Scheduling spatially distributed jobs with degradation: Application to pothole repair

Article *in* Socio-Economic Planning Sciences · August 2020

DOI: 10.1016/j.seps.2020.100904

Scheduling spatially distributed jobs with degradation: application to pothole repair

Fatemeh Aarabi and Rajan Batta

Department of Industrial and Systems Engineering, University at Buffalo, Buffalo, NY 14260, USA

Abstract

This paper considers scheduling spatially distributed jobs with degradation. A mixed integer programming (MIP) model is developed for the linear degradation case in which no new jobs arrive. Properties of the model are analyzed, following which three heuristics are developed, enhanced greedy, chronological decomposition and simulated annealing. Numerical tests are conducted to: (i) establish limits of the exact MIP solution, (ii) identify the best heuristic based on an analysis of performance on small problem instances for which exact solutions are known, (iii) solve large problem instances and obtain lower bounds to establish solution quality, and (iv) study the effect of three key model parameters. Findings from our computational experiments indicate that: (i) exact solutions are limited to instances with less than 14 jobs; (ii) the enhanced greedy heuristic followed by the application of the simulated annealing heuristic yields high quality solutions for large problem instances in reasonable computation time; and (iii) the factors "degradation rate" and "work hours" have a significant effect on the objective function. To demonstrate applicability of the model, a case study is presented based on a pothole repair scenario from Buffalo, New York, USA. Findings from the case study indicate that scheduling spatially dispersed jobs with degradation such as potholes requires: (i) careful consideration of the number of servers assigned, degradation rate and depot location; (ii) appropriate modeling of continuously arriving jobs; and (iii) appropriate incorporation of equity consideration.

Keywords: Scheduling, Enhanced greedy, Pothole repair, Chronological decomposition, Simulated annealing.

1 Introduction

In classical job scheduling, the processing time of jobs is considered constant, c.f. Gawiejnowicz (2020). However, in many real-world applications processing time is an increasing function of when the job is started. Firefighting and medical emergencies are instances of jobs where delayed response leads to a larger amount of processing time Wang et al. (2018). The application that motivated our work is pothole repair. It is a classic case of scheduling deteriorating jobs since the chemicals that bind pavement deteriorate over time making pothole repair an increasing function of time, c.f. Nicholls et al. (2019). Providing humanitarian relief to victims after a disaster also belongs to this class of scheduling problems, since more time for treatment is needed if medical attention is delayed. In these situations, the actual processing time is larger than

the normal service time and is a function of degradation rate and start time, c.f. Kunnathur and Gupta (1990).

Another set of applications of scheduling problems are jobs that are distributed spatially. In many machine scheduling problems, the location of machines are considered fixed and jobs move through them, but there are many occasions that jobs are fixed and machines are mobile, c.f. papers by Chen et al. (1998); Lenstra and Shmoys (2020). Emergency response is an example where jobs are distributed over a spatial region and mobile servers serve them (Caunhye et al. (2012); Larson (2005); Wex et al. (2012)). Other examples include humanitarian relief supply (Lin (2010)) and pothole repair. All of these examples need to consider the network-based nature of the problem and use solution algorithms that consider distances that servers have to traverse to access jobs. The most basic algorithm that is useful in solving spatial problems is the traveling salesman problem, c.f. a recent review paper by Raman and Gill (2017).

In many scheduling problems, it is common that job processing time is a function of its starting time. The first set of researchers to study this important area of job scheduling were Browne and Yechiali (1990) and Gupta and Gupta (1988) in two independent research studies of scheduling degrading jobs on a single machine. In the literature, there are two distinct groups of job scheduling in which processing time is a function of the start time: scheduling with degradation rate and scheduling with learning effects. Scheduling with degradation rate focuses on situations where the processing time for jobs gets larger with the passage of time Browne and Yechiali (1990), whereas scheduling with learning effects deals with situations where the job processing time gets shorter by postponing the start time. Biskup (1999) was the first to address the concept of learning effect in scheduling. Our research is focused on the degradation rate case.

In this paper, we use an MIP formulation to find the optimal schedule for a set of spatially distributed deteriorating jobs. Our formulation is intended for situations (such as pothole repair and providing humanitarian relief to victims) where jobs experience degradation and are also spatially dispersed, requiring servers to travel from one job to the next to provide service. Server travel time impacts job start times, which in turn impacts processing time due to job degradation. Since processing of jobs and travel of servers are both important, the objective we select is to minimize the sum of the processing times of all jobs plus the sum of the travel time for all servers.

Our main contributions are as follows:

1. We present a scheduling problem for spatially distributed degrading jobs.
2. We model job degradation continuously over time such that it occurs even when servers are idle.
3. We demonstrate an application of our model for pothole repair, and explore features that allow consideration of continuously arriving jobs and equity considerations between impacted regions.

The rest of this paper is organized as follow: Section 2 contains our literature review. Section 3 contains a motivating example. Section 4 develops an MIP model for scheduling spatially distributed deteriorating jobs. Section 5 explains our solution methodologies for both the cases of linear and nonlinear processing time functions. Section 6 presents computational testing. In section 7 we present an application of our model to pothole repair in the City of Buffalo, New York, USA. Finally, section 8 presents our summary, conclusions and future research directions.

2 Literature review

In our study we consider jobs dispersed over a spatial region with degradation effect. For this reason we now review the literature in scheduling jobs that are dispersed over a spatial region. In this context there are many significant and practical problems. One such example is the vehicle routing problem (VRP) Toth and Vigo (2002). Time-constrained scheduling problems and m-machine scheduling problems are a generalization of the vehicle routing problem (VRP) that were first introduced by Christofides and Eilon (1969). Vehicle routing problem with time windows were first presented as case studies by Madsen (1976) and Knight and Hofer (1968).

Job scheduling problem with degradation effect has been studied extensively. Since we use an MIP modeling approach we now review contributions in the area of scheduling degrading jobs that use an MIP approach. Wang et al. (2018) study the single machine scheduling with degradation rate and use an MIP approach to minimize completion time. Bahalke et al. (2010) propose an MIP model to solve the problem of degrading job scheduling with setup times. Toksarı and Güner (2010) propose an MIP approach for a problem with both learning and degradation effects. Eren (2009) developed an MIP model for the single machine scheduling problem with learning effects with setup times and degrading effects on job processing times. Scheduling of spatially dispersed jobs have drawn attention from researchers, an example in railway maintenance scheduling is the recent paper by Pour et al. (2019). There has also been interest in solving very large-scale vehicle routing problems by developing parallel heuristic methods, c.f. the recent paper Tu et al. (2017).

One extension of VRP is inclusion of time window considerations, labeled as the class of problems VRPTW. In general, VRPTW problems are NP-hard (Solomon (1987)). In most of the research on VRPTW, service time is ignored or considered constant. However, service time is often a function of when service is provided. In degradation problems, service time is an increasing function of time. In the literature, time-dependent VRPTW problems (TDVRPTW) focus on modeling time dependent travel time. The time-dependent VRPTW was first time introduced by Malandraki (1989). Ichoua et al. (2003) uses Tabu Search to solve the time-dependent VRPTW problem. In their model the horizon is discretized into different time periods and travel speed changes over each time period. The innovation in their work is implementation of FIFO, which was not considered in previous studies. They use parallel Tabu Search to solve the problem. Jung and Haghani (2001)propose a Genetic Algorithm (GA) heuristic to solve the same asymmetric MIP model introduced by Malandraki (1989). Solving the asymmetric VRP by k-opt exchange is difficult since travel time not only is dependent on the start time but also on the direction. Donati et al. (2008) propose an Ant Colony method to solve the problem, building on the work by Ichoua et al. (2003) by incorporating capacity constraints and utilizing an objective function that minimizes the number of vehicles. In 2005, Ibaraki et al. (2005) proposed an efficient algorithm to deal with general time window constraints in TDVRPTW. Hashimoto et al. (2008) developed a model that generalizes that of Ichoua et al. (2003) by allowing a flexible time penalty function for each customer. Their model minimizes both travel time and cost functions that are a function of start time. In 2011, Balseiro et al. (2011) extended the model proposed by Malandraki (1989) by considering an objective function that simultaneously minimizes the number of routes and the total time, while satisfying the FIFO property. The authors present a model which considers time dependent travel times with satisfying the "first-in–first-out" property. The FIFO property ensures that if a vehicle depart node i for node j at time T, any other vehicle that departs node i for node j at a time greater than T, will arrive later to node j. For more research in the field of time dependent VRP problem refer to a review by Gendreau et al. (2015). As far as solution methods, Dabia et al. (2013) present a branch and price algorithm to solve the

TDVRPTW. Also, Huart et al. (2016) propose a column generation approach to solve VRPTW. An excellent comparison of heuristic approaches available to solve the VRPTW is contained in Corstjens et al. (2020).

Of the surveyed research, only Malandraki (1989), Jung and Haghani (2001), Balseiro et al. (2011), Dabia et al. (2013), and Huart et al. (2016) use MIP models to solve the VRPTW. To the our best knowledge none of the time dependent VRP (TDVRP) models consider processing time as a function of start time. The only related study is done by Taş et al. (2016) which is a time dependent traveling salesman problem (TDTSP), a special case of the TDVRP when the fleet size is equal to one.

In contrast to the above cited literature, we consider spatial distribution and degradation of jobs simultaneously in our MIP model. Due to spatial dispersion, servers have to travel from their current job to the next, which takes time to accomplish and thereby changes the processing time of the next job due to the presence of the degradation effect. Our motivation to study the case of spatially dispersed degrading jobs is drawn from the pothole repair context, in which potholes are spatially dispersed over a region and are repair jobs that degrade with time.

3 Motivating example

Suppose there are 20 jobs with linear degradation rates and two servers available to process these jobs in two days. Each day, both servers leave the depot and visit their assigned jobs in the specified order and return back to the depot. The location of jobs over the region is shown in Figure 1. When moving to or between jobs, both servers travel at a fixed speed of 2 miles per hour. Travel time between jobs is assumed to be equal to the Euclidean distance between the coordinates of the jobs divided by the fixed speed of the server. The coordinates and processing time parameters related to each job are presented in Table 1. The objective is to minimize the sum of the processing times of all of the jobs plus the sum of the travel times for both servers. The maximum on-duty time for each server (including the travel time over the region and the service time) cannot exceed T=12 hours in either day, which necessitates that both servers return to the depot (on both days) in a time less than or equal to T. We assume that each job degrades over 24 hours a day, which means that jobs also degrade during the 12-hour gap between the end of the working period for day 1 and the beginning of the working period for day 2.

