
Q1	0.0%	0.0%	0.0%	0.0%	0.0%
Median	0.1%	0.0%	0.0%	0.1%	0.0%
Q3	1.0%	0.2%	0.4%	3.5%	2.1%
Low Outliers	0%	0%	0%	0%	0%
High Outliers	12%	20%	20%	8%	8%

Table A.26: SA accuracy gap summary statistics for varied demand multipliers on BMC, 25 random instances at each multiplier

Statistic	Demand Multiplier				
	0.25	0.50	1.00	2.00	4.00
Mean	18.6%	18.5%	21.2%	20.9%	24.4%
Q1	7.0%	9.2%	12.1%	8.8%	12.4%
Median	12.7%	16.0%	19.3%	16.8%	18.5%
Q3	29.1%	21.2%	30.0%	24.1%	26.4%
Low Outliers	0%	0%	0%	0%	0%
High Outliers	0%	12%	0%	12%	8%

Table A.27: LASR accuracy gap summary statistics for varied demand multipliers on BMC, 25 random instances at each multiplier

Statistic	Demand Multiplier				
	0.25	0.50	1.00	2.00	4.00
Mean	32.6%	34.0%	26.0%	27.2%	28.1%
Q1	11.9%	13.2%	13.1%	11.4%	19.4%
Median	19.8%	24.5%	22.4%	23.4%	25.5%
Q3	50.7%	35.7%	37.7%	34.7%	34.7%
Low Outliers	0%	0%	0%	0%	0%
High Outliers	0%	4%	0%	4%	8%

Table A.28: LAFO accuracy gap summary statistics for varied demand multipliers on BMC, 25 random instances at each multiplier

Appendix B: Multi-crew Summary Statistics

B.1 Direct Solution vs Post-processing Summary Statistics

The below tables correspond to Figure 3.4.

Statistic	Direct Solution			Post-process		
	BS	SA	SQG	BS	SA	SQG
Mean	49.4%	0.1%	119.5%	10.1%	7.5%	98.8%
Q1	20.9%	0.0%	45.2%	4.0%	2.8%	39.8%
Median	50.1%	0.0%	130.2%	8.3%	5.7%	101.5%
Q3	67.2%	0.0%	162.6%	12.7%	9.2%	143.0%
Low Outliers	0%	0%	0%	0%	0%	0%
High Outliers	0%	10%	0%	10%	10%	0%

Table B.1: Multi-crew accuracy gap summary statistics for direct solution vs. post-processing for two crews on Anaheim with 20 broken links, ten random instances

Statistic	Direct Solution			Post-process		
	BS	SA	SQG	BS	SA	SQG
Mean	42.8%	0.9%	93.1%	8.5%	7.8%	87.0%
Q1	21.2%	0.0%	42.5%	0.2%	2.4%	33.5%
Median	42.9%	0.0%	116.3%	6.1%	5.4%	91.5%
Q3	57.1%	0.0%	138.6%	14.4%	14.4%	133.2%
Low Outliers	0%	0%	0%	0%	0%	0%
High Outliers	0%	20%	0%	0%	0%	0%

Table B.2: Multi-crew accuracy gap summary statistics for direct solution vs. post-processing for three crews on Anaheim with 20 broken links, ten random instances

Statistic	Direct Solution			Post-process		
	BS	SA	SQG	BS	SA	SQG
Mean	36.6%	0.1%	119.5%	10.1%	7.5%	98.8%
Q1	20.9%	0.0%	46.2%	0.9%	1.7%	24.2%
Median	35.4%	0.0%	106.1%	6.6%	8.4%	74.2%
Q3	54.9%	0.1%	148.5%	13.8%	12.0%	121.8%
Low Outliers	0%	0%	0%	0%	0%	0%
High Outliers	0%	10%	0%	10%	0%	0%

Table B.3: Multi-crew accuracy gap summary statistics for direct solution vs. post-processing for four crews on Anaheim with 20 broken links, ten random instances

B.2 Multi-crew Simulated Annealing Summary Statistics

The below tables correspond to Figure 3.8. Accuracy gap summary statistics are not shown for simulated annealing because the SA heuristic uses the LAFO/LASR and IF solutions in selecting its starting point. While it is structurally possible for one of the other tested methods (SQG, LZG, SPT) to obtain a lower objective value than the SA heuristic, this scenario was not observed in any of the test instances.