Table 1: Fixed service time and degradation rates of 10 jobs

Jobs	1	2	3	4	5	6	7	8	9	10
Degradation rate (α)	0.02	0.03	0.001	0.01	0.01	0.1	0.02	0.3	0.01	0.01
Initial service time (a)	2.1	2.3	2	2.5	2.1	1.5	2.7	1.3	2	1.9
Coordinate	(6,6)	(7,4)	(6,3)	(5,1)	(3,2)	(1,2)	(2,3)	(2,4)	(3,6)	(4,5)

The time when a server arrives at a job determines its processing time. To identify a schedule which minimizes the sum of processing times of all jobs, we enumerate all feasible routes that visit the assigned jobs. We define a feasible route as one in which the travel time to sequentially visit the jobs in the route plus the sum of the processing times of the jobs is less than or equal to 12 hours, for each day of the route. Figure 1 illustrates an example of a feasible route $A = [[0-2-7-0], [0-8-6-5-0]], [[0-10-3-4-0], [0-9-1-0]]$. We note that 0 signifies the depot. On this route, server 1 completes jobs 2 and 7 (in that order) on day 1

and jobs 10, 3 and 4 (in that order) on day 2. Also, on this route, server 2 completes jobs 8, 6 and 5 (in that order) on day 1 and jobs 9 and 1 (in that order) on day 2.

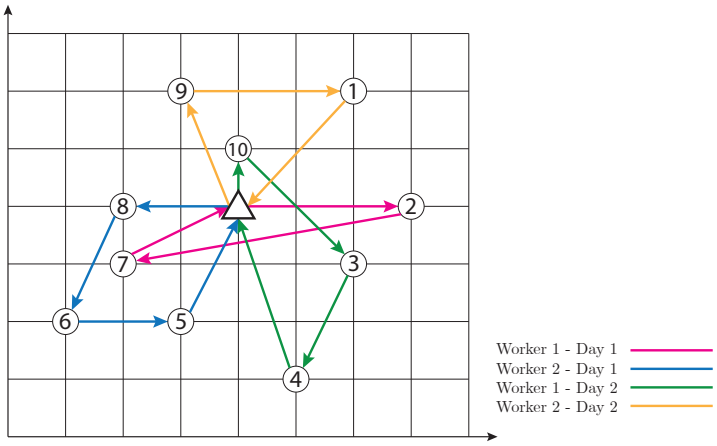

Figure 1: An example of a feasible schedule

The objective function for route A is equal to 40.44 hours. The optimal route for our example is $O = [[0-1-2-3-0],[0-8-6-7-0]],[[0-5-4-0],[0-5-6-0]]$ and has a much lower objective function value of 36.58 hours. As expected, jobs with high degradation rate are processed earlier in the optimal route (notably job 8).

To gain further insight into our problem we revisit the above example with some changes in the parameter settings.

First, we consider the case when there is no job degradation, i.e. $\alpha = 0$ for all jobs. The optimal route is: $B = [[0-1-9-10-0],[0-2-3-4-0]],[[0-5-6-7-8-0]]$, with an objective function value of 33.76. Since there is no job degradation, only the distances between jobs determine the optimal route, like in a traditional VRP.

Second, we increase the impact of travel time by decreasing server speed to 1. The optimal route is $C = [[0-1-9-10-0],[0-8-6-7-0]],[[0-2-3-0],[0-5-4-0]]$ with an objective function value of 43.88. It is noted that the sequence of jobs has changed with each server visiting nearby jobs so as to minimize the impact of travel time.

Third, we study the impact of (dramatically) increasing the degradation rate of a single job. The new degradation rate of job 9 is α equal to 0.5, which is 50 times its original degradation rate. The optimal route is $D = [[0-9-10-0],[0-8-6-7-0]],[[0-2-1-0],[0-5-4-3-0]]$ with an objective function value of the sum of 38.18. Job 9 is completed first so as to avoid an increase in its processing time, as expected.

In general, the optimal route is some combination of VRP considerations (sum of travel times for servers) as well as scheduling of degrading job considerations (sum of job processing times).

4 MIP formulation and properties for the case of linear degradation

In this section we develop an MIP formulation for our problem (for the case of linear degradation rate and in which no new jobs arrive) and establish a key property which motivates the development of an enhanced greedy heuristic algorithm.

4.1 Model and notation

Consider the scheduling of n spatially distributed jobs with specified degradation rates and k servers. Let $G = (N, A)$ be a complete undirected graph with node set $N = \{0, 1, 2, ..., n, n+1\}$ and set of links $A = \{(i, j) \mid i, j \in N, i \neq j\}$. In this graph, node 0 represents the beginning depot for each day and node $n+1$ represents the ending depot for each day, and other nodes represent jobs. Note that nodes 0 and $n+1$ do not have any processing time associated with them. We now focus on non-depot nodes. For node i we consider a_i as the fixed time required to process it and α_i as its degradation rate. What we mean by this is that if job i starts processing at time w, its processing time is $a_i + \alpha_i w$ (i. e. linear degradation). For every link $(i, j) \in A$ let τ_{ij} be the travel time between node i and node j. We let k denote the number of identical servers. Let m be the number of days or time periods (each 24 hours duration) over which the jobs must be processed. Each server is available only during the first T hours of each day, with $T \leq 24$. Each day, every server leaves the depot at the beginning of the day, travels to its assigned jobs in the specified sequence, processes each of its assigned jobs, and returns to the depot within T hours.

To formulate the mathematical programming model, we introduce the following decision variables. Binary variable x_{ijt} is equal to 1 if a server travels from node i to node j during day t, 0 otherwise. Continuous variable w_{it} specifies the time that a server arrives to start the job associated with node i in day t. Table 2 provides the notation associated with the parameters and decision variables used in our model formulation.

Table 2: Mathematical notation

Parameters	
$G = (N, A)$	Un-directed graph with nodes N and links A
N	Set of nodes
A	Set of links
n	Number of nodes
k	Number of servers
m	Number of days
T	Length of on-duty time of servers
a_i	Initial time required for serving node i
α_i	Degradation rate of node i
τ_{ij}	A non-negative arc travel time for each arc $(i, j) \in A$
Decision Variables	
x_{ijt}	Binary variable taking 1, if node j will be served after node i during day t .
w_{it}	Continuous variable that specifies the start time of serving node i during day t

Armed with the above notation we are ready to present an MIP formulation (Q) for scheduling spatially distributed jobs with degradation rate, for the case of linear degradation and in which no new jobs arrive:

$$(Q)\text{:}\min \sum_{i=0}^{n}\sum_{\substack{j=1\\j\neq i}}^{n+1}\sum_{t=1}^{m}(a_i+\alpha_i(w_{it}+24(t-1))x_{ijt}+\sum_{i=0}^{n}\sum_{\substack{j=1\\j\neq i}}^{n+1}\sum_{t=1}^{m}\tau_{ij}x_{ijt} \tag{1}$$

$$\text{s.t.} \quad \sum_{t=1}^{m}\sum_{\substack{j=1\\j\neq i}}^{n+1}x_{ijt}=1 \qquad \forall i\in N\backslash\{0,n+1\} \tag{2}$$

$$\sum_{j=1}^{n+1}x_{0jt}\leq k \qquad \forall t\in M \tag{3}$$

$$\sum_{i=0}^{n}x_{i(n+1)t}\leq k \qquad \forall t\in M \tag{4}$$

$$\sum_{i=0}^{n}x_{iht}-\sum_{\substack{j=1\\j\neq i}}^{n+1}x_{hjt}=0 \qquad \forall h\in N\backslash\{0,n+1\},\ t\in M \tag{5}$$

$$(w_{it}+a_i+\alpha_i(w_{it}+24(t-1))+\tau_{ij}-w_{jt})x_{ijt}\leq 0 \qquad \forall (i,j)\in A,\ t\in M \tag{6}$$

$$w_{0t}=0 \qquad \forall t\in M \tag{7}$$

$$0\leq w_{it}\leq T \qquad \forall i\in N\backslash\{0\},\ t\in M \tag{8}$$

$$x_{ijt}\in\{0,1\} \qquad \forall i\in N\backslash\{n+1\},\ j\in N\backslash\{0\},t\in M \tag{9}$$

The objective function (1) minimizes the sum of all job processing times plus the sum of all server travel times. The first term in the objective function is the summation of the processing times for all jobs; for a job i this is equal to a fixed processing time (a_i) plus its degradation rate α_i times the start time for job i. The second term is the total travel time of all servers; note that there is a constraint that limits the assignment of a job to one specific day and to one specific server. Constraint (2) states that each node (job) is served one time and during one day. Constraint (3) shows that in each day at most k servers are busy and ensures that all of them begin their tour from the depot. It also ensures that the number of servers used during each day is less than or equal to the number of available servers k. Constraint (4) ensures that servers return to the depot each day. Constraint (5) states that a server departs node h only if it enters that particular node. Constraint (6) ensures that we visit node j after finishing service of node i, if a server moves from node i to node j. Constraint (7) guarantees that in each day every server's route begins from its starting depot at time 0. Constraint (8) ensures that each job's starting time is during the time period from 0 to T hours. Constraint (8) for job $n+1$ implies that each server returns to its ending depot by time T hours. Finally, constraints (9) specify the binary decisions variables.

We note that in (Q) we do not have an index for servers. This is because the solution to (Q) provides us with the sequence of jobs for each server for each day. We chose not to have an index for servers as it greatly reduces computation time for solving (Q).

We further note that (Q) contains a product of binary and continuous variables in its objective function, which cannot be solved using the GUROBI optimization solver. To circumvent this issue, we introduce a

new variable $y_{ijt} = w_{it}x_{ijt}$ and add the following constraints instead of $y_{ijt} = w_{it}x_{ijt}$ for each i, j and t:

$$w_{it} - M(1 - x_{ijt}) \leq y_{ijt}$$
$$w_{it} \geq y_{ijt}$$
$$Mx_{ijt} \geq y_{ijt}$$

The resultant reformulation (P) is stated below and can be solved by the GUROBI optimization solver. Linearizing the model adds $3 * m * n^2$ constraints to the model. Since w_{it} is an upper bound for y_{ijt} and we know that maximum amount for w_{it} is equal to T, we let $M = T$.

$$(P)\text{:min} \sum_{i=0}^{n} \sum_{\substack{j=1 \\ j\neq i}}^{n+1} \sum_{t=1}^{m} a_i + \alpha_i y_{it} + 24(t-1)\alpha_i x_{ijt} + \sum_{i=0}^{n} \sum_{\substack{j=1 \\ j\neq i}}^{n+1} \sum_{t=1}^{m} \tau_{ij} x_{ijt} \tag{10}$$

$$\text{s.t.} \quad \sum_{t=1}^{m} \sum_{\substack{j=1 \\ j\neq i}}^{n+1} x_{ijt} = 1 \qquad \forall i \in N\backslash\{0,n+1\} \tag{11}$$

$$\sum_{j=1}^{n+1} x_{0jt} \leq k \qquad \forall t \in M \tag{12}$$

$$\sum_{i=0}^{n} x_{i(n+1)t} \leq k \qquad \forall t \in M \tag{13}$$

$$\sum_{i=0}^{n} x_{iht} - \sum_{\substack{j=1 \\ j\neq i}}^{n+1} x_{hjt} = 0 \qquad \forall h \in N\backslash\{0,n+1\},\ t \in M \tag{14}$$

$$(w_{it} + a_i + \alpha_i(w_{it} + 24(t-1)) + \tau_{ij} - w_{jt})x_{ijt} \leq 0 \qquad \forall (i,j) \in A,\ t \in M \tag{15}$$

$$w_{it} - M(1 - x_{ijt}) \leq y_{ijt} \qquad \forall t \in M,\ i,j \in N \tag{16}$$

$$w_{it} \geq y_{ijt} \qquad \forall t \in M,\ i,j \in N \tag{17}$$

$$Mx_{ijt} \geq y_{ijt} \qquad \forall t \in M,\ i,j \in N \tag{18}$$

$$w_{0t} = 0 \qquad \forall t \in M \tag{19}$$

$$0 \leq w_{it} \leq T \qquad \forall i \in N\backslash\{0\},\ t \in M \tag{20}$$

$$x_{ijt} \in \{0,1\} \qquad \forall i \in N\backslash\{n+1\},\ j \in N\backslash\{0\}, t \in M \tag{21}$$

4.2 Properties

Intuitively, jobs with high degradation rate should be done sooner, as (P) seeks to minimize the sum of job processing times as one component of its objective. Also, a job with less travel time to reach it should be done earlier, as this assures an earlier start to jobs and hence less processing time. So if there is a match between the degradation rate ordering and the travel time ordering we have an optimal solution to (P). This condition is rare but provides a useful basis for our enhanced greedy heuristic presented in Section 5. We now establish that a sequence of the jobs according to a non-decreasing order of α_i and $\min_i\{\tau_{J_{i-1}J_i}\}$, where $\tau_{\cdot i}$ denotes the travel time between job i and its previous job in the route, is optimal.

Proposition: Suppose that there exists a sequence of jobs $J_1, J_2, ..., J_n$ such that $\alpha_1 \geq \alpha_2 \geq ... \geq \alpha_n$ and $\tau_{01} = \min_i\{\tau_{0i}\}$, $\tau_{12} = \min_i\{\tau_{1i}\}$,...,$\tau_{(n-1)n} = \min_i\{\tau_{(n-1)i}\}$. This sequence is optimal to (P) for the case $k = 1$.

Proof. We use a pairwise job interchange argument. Let S and S' be two job schedules where the difference between them is the pairwise interchange of two adjacent jobs J_i and J_j that is, $S = (\pi, J_j, J_i, \pi')$ and $S'=(\pi, J_i, J_j, \pi')$, where π and π' are partial sequences. Furthermore, we assume that the last job in π is job J_x and the first job in π' is J_k. We also let $\tau_{.i}$ denotes the travel time between job i and its previous job in the sequence. Let $C_j(S)$ presents the completion processing time of job J_j in route S . We suppose that S is the optimal solution in which $\alpha_j < \alpha_i$ and $\tau_{xj} < \tau_{xi}$. So, if we consider l as the last job in both S and S', it suffices to show that $C_l(S) \leq C_l(S')$.

$$C_i(S) = C_\pi + \tau_{xj} + a_j + \alpha_j(C\pi + \tau_{xj}) + \tau_{ji} + a_i + \alpha_i(C\pi + \tau_{xj} + \alpha_j(C\pi + \tau_{xj}) + \tau_{ji})$$

$$C_j(S') = C_\pi + \tau_{xi} + a_i + \alpha_i(C_\pi + \tau_{xi}) + \tau_{ij} + a_j + \alpha_j(C_\pi + \tau_{xi} + \alpha_i(C_\pi + \tau_{xi}) + \tau_{ij})$$

$$\begin{aligned} C_j(S') - C_i(S) &= \tau_{xi} - \tau_{xj} + C\pi(\alpha_i - \alpha_j) + \alpha_i\tau_{xi} - \alpha_j\tau_{xj} + C\pi(\alpha_j - \alpha_i) + \alpha_j\tau_{xi} - \alpha_i\tau_{xj} \\ &\quad + \alpha_i\alpha_j(\tau_{xi} - \tau_{xj}) + \alpha_j\tau_{ji} + \alpha_i\tau_{ji} \\ &= \tau_{xi} - \tau_{xj}[1 + \alpha_j + \alpha_i + \alpha_i\alpha_j] + (\alpha_j - \alpha_i)\tau_{ij} \\ &\geq 0 \end{aligned}$$

We know that in the undirected graph for any i and j, $\tau_{ij} = \tau_{ji}$. Consider k as the first scheduled job in π'. Then, we have:

$$C_k(S) = C_i(S) + \tau_{ik} + \alpha_k(C_i(S) + \tau_{ik})$$

$$C_k(S') = C_j(S') + \tau_{jk} + \alpha_k(C_j(S') + \tau_{jk})$$

$$\begin{aligned} C_k(S') - C_k(S) &= C_j(S') - C_i(S) + \tau_{jk} - \tau_{ik} + \alpha_k(C_j(S') - C_i(S)) + \alpha_k(\tau_{jk} - \tau_{ik}) \\ &= (1 + \alpha_k)(C_j(S') - C_i(S)) + (1 + \alpha_k)(\tau_{jk} - \tau_{ik}) \end{aligned}$$

We observe that $\alpha_k > 0$ and that we have shown that $C_j(S') > C_i(S)$. Therefore, if $\tau_{jk} > \tau_{ik}$ then we get $C_k(S') > C_k(S)$. In a similar way we have $C_l(S') > C_l(S)$. □

5 Heuristic solution procedures

The special case for (P) in which $\alpha = 0$ and $T = 24$ is a VRP, which is a known NP-hard problem. Thus we fully expect the general cases for (P) to be computationally challenging and focus on the development of a robust set of heuristics to solve larger problem instances. In this section, we present three heuristic algorithms for cases when GUROBI is unable to solve (P) (this turns out to be 14 jobs or larger according

to our experiments in Section 6.2).

Algorithm 1 Enhanced Greedy Heuristic

procedure CONSTRUCTIVE GREEDY
Inputs: A set of all the nodes that are going to be processed
Outputs: A set of routes for m days and k servers that do not violate the time limit constraint.
Step 1: Initialization
    Get the attributes related to each node, get the set of servers, k
    Initialize a set to store all nodes that are not visited, Φ,
    Initialize a set to store each visited node, Γ.
Step 2:
  while $\Phi \neq \emptyset$ **do**
  Make a new route for any server:
    for each $i \in k$ **do**
    $\{u, \alpha_u\} := \{\{u, \alpha_u\} : \alpha_u = max\{\alpha_w\}, \forall\{w, \alpha_w\} \in \Phi\}$
    Add new route 0-u-0 to the routed set, Γ and remove u from Φ
      for each $w \in \Phi$ **do**
      Find the Best Seed:
      $\{\nu, \alpha_\nu\} := \{\{\nu, \alpha_\nu\} : \alpha_u = max\{\alpha_w\}, \forall\{w, \alpha_w\} \in \Phi\}$
      $IU(w, I) = \tau_{Iw} + \tau_{wI+1} - \tau_{I(I+1)} + \sum_{i=I+1}^{n} \alpha_i \Delta T_i + \alpha_w T(w)$
        Find the Best Position:
      $\{J, IU(\nu, J)\} := \{\{J, IU(\nu, J) : IU(\nu, J) = min\{IU(I)\}, \forall\{I, IU(\nu, I)\} \in \Gamma\}$
      end for
    end for
  end while
end procedure

procedure LOCAL SEARCH
Inputs: Route I which is stored in set Γ as the output of previous procedure
Outputs: An improved route stored as Best.
  Best:=I
  while not Terminate() **do**
    Find a new neighborhood:
    Current:=LocalSearch(Best)
    if ObjValue(Current) < ObjValue(Best) **then**
      Best:=Current
    end if
  end while
end procedure

5.1 Enhanced greedy heuristic

The pseudo code for the enhanced greedy heuristic is provided in Algorithm 1. There are two stages of this algorithm, constructive greedy followed by local search.

In the constructive greedy stage, we implement a greedy constructive algorithm (motivated by the result in our proposition). The greedy constructive algorithm is based on the method to find an initial solution in Clarke and Wright (1964); it creates a route for each worker by sequentially assigning nodes (from those remaining) in decreasing α order and inserting it in a position that adds the minimum possible amount of time to the route. This algorithm does not start a new day until the servers cannot process additional jobs

in the current day due to the time constraint.

The second stage of the enhanced greedy heuristic, we attempt to improve the solution from the first stage by applying local search. The method that we used for local search is based on three neighborhood definitions, "remove", "exchange" and "2-opt". In "remove" we randomly select a node and remove from its current position and add it to another, randomly selected, position. In "exchange" we randomly select two nodes and exchange their position. In "2-opt" we select two different nodes and reverse the sequence of job completions between these two nodes. For producing new solutions or neighborhoods of a current solution, we use three movements, remove, exchange and 2-opt. We randomly select one of the three movements at each stage and search the new neighborhood. In remove movement, the algorithm randomly selects a node, removes it from its current position and adds it in a randomly selected new position. In an exchange move, two nodes are randomly selected and change places. In the 2-opt move, the edges $(i, i+1)$ and $(j, j+1)$ are replaced by edges (i, j) and $(i+1, j+1)$, causing reversal in the direction of edges between $i+1$ and j.

5.2 Chronological decomposition

This heuristic simplifies our (P) problem. The concept behind this algorithm is as follows: first, we prioritize the jobs and assign them to each day, then for any day we use a suitable algorithm (either an exact MIP model or an enhanced greedy heuristic) to assign jobs to the servers considering the time limits. Let Φ be the set of n jobs that need to be processed in m days and M be the set of days. We consider the first degradation model mentioned in the literature review to approximate the degradation effect in our model, $\rho_j = P_j + \alpha_j w_j$.