Statistic	Number of Broken Links									
	10	20	30	40	50	60	70	80	90	100
Mean	0.01	0.06	0.31	0.78	1.84	3.13	4.15	7.11	10.44	13.11
Q1	0.00	0.04	0.23	0.60	1.37	2.66	3.35	5.47	9.65	11.98
Median	0.01	0.05	0.29	0.75	1.62	3.00	3.81	6.36	10.11	12.97
Q3	0.01	0.07	0.38	0.84	2.10	3.81	4.14	7.23	11.39	14.31
Low Outliers	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%
High Outliers	0%	0%	0%	10%	10%	0%	10%	10%	0%	0%

Table B.4: Multi-crew SA run time (hours) summary statistics on Anaheim; ten random instances each for 10–80 broken links; five random instances for 90, 100 broken links

Statistic	Number of Broken Links									
	10	20	30	40	50	60	70	80	90	100
Mean	0.00	0.04	0.16	0.46	0.90	1.63	3.22	4.54	9.50	10.16
Q1	0.00	0.03	0.13	0.38	0.77	1.26	2.97	3.68	8.14	8.92
Median	0.00	0.04	0.16	0.44	0.87	1.56	3.32	4.60	10.18	10.11
Q3	0.00	0.05	0.20	0.55	0.97	2.01	3.65	5.41	10.53	11.43
Low Outliers	0%	0%	0%	0%	0%	0%	10%	0%	0%	0%
High Outliers	0%	0%	0%	0%	10%	0%	0%	0%	0%	0%

Table B.5: Multi-crew SA run time (hours) summary statistics on BMC; ten random instances each for 10–80 broken links; five random instances for 90, 100 broken links

B.3 Multi-crew LAFO/LASR Summary Statistics

The below tables correspond to Figure 3.8.

Statistic	Number of Broken Links									
	10	20	30	40	50	60	70	80	90	100
Mean	0.13	0.44	1.23	2.00	9.52	5.02	12.15	8.80	15.36	13.48
Q1	0.09	0.40	0.94	1.79	2.92	4.46	4.93	7.11	10.28	11.15
Median	0.12	0.43	1.21	2.00	3.31	4.83	5.84	7.65	14.43	11.90
Q3	0.15	0.49	1.46	2.15	9.33	5.47	8.98	8.82	20.91	16.62
Low Outliers	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%
High Outliers	0%	0%	0%	0%	20%	0%	10%	10%	0%	0%

Table B.6: Multi-crew LAFO/LASR run time (minutes) summary statistics on Anaheim; ten random instances each for 10–80 broken links; five random instances for 90, 100 broken links

Statistic	Number of Broken Links									
	10	20	30	40	50	60	70	80	90	100
Mean	0.07	0.33	0.74	1.43	2.37	3.35	5.82	6.66	10.70	11.21
Q1	0.07	0.29	0.72	1.31	2.22	3.07	4.54	6.18	9.91	10.84
Median	0.07	0.31	0.73	1.43	2.28	3.24	4.90	6.48	10.32	11.29
Q3	0.08	0.38	0.76	1.53	2.55	3.58	5.53	7.04	11.69	11.54
Low Outliers	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%
High Outliers	0%	0%	0%	0%	0%	0%	10%	0%	0%	0%

Table B.7: Multi-crew LAFO/LASR run time (minutes) summary statistics on BMC; ten random instances each for 10–80 broken links; five random instances for 90, 100 broken links