We let $f_{\alpha_i}(x)$ represents the processing time function of job i at time w_i and define it as $f_{\alpha_i}(w_i) = P_i + \alpha_i w_i$. Since we don't know the exact time in the day when a job will start to be processed we assume that it starts in the middle of the day, i.e. at time $t = T/2$. Therefore the approximation of the processing time of job i in day t that we use is $D_{it} = f_{\alpha_i}(T/2 + 24(t-1))$, in which t is the index and T is the length of the day. Since jobs degrade over time we need an approximation of how many jobs can be processed during each day. To achieve this goal we examine the solution provided by the enhanced greedy method and use the number of jobs processed in each day as an estimation. This provides the parameter values N_t in the model. We define a binary variable y_{it} which is 1 if job i is processed during day t and 0 otherwise. Armed with this notation, we formulate an MIP model (CD) to assign jobs to days as follows:

$$\text{(CD):min} \sum_{i=1}^{n} \sum_{t=1}^{m} D_{it} y_{it} \tag{22}$$

$$\text{s.t.} \quad \sum_{t=1}^{m} y_{it} = 1 \qquad \forall i \in \Phi \tag{23}$$

$$\sum_{i=1}^{n} y_{it} \leq N_t \qquad \forall t \in M \tag{24}$$

$$y_{it} \in \{0, 1\} \qquad \forall i \in \Phi, t \in M \tag{25}$$

The solution of (CD) yields a set of jobs to be done for each day. We now run single day versions of (P) separately for each day to determine the assignment and sequence of jobs to servers, for each day. The advantage of this model is that it divides a large problem into a number of smaller problems that are each

solvable in a reasonable amount of time.

We note that the chronological decomposition heuristic simply requires the processing time of a job as a specified function of job start time. This function can be linear or non-linear. To demonstrate the capability of the chronological decomposition method for a non-linear processing time situation, we revisit our earlier example with non-linear processing times.

Table 3: Data for case of non-linear service time for situation with 10 jobs and 4 days

Jobs	1	2	3	4	5	6	7	8	9	10
Service time on day 1	2.1	2.3	2	2.5	2.1	1.5	2.7	1.3	2	1.9
Service time on day 2	2.7	2.4	2.1	2.2	3.1	3.0	2.8	2.3	2	1.9
Service time on day 3	2.0	2.5	2	2.5	3.1	2.5	2.8	2.3	2.2	1.5
Service time on day 4	2.2	2.1	2.2	2.7	2.7	1.7	2.7	1.0	2	1.8
Coordinate	(6,6)	(7,4)	(6,3)	(5,1)	(3,2)	(1,2)	(2,3)	(2,4)	(3,6)	(4,5)

Recall the motivating numerical example that we used to illustrate the features of our model in Section 3, The same jobs with non-linear processing time during four consecutive days are shown in Table 3. For instance, repairing job 1 in day 1 takes 2.1 hours but it takes 2.7 hours in second day, and as it is clear, the processing time is not ascending and not linear. One server with a fixed speed of 2 mile per hour serves the jobs. Using the chronological decomposition method, the optimal route is $O = [[0-7-6-5-0],[0-4-0],[0-1-3-10-0],[0-2-9-8-0]]$. The chronological decomposition method tries to schedule each job on the day where its processing time is smaller. For instance, jobs 7, 6 and, 5 have their minimum processing times on day 1, as 2.7, 1.5, 2.1, respectively, and job 4 has its minimum processing time on day 2.

5.3 Simulated annealing

Simulated annealing (SA) was proposed first time by Kirkpatrick et al. (1983). The method models the physical process of heating a material and decreasing the temperature slowly to decrease defects, thus minimizing the system energy. The algorithm accepts all new points that lower the objective, but also, with a certain probability, points that raise the objective. By accepting points that raise the objective, the algorithm avoids being trapped in local minima, and is able to explore globally for more possible solutions.

We found that the enhanced greedy was much better than SA when applied to an initial solution. However, we also found that if SA was provided with an excellent starting solution like the one obtained after the enhanced greedy approach, it was able to, in many instances, improve it further. Thus our suggested method is a composite heuristic, in which we apply SA to an initial solution produced by the enhanced greedy algorithm.

The SA algorithm we implemented had 10 phases, each with 200 iterations. The first phase started with an initial temperature of 9, and each subsequent phase reduced the temperature by a factor of 0.95.

6 Computational testing

We organize our computational results into four parts. In the first part we establish limits of the exact method. The second part compares the three different heuristics we have developed on small instances that can be solved optimally using the exact method, and use these to select a preferred heuristic approach. The third part shows results of our preferred heuristic approach on large problem instances, both in terms of objective function value and in terms of determining a lower bound. The fourth part studies the effect of key factors that provide input to the problem (P), using a factorial design method. The solver we use is GUROBI 8 (64 bit), which is programmed in Python 3.6 on a computer running Windows 10 Enterprise (64bit) with a processor Intel(R)Core(TM) i7-6500U CPU @ 3.50GHz and 7.88GB usable RAM. Prior to delving into our analysis, we would like to point to the fact that there exist many algorithms and approaches that are complementary to our selected methods–in particular, the reader is referred to papers by Riazi et al. (2017) and Pour et al. (2018).

6.1 Generation of instances

We started by simulating data for 200 nodes, where each node represents a job. Each node has X and Y coordinates and the length of the arc between two nodes is given by the Euclidean distance between them. X and Y belong to a uniform distribution $(0, 20)$. A speed parameter is used to convert distances to travel time. A fixed service time with average of 0.25 and a degradation rate with an average 0.005 is assigned to each node. Including the depot, we have a complete graph (there are links between all pairs of nodes) for a graph with 201 nodes. Instances of various sizes were created by selecting portions of this complete graph. The repository of the data sets we used is our testing is contained in Aarabi and Batta (2019).

6.2 Limits of the exact method

In order to maximize GUROBI performance we use a tuning parameter tool along with an initial solution obtained by our simulated annealing (SA) heuristic. The GUROBI optimizer provides a wide range of available parameter settings to permit users to control the operation of its optimization engine. We used automatic tuning to find the best parameter setting useful for a variety of instances. The best setting we found is $m.params.cutpasses = 5$ and it resulted in an improved computation time. This parameter controls the maximum number of cutting plane passes performed during root cut generation.

Table 4 presents the results of our "standard" exact algorithm as well as the results for the exact algorithm which uses tuning aided with the initial solution supplied by the enhanced greedy heuristic. We note that the initial solution is also referred to in the literature as a warm-start, for more information please see Marcucci and Tedrake (2019). The first column indicates the number of nodes for each instance. The second column is the CPU time required to run the exact model (using standard settings with no initial solution provided). The third column shows the CPU running time for the exact model using tuning parameters provided with the initial solution for the enhanced greedy heuristic. As expected, the time decreases when using parameter tuning. However, from Table 4 it is evident that even by using tuning and a good initial solution the time it takes to solve an instance with 14 nodes is 1355 seconds and beyond this the solver cannot deliver a solution within 3600 seconds. This makes exploration of heuristics inevitable.

Table 4: Comparison between run-time for Exact algorithm when using GUROBI and GUROBI aided by tuning and providing an initial solution

	Exact	Exact(Tuning + Initial Sol.)
No.	CPU time*	CPU time*
4	0.01	0.17
5	0.02	0.22
6	0.28	0.25
7	0.32	0.30
8	0.86	0.56
9	1.11	1.11
10	3.10	3.04
11	15.17	7.94
12	15.46	13.53
13	142.77	80.20
14	3702.05	1355.03
15	3600.09	2457.93
16	3600.48	3600.51
17	3600.21	3600.25
18	3600.20	3600.41
19	3600.39	3600.41
20	3600.16	3600.27

(*) Time in seconds

Table 5: Comparison between optimal and heuristics solution for small instances

	Exact	Enhanced Greedy			Chronological Decomposition			SA		
No.	Obj.	Obj.	CPU time*	Gap**	Obj.	CPU time*	Gap**	Obj.	CPU time*	Gap**
4	0.182	0.182	0.014	0.000	0.182	8.680	0.000	0.182	8.176	0.000
5	0.229	0.230	0.003	0.003	0.229	8.323	0.000	0.235	8.136	0.027
6	0.273	0.274	0.003	0.001	0.273	9.849	0.000	0.287	8.116	0.048
7	0.303	0.303	0.003	0.000	0.303	8.467	0.000	0.334	8.139	0.105
8	0.351	0.351	0.003	0.000	0.351	8.950	0.000	0.362	8.370	0.031
9	0.381	0.382	0.004	0.003	0.382	8.215	0.003	0.403	8.158	0.057
10	0.403	0.405	0.202	0.005	0.405	8.304	0.005	0.448	15.715	0.112
11	0.409	0.413	0.006	0.009	0.413	17.682	0.009	0.448	8.295	0.094
12	0.415	0.419	0.006	0.009	0.419	20.059	0.009	0.463	8.138	0.115
13	0.465	0.468	0.019	0.008	0.468	16.224	0.008	0.523	8.279	0.126
14	0.515	0.519	0.007	0.008	0.520	17.913	0.009	0.586	8.001	0.138

(*) Time in seconds
(**) Output in percent

6.3 Comparison between heuristics for small problem instances

We start by running all three heuristics on small problem instances for which the exact solutions are known. As it was shown in Table 4, one day, single server instances where the number of nodes is less than 14 can be solved optimally. We therefore solved (P) first for a set of small single day, single server instances that had between 4 and 14 nodes using the three different heuristics we developed (enhanced greedy, chronological decomposition and SA) and compared the results in Table 5. The first column is the number of nodes in each instance, and the second, third and fourth columns, respectively, indicate the objective function value, running time and gap, for the three different heuristics. From Table 5, one can see that the gap between the exact objective function value and the enhanced greedy algorithm is less than that with the SA solution.

Based on the above results, our preferred heuristic is to first apply the enhanced greedy method, and further improve the solution generated using the SA method. We test this preferred method on larger instances.