Statistic	Number of Broken Links									
	10	20	30	40	50	60	70	80	90	100
Mean	10%	12%	16%	23%	13%	28%	23%	33%	23%	37%
Q1	1%	2%	5%	9%	7%	18%	12%	20%	19%	22%
Median	7%	11%	11%	18%	13%	28%	22%	35%	22%	35%
Q3	15%	18%	31%	33%	17%	35%	33%	46%	27%	54%
Low Outliers	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%
High Outliers	0%	0%	0%	0%	10%	0%	0%	0%	0%	0%

Table B.8: Multi-crew LAFO/LASR accuracy gap summary statistics on Anaheim; ten random instances each for 10–80 broken links; five random instances for 90, 100 broken links

Statistic	Number of Broken Links									
	10	20	30	40	50	60	70	80	90	100
Mean	9%	12%	12%	24%	24%	38%	27%	34%	32%	34%
Q1	0%	2%	5%	11%	14%	25%	24%	23%	22%	27%
Median	3%	11%	8%	25%	21%	28%	27%	32%	32%	30%
Q3	18%	22%	13%	32%	33%	47%	32%	42%	42%	43%
Low Outliers	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%
High Outliers	0%	0%	10%	0%	0%	10%	0%	0%	0%	0%

Table B.9: Multi-crew LAFO/LASR accuracy gap summary statistics on BMC; ten random instances each for 10–80 broken links; five random instances for 90, 100 broken links

Bibliography

Ravindra K. Ahuja, Thomas L. Magnanti, and James B. Orlin. *Network Flows*. Prentice Hall, Englewood Cliffs, NJ, 1993.

Edward J. Anderson, Celia A. Glass, and Chris N. Potts. Machine Scheduling. In E. Aarts and J. Lenstra, editors, *Local Search in Combinatorial Optimization*, chapter 2, pages 361–414. Princeton University Press, 2003.

Hillel Bar-Gera. Origin-based algorithm for the traffic assignment problem. *Transportation Science*, 36(4):398–417, 2002.

Hillel Bar-Gera. Traffic assignment by paired alternative segments. *Transportation Research Part B: Methodological*, 44(8–9):1022–1046, 2010.

Martin Beckmann, C. B. McGuire, and Christopher B. Winsten. *Studies in the Economics of Transportation*. Yale University Press, New Haven, CT, 1956.

Paolo Bocchini and Dan M. Frangopol. Restoration of bridge networks after an earthquake: Multicriteria intervention optimization. *Earthquake Spectra*, 28(2): 427–455, 2012.

Ihor O. Bohachevsky, Mark E. Johnson, and Myron L. Stein. Generalized simulated annealing for function optimization. *Technometrics*, 28(3):209–217, 1986.

Stephen D. Boyles. TAP-B Implementation. Accessed November 19, 2022, `https://github.com/spartalab/tap-b/`, 2022.

Stephen D. Boyles, Nicholas E. Lownes, and Avinash Unnikrishnan. *Transportation Network Analysis*. 0.91 vol. 1, 2023.

Michel Bruneau, Stephanie E. Chang, Ronald T. Eguchi, George C. Lee, Thomas D. O'Rourke, Andrei M. Reinhorn, Masanobu Shinozuka, Kathleen Tierney, William A. Wallace, and Detlof von Winterfeldt. A framework to quantitatively assess and enhance the seismic resilience of communities. *Earthquake Spectra*, 19(4):733–752, 2003.

James Bruno, Edward G. Coffman Jr., and Ravi Sethi. Scheduling independent tasks to reduce mean finishing time. *Communications of the ACM*, 17(7):382–387, 1974.

Lichun Chen and Elise Miller-Hooks. Resilience: An indicator of recovery capability in intermodal freight transport. *Transportation Science*, 46(1):109–123, 2012.

Yanan Cheng and Zilin Zhang. A resilience-based routing planning and scheduling model for post-disaster transportation network recovery with multiple repair teams. In *4th International Conference on System Reliability and Safety Engineering*, pages 97–103. IEEE, 2022.

Robert B. Dial. A path-based user-equilibrium traffic assignment algorithm that obviates path storage and enumeration. *Transportation Research Part B*, 40(10):917–936, 2006.