Table 6: Test results for large instances

$N = 50$, $T = 8$		Enhanced Greedy		Enhanced Greedy followed by SA						
Number of servers	Number of days	Objective function	CPU time*	Objective function	Routing value	Processing value	CPU time*	Improvement by SA**	LB	Gap**
$k = 1$	2	14.59	0.02	14.48	0.20	14.28	7.01	0.74	14.33	1.067
$k = 2$	1	13.36	0.03	13.16	0.20	12.96	7.05	1.55	13.09	0.547
$k = 3$	1	13.10	0.05	12.98	0.18	12.79	6.26	0.96	12.91	0.598
$k = 4$	1	13.02	0.03	12.93	0.23	12.70	6.36	0.7	12.83	0.792
$N = 75$, $T = 8$		Enhanced Greedy		Enhanced Greedy followed by SA						
Number of servers	Number of days	Objective function	CPU time*	Objective function	Routing value	Processing value	CPU time*	Improvement by SA**	LB	Gap**
$k = 1$	3	23.37	0.08	22.40	0.34	22.06	10.92	4.16	22.10	1.359
$k = 2$	2	19.00	0.04	18.68	0.33	18.35	10.26	1.68	18.55	0.686
$k = 3$	1	18.38	0.04	18.17	0.31	17.86	10.40	1.18	18.04	0.747
$k = 4$	1	18.23	0.06	18.03	0.34	17.69	10.83	1.15	17.87	0.890
$N = 100$, $T = 8$		Enhanced Greedy		Enhanced Greedy followed by SA						
Number of servers	Number of days	Objective function	CPU time*	Objective function	Routing value	Processing value	CPU time*	Improvement by SA**	LB	Gap**
$k = 1$	5	35.75	0.06	31.84	0.54	31.30	11.98	10.96	31.77	0.224
$k = 2$	2	25.41	0.06	24.92	0.40	24.52	11.61	1.91	24.82	0.399
$k = 3$	2	23.84	0.045	23.37	0.35	23.01	7.69	1.99	23.27	0.429
$k = 4$	1	23.33	0.33	23.08	0.36	22.72	9.60	1.10	23.06	0.108
$N = 125$, $T = 8$		Enhanced Greedy		Enhanced Greedy followed by SA						
Number of servers	Number of days	Objective function	CPU time*	Objective function	Routing value	Processing value	CPU time*	Improvement by SA**	Best LB	Gap**
$k = 1$	7	55.17	0.08	48.96	0.70	48.26	13.80	11.26	48.89	0.142
$k = 2$	3	35.48	0.10	35.07	0.54	34.52	14.05	1.17	34.92	0.425
$k = 3$	2	32.40	0.06	31.90	0.55	31.35	44.14	1.54	31.84	0.199
$k = 4$	1	31.38	0.05	30.88	0.48	30.40	14.85	1.60	30.85	0.090
$N = 150$, $T = 8$		Enhanced Greedy followed by SA								
Number of servers	Number of days	Objective function	CPU time*	Objective function	Routing value	Processing value	CPU time*	Improvement by SA**	LB	Gap**
$k = 1$	15	86.6	0.07	79.37	1.11	78.27	12.42	8.34	79.37	0.000
$k = 2$	4	48.01	0.09	46.67	0.78	45.89	10.46	2.80	46.66	0.019
$k = 3$	3	41.95	0.14	41.38	0.68	40.70	20.32	1.34	41.35	0.080
$k = 4$	2	39.29	0.13	38.67	0.59	38.09	17.34	1.34	38.65	0.054

(*) Time in seconds
(**) Output in percent

6.4 Solving large problem instances

6.4.1 Solution time

We use the preferred heuristic (enhanced greedy followed by SA) on a set of large problem instances. Our results in Table 6 show that we can solve problems with up to 150 jobs, 15 days and 4 servers within 44 seconds of computation time. This implies that our preferred heuristic method is capable of delivering solution within reasonable time, which is a key factor if we need to repeatedly re-optimize our problem as more jobs arrive. We also have broken down the objective function value into the "routing" component and the "processing" component for the instances solved. It is apparent that the majority of time is spent in repairing the jobs.

6.4.2 Lower bound

To establish a lower bound we use the solution obtained by our preferred heuristic as an initial solution to the solver and let it try to improve it as well as generated a good lower bound for an extended period of time (24 hours for instances with 1 server, and 10 hours for instances with 2, 3 or 4 servers).

6.5 Impact of key model parameters

In order to understand the impact of key parameter values on the objective function we perform a factorial design experiment, which studies one-way, two-way and three-way effects. We are interested in finding the impact of three parameters of our model. Therefore, we run a 3^3 full factorial design. Here l^k represents a factorial design in which there are k factors with each one having l levels. We considered the following three factors for our design: degradation rate (factor A) with three levels, low, medium (3 times low) and high (6 times low); distance between jobs (factor B) with three levels, short, medium (1.5 times short) and long (3 times short); and the number of hours that each worker can work during a day (factor C) with three levels, 8, 12 and 16 hours. The response in our problem is the objective function value obtained using the enhanced greedy algorithm. For each factor combination, 5 replications are used so as to gain statistical significance. For the above experiments we consider two servers.

Table 7: ANOVA table for factorial design

Source	DF	Adj SS	Adj MS	F-Value	P-Value
Model	26	5102734	196259	3.67	0.000
Linear	6	3290175	548362	10.25	0.000
Work hour	2	1046893	523446	9.78	0.000
Degradation rate	2	2042440	1021220	19.08	0.000
Distance	2	200842	100421	1.88	0.158
2-Way Interactions	12	1751212	145934	2.73	0.003
Work hour*Degradation rate	4	1556100	389025	7.27	0.000
Work hour*Distance	4	54319	13580	0.25	0.907
Degradation rate*Distance	4	140793	35198	0.66	0.623
3-Way Interactions	8	61347	7668	0.14	0.997
Degradation Rate*Distance*Work Hour	8	61347	7668	0.14	0.997
Error	108	5779318	53512		
Total	134	10882052			

Table 8 shows the treatment combinations and the results obtained and Figure 2 shows the iteration plot for the mean of the response variable. From both Table 7 and Figure 2 we conclude the following:

From the results shown in Table 7 we conclude that factors A (degradation) and C (work hours) have the most significant effect on the response variable, as now argued. P_Value related to both factors A and C are 0, which means that for any confidence level $(1-\alpha)$, degradation rate and work hour have significant effects on the objective function value. The next important factor is B (distance). For factor B, $P\_value = 0.158$. In other words, for confidence levels less than 84.2% the effect of factor B is significant.

Figure 2 shows the main effect plot for response. One can see that by a reduction in factor C the objective function value decreases. This is due to the fact that in longer repair periods, workers have more time during a day and nodes can be visited before the degradation effect significantly increases processing time. Also it is evident that by increasing factor A (degradation rate) the response variable increases. We can see that by applying the same change in larger amount of degradation rate, the change in objective function value is more severe. It is also evident that increasing the distances between nodes increases the objective function value.

No line crosses the right plot in Figure 2, implying no interaction between factors. In Table 8 the average

Table 8: Numerical results from factorial design experiments

	Factors			Result
Run	Work hours (T)	Degradation rate	Distance	Objective
1	8	Low	Short	476
2	8	Low	Medium	581
3	8	Low	Long	711
4	8	Medium	Short	541
5	8	Medium	Medium	686
6	8	Medium	Long	866
7	8	High	Short	2640
8	8	High	Medium	3541
9	8	High	Long	4422
10	12	Low	Short	440
11	12	Low	Medium	508
12	12	Low	Long	589
13	12	Medium	Short	475
14	12	Medium	Medium	559
15	12	Medium	Long	664
16	12	High	Short	842
17	12	High	Medium	1159
18	12	High	Long	1847
19	16	Low	Short	422
20	16	Low	Medium	476
21	16	Low	Long	541
22	16	Medium	Short	449
23	16	Medium	Medium	513
24	16	Medium	Long	590
25	16	High	Short	631
26	16	High	Medium	761
27	16	High	Long	930

of all 5 replications for each combinations of factor levels is presented. The minimum objective function value is obtained by run 19, which has settings of maximum level of work hours and minimum levels for degradation rate and distance.

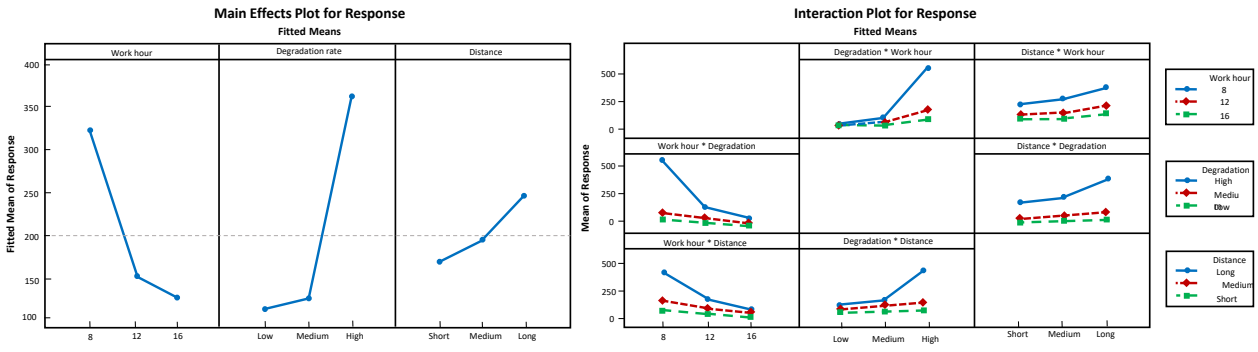

Figure 2: Main and interaction effect plots from factorial design experiments

7 Pothole repair case study

7.1 Context

Potholes are an excellent example of spatially dispersed degrading jobs. Potholes occur in many different portions of the transportation network. Potholes are the result of water expansion and contraction during the winter. Time and environmental conditions degrade the chemicals that bind pavement and cause the cracks Obaidi et al. (2017). Water penetrates through the pavement cracks, freezes and expands. The expansion of frozen water leads to the loss of the adhesion between aggregate, then vehicular loading over the weak pavement forms potholes. Through time the size of potholes grow and the time and effort that is needed to repair them increases as a result of this growth.

The importance of studying scheduling of pothole repair stems from the fact that potholes in the roads have many consequences. Hitting potholes can damage cars and in the worst case cause an accident. These consequences make pothole repair urgent. A further motivation for our study of pothole repair is the fact that the community our university is housed in, Buffalo, New York, is a classic pothole prone area because it receives an average of 93 inches of snowfall in the winter, and a typical winter day has daytime high temperatures above freezing and night time low temperatures below freezing. These constant melting and freezing cycles rapidly create potholes and their size increases as the winter progresses, leaving significant work for the road repair crews at the end of the winter season.

Pothole repair usually is done in two ways, as an emergency repair or as planned repair. Emergency repair is done during winter time. These emergency repairs are not the focus of our study. Planned repairs are done during drier and warmer periods of time, i.e., during the 3 months of Spring.

7.2 Goals

The goal in pothole repair is to fix all of the potholes as quickly as possible, while maintaining equity between different neighborhoods in terms of pothole repair and while considering new potholes that develop during the spring season (these are small cracks which develop into potholes).

We study this goal in steps. We first consider the case where potholes are all available for scheduling at the beginning of the time horizon and no new potholes arise. We directly apply (P) to this situation after creation of an appropriate data set. This allows us to study the impact of the number of servers required, the impact of degradation rate, and the impact of the central depot location.

The second step is to model equity of pothole repair among sub-regions. Since potholes are a major nuisance, communities are sensitive to equal treatment from the government agency responsible for fixing potholes. Here we study the cost impact of introducing an equity constraint to the model.

The third step is to model continuously arriving jobs, which is important since at the beginning of the repair season there are many road cracks that later develop into potholes as the repair season evolves. Here is the goal is to study the impact of continuously arriving jobs on number of servers required.

The fourth step is to isolate the effects of degradation, spatial distribution of jobs and equity, and study

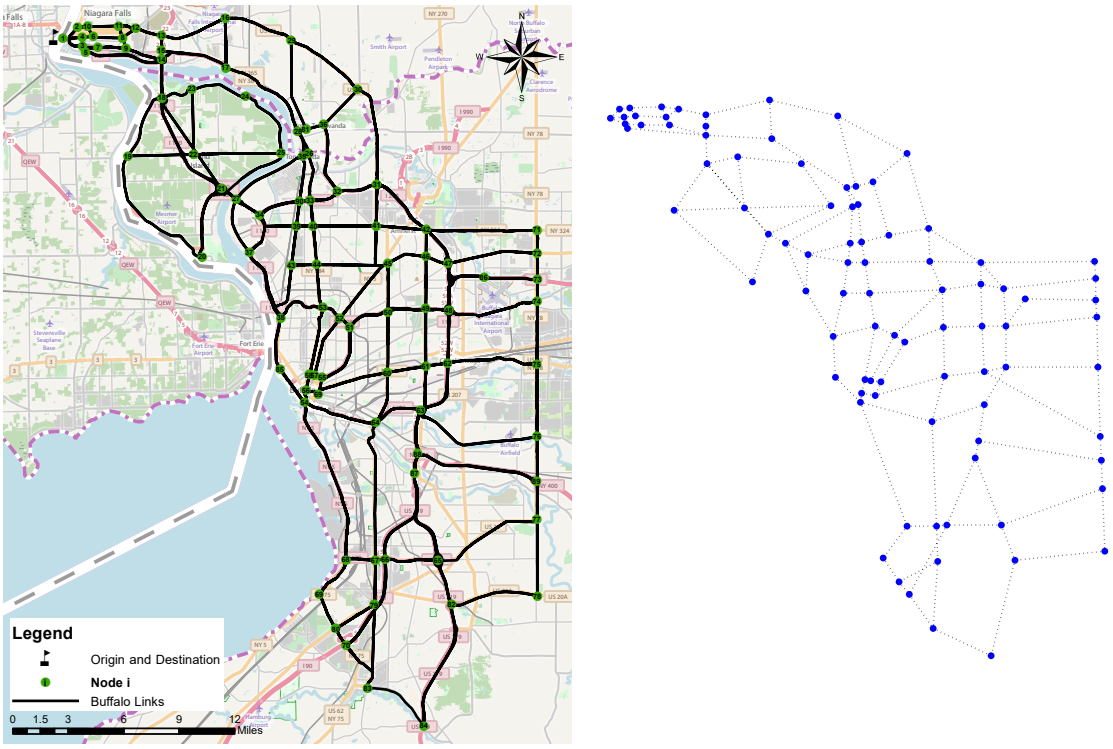

Figure 3: Buffalo, New York road network used for case study

the impact of each of these individually.

Overall, we believe that our methods can form the basis of a planning tool for scheduling pothole repair.

7.3 Data used for generation of problem instances

To illustrate the findings of this research we conducted our experiments on a portion of the road network of the City of Buffalo, New York, used in Toumazis and Kwon (2013). The network contains of 90 nodes and 149 arcs in which node 0 is considered as the depot. Each arc of the network has two attributes: Annual Average Daily Traffic (AADT) and free flow travel time. Our model needs initial repair time and degradation rate related to each arc of the network. We now specify the method we used to compute these parameters.

7.3.1 Computation of initial repair time

We postulate that the number of potholes on each arc, N, follows a Poisson distribution ($N \sim Poisson(\lambda)$) with parameter λ. To estimate the initial repair time associated with fixing all of the potholes on any arc, which we consider as a single job in our model, we consider pothole repair as an $M/M/1$ queuing system. We let λ denote the average arrival rate of potholes on any arc and μ represent the average initial repair time. Larson and Odoni (1981) show that the following relationship applies to λ and μ in an $M/M/1$ system:

$\bar{w} = 1/\mu + \lambda/\mu(\mu - \lambda) = 1/(\mu - \lambda)$
$\mu = 1/\bar{w} + \lambda$

By assuming average waiting time as 24 hours, and assuming average λ as 0.7, we can use the Poisson distribution to find the average μ. Therefore, we generate random numbers following a Poisson distribution with mean λ and assign repair time to the links of the network as μ.

7.3.2 Computation of degradation rate

Degradation Jain et al. (2005) show in their study that there is a relationship between degradation rate and AADT. We therefore assume that the degradation rate of an arc has a linear relationship with its AADT. From a practical point of view this makes sense, since the higher the traffic volume on a road segment, the greater the increase in damage to existing potholes as well as the greater the chances of new potholes. Thus degradation rate is different for each arc and is based on traffic volume.

7.3.3 Graph conversion

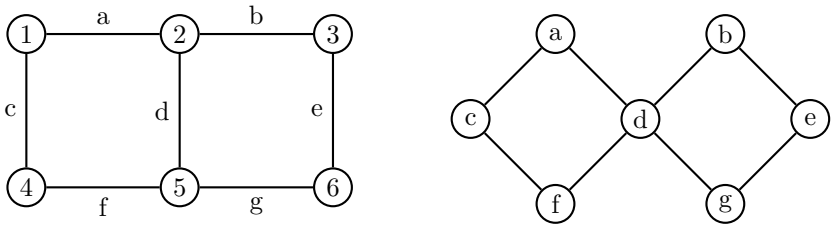

Figure 4: An example of graph conversion. left:graph G, right:graph H

Since potholes occur on arcs and all potholes on an arc are modeled as a single job, we convert the transportation network to its dual graph in which each arc becomes a node and in which each node becomes an arc. As it is shown in Figure 4, graph G is converted to graph H. We assign attributes on links in graph G to obtain that of each node in graph H, for instance, travel time of link $c-a$ in graph H, named t_{ca}, is equal to $(t_a + t_c)/2$ on graph G. Graph G of our case study has 90 nodes and 149 links, therefore the converted graph, graph H, has 149 nodes and 374 links. We use the converted network for the City of Buffalo with a time period of length 12 hours ($T = 12$) and the number of servers as $k = 1, 2, 3$.

7.4 Results for base model

We now present our numerical results (using the enhanced greedy heuristic followed by SA) for the base model (P), for cases with one, two and three servers, with the depot being near node 1, which is in the upper left hand corner of the study region. We would like to point out that throughout the case study we have used enhanced greedy followed by SA as the solution method, since this was the solution of choice for large problem instances in section 5. As one can see from Table 9, by using one server it is possible to fix all potholes in 36 days. By increasing the number of servers to two, the total repair can be accomplished in 10 days. The reason that the overall task decreases by 3 fold when two servers are used is due to the effect of job degradation. When using a single server, jobs are delayed and when they finally are ready to be processed they take much longer to do, causing a dramatic increase in overall task time. By further increasing the number of servers to three, the total repair can be accomplished in 6 days. As expected, the decrease in number of days required is higher when we go from 1 to 2 workers, as opposed to when we go from 2 to 3 workers (i.e. there is a decreasing marginal benefit when increasing the number of workers). In the second row of Table 9, the degradation rate is increased by a factor of 2. Now, the total repair times are 110 days

(1 server), 17 days (2 servers), and 8 days (3 servers). Thus the degradation rate is a determining factor for the number of servers we should use.

To study the impact of the depot location we moved the depot to a central location on the link that joins nodes 50 and 51. Results for this case are shown in Table 10. The findings for the impact of the number of servers and the degradation rate are similar to the earlier case where the depot was in the upper left hand corner of the study region. However, we note that in all cases the objective function value is smaller due to the more favorable location of the depot.

Table 9: Numerical results for case study with one, two and three servers (Node 0 is considered as depot)

$\alpha = Low$			
Number of workers	Number of days	Objective function	CPU time*
1	36	423.171	0.188
2	10	222.025	0.198
3	6	190.739	0.136
$\alpha = High$			
Number of workers	Number of days	Objective function	CPU time*
1	110	2471.336	0.278
2	17	400.062	0.208
3	8	252.035	0.139

(*) Time in seconds

Table 10: Numerical results for case study with one, two and three servers (a middle link, 50-51, is considered as depot)

$\alpha = Low$			
Number of workers	Number of days	Objective function	CPU time*
1	31	360.945	0.148
2	9	211.027	0.151
3	6	182.571	0.136
$\alpha = High$			
Number of workers	Number of days	Objective function	CPU time*
1	104	1947.218	0.151
2	15	337.763	0.136
3	7	234.329	0.141

(*) Time in seconds

7.5 Incorporation of equity considerations

We now extend the base model (P) to incorporate an equity constraint. According to the National Academy of Public Administration, public equity is defined as "The fair, just and equitable management of all institutions serving the public directly or by contract; the fair, just and equitable distribution of public services and implementation of public policy; and the commitment to promote fairness, justice, and equity in the formation of public policy" National Academy of Public Administration (2009). In the context of pothole repair we model equity between different sub-regions. The performance measure for a sub-region is the average

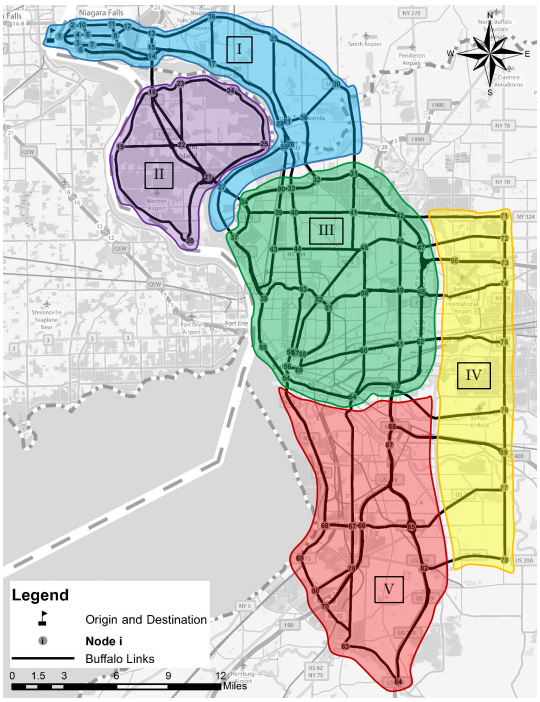

Figure 5: Division of the case study region into sub-regions to study impact of equity considerations

completion time of jobs in that sub-region. We want the average job completion time to be equitable between sub-regions. Therefore we need our model to consider a limit that ensures no significant difference between the average job completion times in between pairs of sub-regions. To formalize the introduction of equity in (P), we define N_i as a set of nodes belonging to sub-region i. We consider δ (equity level) as the maximum allowed difference between the average job completion times of two sub-regions. We can now add the following constraints for any pair (A, B) of sub-regions to our model. We note that the units of the parameter δ is time.