Michael Florian, Isabelle Constantin, and Dan Florian. A new look at projected gradient method for equilibrium assignment. *Transportation Research Record*, 2090(1):10–16, 2009.

Marguerite Frank and Philip Wolfe. An algorithm for quadratic programming. *Naval Research Logistics Quarterly*, 3(1-2):95–110, 1956.

Alberto Franzin and Thomas Stützle. Revisiting simulated annealing: A component-based analysis. *Computers & Operations Research*, 104:191–206, 2019.

Michael R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., New York, 1990.

Can Gokalp and Abigail J. Crocker. Recovery sequencing repository. Accessed March 4, 2024, `https://github.com/ajcrocker14/optimal_recovery_sequencing`, 2024.

Can Gokalp, Priyadarshan N. Patil, and Stephen D. Boyles. Post-disaster recovery sequencing strategy for road networks. *Transportation Research Part B*, 153:228–245, 2021.

R. L. Graham. Bounds on multiprocessing timing anomalies. *Journal of Applied Mathematics*, 17(2):416–429, 1969.

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. *Annals of Discrete Mathematics*, 5:287–326, 1979.

Jürgen Hackle, Bryan T. Adey, and Nam Lethanh. Determination of near-optimal restoration programs for transportation networks following natural hazard events using simulated annealing. *Computer-Aided Civil and Infrastructure Engineering*, 33, 2018.

R. Jayakrishnan, Wei T. Tsai, Joseph N. Prashker, and Subodh Rajadhyaksha. A faster path-based algorithm for traffic assignment. Technical report, University of California, Transportation Center, 1994.

Richard M. Karp. Reducibility among combinatorial problems. In Raymond E. Miler, James W. Thatcher, and John D. Bohlinger, editors, *Complexity of Computer Computations*, pages 85–103. Plenum Press, New York, 1972.

Tsuyoshi Kawaguchi and Seiki Kyan. Worst case bound of an LRF schedule for the mean weighted flow-time problem. *SIAM Journal on Computing*, 15(4): 1119–1129, 1986.

Scott Kirkpatrick, C. Daniel Gelatt Jr., and Mario P. Vecchi. Optimization by simulated annealing. *Science*, 220(4598):671–680, 1983.

J. K. Lenstra, A.H.G. Rinnooy Kan, and P. Brucker. Complexity of machine scheduling problems. *Annals of Discrete Mathematics*, 1:343–362, 1977.

Neale F. Lunderville. Irene recovery report: A stronger future. In: A Representative To the Governor of Vermont, State of Vermont, 2012.

Merriam-Webster. Resilience. In *Merriam-Webster.com dictionary*. Accessed February 6, 2024, `https://www.merriam-webster.com/dictionary/resilience`.

Eric Merschman, Mehrnaz Doustmohammadi, Abdullahi M. Salman, and Michael Anderson. Postdisaster decision framework for bridge repair prioritization to improve road network resilience. *Transportation Research Record*, 2674(3):81–92, 2020.

Nicholas Metropolis, Arianna W. Rosenbluth, Marshall N. Rosenbluth, Augusta H. Teller, and Edward Teller. Equation of state calculations by fast computing machines. *Journal of Chemical Physics*, 21(6):1087–1092, 1953.

Maria Mitradjieva and Per Olov Lindberg. The stiff is moving—conjugate direction frank-wolfe methods with applications to traffic assignment. *Transportation Science*, 47(2):280–293, 2013.

ManWo Ng and Paul Schonfeld. Sequencing interdependent disruption recovery projects: exact solution via network flow reformulation. *Tranportation Research Part D*, 115:103565, 2023.

Michael Patriksson. *The Traffic Assignment Problem: Models and Methods*. VSP, Utrecht, The Netherlands, 1994.

Michael L. Pinedo. *Scheduling: Theory, Algorithms, and Systems*. Springer, New York, fifth edition, 2016.