$$\sum_{i\in N_A}\sum_{\substack{j\in N_A\\ j\neq i}}\sum_{t\in M}(24(t-1)+w_{it}+a_i+\alpha_i(w_{it}+24(t-1))x_{ijt}-\sum_{i\in N_A}\sum_{\substack{j\in N_A\\ j\neq i}}\sum_{t\in M}(24(t-1)+w_{it}+a_i+\alpha_i(w_{it}+24(t-1))x_{ijt}\leq\delta \tag{26}$$

$$\sum_{i\in N_B}\sum_{\substack{j\in N_B\\ j\neq i}}\sum_{t\in M}(24(t-1)+w_{it}+a_i+\alpha_i(w_{it}+24(t-1))x_{ijt}-\sum_{i\in N_B}\sum_{\substack{j\in N_B\\ j\neq i}}\sum_{t\in M}(24(t-1)+w_{it}+a_i+\alpha_i(w_{it}+24(t-1))x_{ijt}\geq\delta \tag{27}$$

Table 11: Numerical results for the case study with one and two servers after applying an equity constraint (δ=30)

$\alpha = Low$		Enhanced Greedy followed by SA-before equity	Enhanced Greedy followed by SA-after equity
Number of workers	Number of days	Objective function value	Objective function value
1	36	423	966
2	10	222	474
$\alpha = High$		Enhanced Greedy followed by SA-before equity	Enhanced Greedy followed by SA-after equity
Number of workers	Number of days	Objective function value	Objective function value
1	110	2471	2504
2	17	400	896

We now revisit our case study for the City of Buffalo to model equity in five sub-regions, as shown in Figure 5. In Table 11 we study the cost of adding the equity constraint, in terms of the increase in the objective function value of (P). In practical terms, we want to select an equity level that yields a solution which causes no more than a 10% increase in objective function value. As one expects, average completion time increases by adding equity constraint even though the difference between average job completion times of sub-regions decreases. One way to improve the average job completion time is to increase the number of servers, and a practical solution may be to increase the number of servers to reach solutions that address both the average completion time objective for the whole region as well as equity between average job completion times between sub-regions. Increasing the number of servers would also be needed for situations where a higher degradation rate is expected.

7.6 Case of continuously arriving jobs

The models (P) and (CD) assume a fixed number of jobs that are all available at the beginning of the planning period. In practice, additional potholes (jobs) continuously arrive over the repair horizon and must be incorporated in the schedules for the available servers through re-optimization. We use the well-studied rolling horizon method to do this. For example, if the length of our rolling horizon is 5 days and there are 30 days in our time horizon, our initial problem will take all of the initially available jobs and schedule these over the 30 day horizon. The first five days of this solution is implemented. The next problem to be solved is a 25 day problem with the remaining jobs from the original set that have not been processed and the new set of jobs that arrived during the first five days. This process continues till the 30 day horizon is complete. We note that there are implementations of the rolling horizon framework that are based on job arrivals (e.g. wait for five new jobs to arrive before solving the next problem). We selected the framework based on time for our case study demonstration, as the pothole repair schedule typically needs to be made public for a period of time. A key decision variable is the length of the rolling horizon. Determination of the optimal length requires an empirical investigation which we demonstrate in our case study.

In our case study, we assume that 50 jobs out of 149 are available at the beginning and the rest of them arrive during a period of 24 days. This is equivalent to assuming that there are 50 confirmed potholes at the beginning of the Spring repair season and the existence of numerous road cracks, 99 of which develop into potholes as the repair season progresses. We assume that the arrival of jobs (cracks that become potholes) follows a Poisson distribution with $\lambda = 5$. To find the optimal value of the rolling horizon length l, we run the

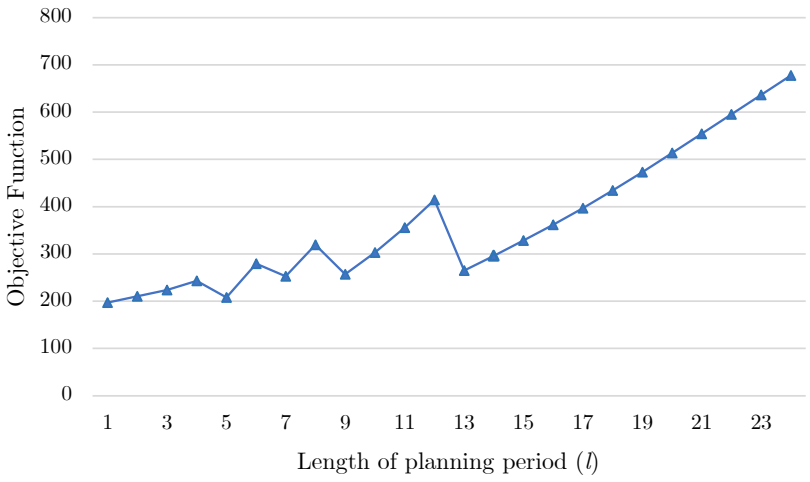

Figure 6: Impact of the length of the planning period (l)

rolling horizon method for different scheduling period lengths. For any specific length, we use the enhanced greedy followed by SA method to find the total objective function value. The objective function value of the rolling horizon method for different values of the lengths are shown in Table 12 and in Figure 6. For our situation, the best result occurs by setting l equal to 1.

In most realistic situations for pothole repair we cannot re-optimize every day as the repair schedule for a length of time (typically a week) has to be published and made known to citizens. Thus l cannot be less than a specified value L. When $L = 4$ we would select l as 5 since it is the minimum amount shown in Figure 6.

Table 12: Objective function values for different planning period lengths (l)

Planning period length (l)	1	2	3	4	5	6	7	8	9	10	11	12
Objective function	197	210	224	243	208	279	252	319	257	303	355	414
Planning period length (l)	13	14	15	16	17	18	19	20	21	22	23	24
Objective function	265	296	328	361	396	434	473	513	554	595	636	677

7.7 Isolation of the impact of degradation, spatial distribution and equity

There are two modeling aspects in the base model (P), one related to degradation rate and the other related to spatial distribution of jobs. We can consider both extreme cases to see their individual impacts. To study the impact of just spatial distribution of jobs we set the degradation rate for each job i, $\alpha_i = 0$. To study the impact of just degradation rate we increase α_i for each job in equal proportion by multiplying with a very large value so that the impact of distance is minimal. For both situations we obtain optimal solutions and then evaluate these optimal solutions using the original data for our case study (i.e. the real value of α_i).

The results of our numerical experiments for the case study of 149 nodes and 1 server are reported in Table 13. It is evident that considering degradation rate significantly reduces the objective function value. We also consider the case where jobs are not spatially distributed, i.e., *no-distance* case. This allows us to further evaluate the impact of the scheduling jobs with respect to the degradation rate. Therefore, we

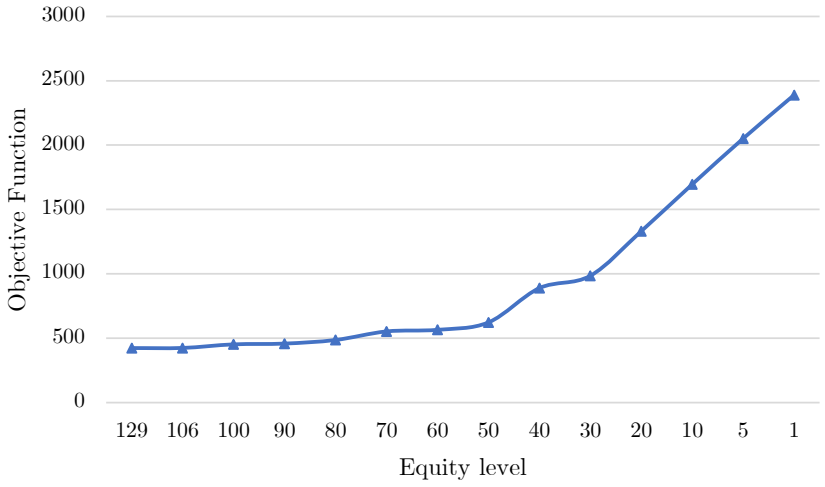

Figure 7: Impact of different levels of equity on the objective function

consider a large value for degradation rate so that it eliminates the effect of distance. We solve the case study of one server and an increased alpha by a factor of 200. The results are presented in Table 13.

Table 13: Numerical results for the *no-distance* case versus one including spatially distributed jobs

	Objective function	Objective function (no-degradation)	Change of objective function
$\alpha = 0$	423	597	41
Large α	423	2422	472

To study the impact of equity, we add the equity constraints (14) and (15) to the formulation (P) and solve using different equity levels. Results for the case study of 149 nodes and 1 server are shown in Figure 7. We can see that for an equity level (δ) more than 129 the equity constraint is redundant. By decreasing δ the value of objective function increases. In most practical cases, we can only increase the objective function value by 10% to accommodate equity considerations. Even this small increase in objective function value leads to a 30% increase in the equity level (δ). Another observation is that for smaller amounts of δ the model chooses to assign a small number of jobs to each day so as to control the equity between different sub-regions.

8 Summary, conclusions and future research

This paper considers scheduling spatially distributed jobs with degradation rate. Our fundamental assumption is that all jobs are available at the beginning and no new jobs arrive. We formulate this problem as an MIP for the linear degradation rate situation and are able to solve using GUROBI for situations with up to 14 jobs. To solve larger problems we develop three heuristic solution methodologies. Our first heuristic is an enhanced greedy algorithm that iteratively finds jobs with highest degradation rates and arranges them in routes for each server during any day, followed by a local search. Our second heuristic is a chronological decomposition, in which we use an MIP model to assign jobs to each day, and then develop each day's job schedule separately, using either an MIP exact approach or the enhanced greedy heuristic. Our third heuristic is simulated annealing, that uses various neighborhoods in a simulated annealing framework. To determine

the effectiveness of our heuristics, we carry out experiments with small size instances where the exact solution is known. From these experiments it became evident that the preferred solution method is to first use the enhanced greedy heuristic and then improve the resultant solution further by using the simulated annealing heuristic. Even though the chronological decomposition heuristic is not the preferred solution method, it is shown to be applicable to situations of non-linear degradation rate. To study the impact of key model parameters, we developed a factorial design experiment. Results show that the factors "degradation rate" and "work hours" have the largest impact on the objective function value. We also presented a case study for pothole repair in the City of Buffalo, New York, USA. In this case study we demonstrate a practical application of the model and study several variations to make it suitable to the pothole repair scenario. One of these variations included a situation where additional potholes arrive for repair after time zero. For this we use a rolling horizon framework. Another variation studies the impact of equity between sub-regions of the study area. We found that equity could be improved by 30% with a modest increase in objective function value of 10%. We further study the impact of degradation rate, spatial distribution of jobs, and equity separately to illustrate the importance of considering all three elements in one model.

Our conclusions are as follows:

- With respect to the linear degradation model, (i) exact solutions are limited to small instances with less than 14 jobs; (ii) the enhanced greedy heuristic followed by the application of the simulated annealing heuristic yields high quality solutions for large problem instances in reasonable computation time; and (iii) the factors "degradation rate" and "work hours" have a significant effect on the objective function.
- For the pothole repair application, consideration is needed for (i) the number of servers assigned, degradation rate of jobs and depot location, (ii) appropriate modeling of continuously arriving jobs, and (iii) appropriate incorporation of equity considerations.

Several directions for future research are possible. We mention two that we believe are highly relevant. The entire problem can be viewed as a queue control problem when jobs arrive continuously over time and theories from spatially distributed queues could potentially be applied. Also, the model may be helpful for other application settings, such as forest-fire control.

Acknowledgement

The authors are grateful to three anonymous referees for their insightful comments, which had led to more in-depth analysis and to a much clearer presentation of the material.

References

F. Aarabi and R. Batta. Data repository for artificial instances for scheduling spatially distributed jobs with degradation. *https://github.com/Faarabi/Scheduling-Network-Data*, 2019.

U. Bahalke, A. M. Yolmeh, and K. Shahanaghi. Meta-heuristics to solve single-machine scheduling problem with sequence-dependent setup time and deteriorating jobs. *The International Journal of Advanced Manufacturing Technology*, 50(5-8):749–759, 2010.

S. Balseiro, I. Loiseau, and J. Ramonet. An ant colony algorithm hybridized with insertion heuristics for the time dependent vehicle routing problem with time windows. *Computers & Operations Research*, 38(6): 954–966, 2011.

D. Biskup. Single-machine scheduling with learning considerations. *European Journal of Operational Research*, 115(1):173–178, 1999.

S. Browne and U. Yechiali. Scheduling deteriorating jobs on a single processor. *Operations Research*, 38(3): 495–498, 1990.

A. M. Caunhye, X. Nie, and S. Pokharel. Optimization models in emergency logistics: A literature review. *Socio-Economic Planning Sciences*, 46(1):4–13, 2012.

B. Chen, C. N. Potts, and G. J. Woeginger. A review of machine scheduling: Complexity, algorithms and approximability. In *Handbook of combinatorial optimization*, pages 1493–1641. Springer, 1998.

N. Christofides and S. Eilon. An algorithm for the vehicle-dispatching problem. *Journal of the Operational Research Society*, 20(3):309–318, 1969.

G. Clarke and J. W. Wright. Scheduling of vehicles from a central depot to a number of delivery points. *Operations Research*, 12(4):568–581, 1964.

J. Corstjens, B. Depaire, A. Caris, and K. Sorensen. A multilevel evaluation method for heuristics with an application of the vrptw. *International Transactions in Operational Research*, 27(1):168–196, 2020.

S. Dabia, S. Ropke, T. Van Woensel, and T. De Kok. Branch and price for the time-dependent vehicle routing problem with time windows. *Transportation Science*, 47(3):380–396, 2013.

A. V. Donati, R. Montemanni, N. Casagrande, A. E. Rizzoli, and L. M. Gambardella. Time dependent vehicle routing problem with a multi ant colony system. *European Journal of Operational Research*, 185 (3):1174–1191, 2008.

T. Eren. Human and machine effects in a just-in-time scheduling problem. *Human Factors and Ergonomics in Manufacturing & Service Industries*, 19(4):294–299, 2009.

S. Gawiejnowicz. A review of four decades of time-dependent scheduling: main results, new topics, and open problems. *Journal of Scheduling*, pages 1–45, 2020.

M. Gendreau, G. Ghiani, and E. Guerriero. Time-dependent routing problems: A review. *Computers & Operations Research*, 64:189–197, 2015.

J. N. Gupta and S. K. Gupta. Single facility scheduling with nonlinear processing times. *Computers & Industrial Engineering*, 14(4):387–393, 1988.

H. Hashimoto, M. Yagiura, and T. Ibaraki. An iterated local search algorithm for the time-dependent vehicle routing problem with time windows. *Discrete Optimization*, 5(2):434–456, 2008.

V. Huart, S. Perron, G. Caporossi, and C. Duhamel. A heuristic for the time-dependent vehicle routing problem with time windows. In *Computational Management Science*, pages 73–78. Springer, 2016.

T. Ibaraki, S. Imahori, M. Kubo, T. Masuda, T. Uno, and M. Yagiura. Effective local search algorithms for routing and scheduling problems with general time-window constraints. *Transportation Science*, 39(2): 206–232, 2005.

S. Ichoua, M. Gendreau, and J.-Y. Potvin. Vehicle dispatching with time-dependent travel times. *European Journal of Operational Research*, 144(2):379–396, 2003.

S. Jain, S. Aggarwal, and M. Parida. Hdm-4 pavement deterioration models for indian national highway network. *Journal of Transportation Engineering*, 131(8):623–631, 2005.

S. Jung and A. Haghani. Genetic algorithm for the time-dependent vehicle routing problem. *Transportation Research Record: Journal of the Transportation Research Board*, (1771):164–171, 2001.

S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. Optimization by simulated annealing. *Science*, 220(4598): 671–680, 1983.

K. Knight and J. Hofer. Vehicle scheduling with timed and connected calls: A case study. *Journal of the Operational Research Society*, 19(3):299–310, 1968.

A. S. Kunnathur and S. K. Gupta. Minimizing the makespan with late start penalties added to processing times in a single facility scheduling problem. *European Journal of Operational Research*, 47(1):56–64, 1990.

R. C. Larson. Decision models for emergency response planning. In *Handbook of Homeland Security*. Citeseer, 2005.

R. C. Larson and A. R. Odoni. *Urban Operations Research*. Prentice Hall, 1981.

J. K. Lenstra and D. B. Shmoys. Elements of scheduling. *arXiv preprint arXiv:2001.06005*, 2020.

Y.-H. Lin. *Delivery of critical items in a disaster relief operation: centralized and distributed supply strategies*. PhD Dissertation, Department of Industrial and Systems Engineering, University at Buffalo, 2010.

O. Madsen. Optimal scheduling of trucks-a routing problem with tight due times for delivery. *Optimization Applied to Transportation Systems*, pages 126–136, 1976.

C. Malandraki. Time dependent vehicle routing problems: Formulations, solution algorithms and computational experiments, retrieved from https://books.google.com/books?id=kx2osgaacaaj. 1989.

T. Marcucci and R. Tedrake. Warm start of mixed-integer programs for model predictive control of hybrid systems. *arXiv preprint arXiv:1910.08251*, 2019.

National Academy of Public Administration, 2009. https://www.napawash.org/aa_social_equity/index.html.

J. Nicholls, M. McHale, and I. Carswell. The selection of pothole repair options. In *Bituminous Mixtures and Pavements VII: Proceedings of the 7th International Conference'Bituminous Mixtures and Pavements'(7ICONFBMP), June 12-14, 2019, Thessaloniki, Greece*, page 425. CRC Press, 2019.

H. Obaidi, B. Gomez-Meijide, and A. Garcia. A fast pothole repair method using asphalt tiles and induction heating. *Construction and Building Materials*, 131:592–599, 2017.

S. Pour, J. Drake, L. Ejlertsen, K. Rasmussen, and E. Burke. A hybrid constraint programming/mixed integer programming framework for the preventive maintenance crew scheduling problem. *European Journal of Operational Research*, 269(1):341–352, 2018.

S. Pour, K. Rasmussen, J. Drake, and E. Burke. A constructive framework for the preventive signalling maintenance crew scheduling problem in the danish railway system. *Journal of the Operational Research Society*, 70(11):1965–1982, 2019.

V. Raman and N. S. Gill. Review of different heuristic algorithms for solving travelling salesman problem. *International Journal of Advanced Research in Computer Science*, 8(5), 2017.

S. Riazi, O. Wigstrom, K. Bengtsson, and B. Lennartson. Decomposition and distributed algorithms for home healthcare routing and scheduling problem. *22nd IEEE International Conference on Emerging Technologies and Factory Automation*, pages 1–7, 2017.

M. M. Solomon. Algorithms for the vehicle routing and scheduling problems with time window constraints. *Operations Research*, 35(2):254–265, 1987.

D. Taş, M. Gendreau, O. Jabali, and G. Laporte. The traveling salesman problem with time-dependent service times. *European Journal of Operational Research*, 248(2):372–383, 2016.

M. D. Toksarı and E. Güner. Parallel machine scheduling problem to minimize the earliness/tardiness costs with learning effect and deteriorating jobs. *Journal of Intelligent Manufacturing*, 21(6):843–851, 2010.

P. Toth and D. Vigo. *The vehicle routing problem*. SIAM, 2002.

I. Toumazis and C. Kwon. Routing hazardous materials on time-dependent networks using conditional value-at-risk. *Transportation Research Part C: Emerging Technologies*, 37:73–92, 2013.

W. Tu, Q. Li, Q. Li, J. Zhu, B. Zhou, and B. Chen. A spatial parallel heuristic approach for solving very large-scale vehicel routing problems. *Transactions in GIS*, 21(6):1130–1147, 2017.

T. Wang, R. Baldacci, A. Lim, and Q. Hu. A branch-and-price algorithm for scheduling of deteriorating jobs and flexible periodic maintenance on a single machine. *European Journal of Operational Research*, 271(3): 826–838, 2018.

F. Wex, G. Schryen, and D. Neumann. Operational emergency response under informational uncertainty: A fuzzy optimization model for scheduling and allocating rescue units. *International Conference on Information Systems for Crisis Response and Management (ISCRAM)*, 2012.