Warren B. Powell and Yosef Sheffi. The convergence of equilibrium algorithms with hpredetermined step sizes. *Transportation Science*, 16(1):45–55, 1982.

David Rey and Hillel Bar-Gera. Long-term scheduling for road network disaster recovery. *International Journal of Disaster Risk Reduction*, 42:101353, 2020.

David Rey, Hillel Bar-Gera, Vinayak V. Dixit, and S. Travis Waller. A branch-and-price algorithm for the bilevel network maintenance scheduling problem. *Transportation Science*, 53(5):1455–1478, 2019.

Sartaj K. Sahni. Algorithms for scheduling independent tasks. *Journal of the ACM*, 23(1):116–127, 1976.

Martin Skutella and Gerhard J. Woeginger. A PTAS for minimizing the total weighted completion time on identical parallel machines. *Mathematics of Operations Research*, 25(1):63–75, 2000.

Wayne E. Smith. Various optimizers for single-stage production. *Naval Research and Logistics Quarterly*, 3:59–66, 1956.

Harold Szu and Ralph Hartley. Fast simulated annealing. *Physics Letters A*, 122(3):157–162, 1987.

Transportation Networks for Research Core Team. Transportation Networks for Research. Accessed November 19, 2022, `https://github.com/bstabler/TransportationNetworks`, 2022.

Eric D. Vugrin, Mark A. Turnquist, and Nathanael J. K. Brown. Optimal recovery sequencing for enhanced resilience and service restoration in transportation networks. *International Journal of Critical Infrastructures*, 10(3/4):218–246, 2014.

John Glen Wardrop. Road Paper. Some theoretical aspects of road traffic research. *Proceedings of the Institution of Civil Engineers*, 1(3):325–362, 1952.

Jun Xie and Chi Xie. An improved tapas algorithm for the traffic assignment problem. In *17th International IEEE Conference on Intelligent Transportation Systems (ITSC)*, pages 2336–2341, 2014.

Qing Ye and Satish V. Ukkusuri. Resilience as an objective in the optimal reconstruction sequence for transportation networks. *Journal of Transportation Safety & Security*, 7(1):91–105, 2015.

Ning Zhang, Alice Alipour, and Laura Coronel. Application of novel recovery techniques to enhance the resilience of transportation networks. *Transportation Research Record*, 2672(1):138–147, 2018a.

Weili Zhang, Naiyu Wang, and Charles Nicholson. Resilience-based post-disaster recovery strategies for road-bridge networks. *Structure and Infrastructure Engineering*, 13(11):1404–1413, 2017.

Weili Zhang, Naiyu Wang, Charles Nicholson, and Mohammad Hadikhan Tehrani. A stage-wise decision framework for transportation network resilience planning. arXiv preprint, 2018b.

Yaoming Zhou, Junwei Wang, and Hai Yang. Resilience of transportation systems: concepts and comprehensive review. *IEEE Transactions on Intelligent Transportation Systems*, 20(12):4262–4276, 2019.

W.L. Zhuang, Z.Y. Liu, and J.S. Jiang. Earthquake-induced damage analysis of highway bridges in Wenchuan earthquake and countermeasures. *Chinese Journal of Rock Mechanics and Engineering*, 28:1377–1387, 2009.

Vita

Abigail June Crocker is originally from Houston, Texas, and graduated from The John Cooper School in The Woodlands, Texas. She matriculated from the United States Military Academy in 2014 with a Bachelors of Science in Civil Engineering with Honors. In 2018, she earned a Masters of Science in Engineering Management from Missouri University of Science of Technology. As part of the U.S. Army Advanced Civil Schooling program, she entered The Graduate School at the University of Texas at Austin in August 2021, and joined Dr. Stephen Boyles' research group, SPARTA Lab, in June 2022.

Email Address: ajcrocker14@gmail.com

This dissertation was typeset with LaTeX[†] by the author.

[†]LaTeX is a document preparation system developed by Leslie Lamport as a special version of Donald Knuth's TeX Program.