The Machine Learning and Traveling Repairman Problem

Theja Tulabandhula, Cynthia Rudin, and Patrick Jaillet

Massachusetts Institute of Technology, Cambridge MA 02139, USA
`{theja,rudin,jaillet}@mit.edu`

Abstract. The goal of the *Machine Learning and Traveling Repairman Problem* (ML&TRP) is to determine a route for a "repair crew," which repairs nodes on a graph. The repair crew aims to minimize the cost of failures at the nodes, but the failure probabilities are not known and must be estimated. If there is uncertainty in the failure probability estimates, we take this uncertainty into account in an unusual way; from the set of acceptable models, we choose the model that has the lowest cost of applying it to the subsequent routing task. In a sense, this procedure agrees with a managerial goal, which is to show that the data can support choosing a low-cost solution.

Keywords: machine learning, traveling repairman, integer programming, uncertainty, generalization bound, constrained linear function classes

1 Introduction

We consider the problem of determining a route for a "repair crew" on a graph, where each node on the graph has some probability of failure. These probabilities are not known and must be estimated from past failure data. Intuitively the nodes that are more prone to failure should be repaired first. But if those nodes are far away from each other, the extra time spent by the repair crew traveling between nodes might actually increase the chance of failures occurring at nodes that have not yet been repaired. In that sense, it is better to construct the route to minimize the possible cost of failures, taking into account the travel time between nodes and also the (estimated) failure probabilities at each of the nodes. We call this problem the *machine learning and traveling repairman problem* (ML&TRP). There are many possible applications of the ML&TRP, including the scheduling of safety inspections or repair work for the electrical grid, oil rigs, underground mining, machines in a factory, or airlines.

One key idea we present here concerns the way that uncertainty is handled in probabilistic modeling, and the way the uncertainty relates to how the models are used in applications. Namely, when there is uncertainty in modeling, our idea is to choose a model that has advantages for our specific application, when we act on the predictions made by the model. For estimation problems, it is possible for many predictive models to be equally plausible, given finite data. In standard statistical and machine learning practice, we choose one of these models, but the choice of model is oblivious to the way that the model will be used in the application. Our idea is that we choose a model that predicts well, but that also has the advantage that it has a low "operating cost," which is the cost to act on the predictions made by the model. In this work, among all equally good predictive models for failure probabilities, we choose the one that leads to the lowest failure cost.

We present two formulations for the ML&TRP. The first formulation is *sequential*: the failure probabilities are estimated in a way that is oblivious to the failure cost; then, the route is determined by minimizing failure cost (which depends on the chosen probabilistic model). The second formulation handles uncertainty as discussed above, by computing the failure probabilities and the route *simultaneously*. This means that the estimated failure probabilities and the route are chosen together in a way that the failure cost will be low if possible; when there is uncertainty, the simultaneous formulation chooses the model with the lowest failure cost. The simultaneous formulation is optimistic; it provides the best possible, but still reasonable, scenario described by the data. The company might wish to know whether it is at all possible that a low-failure-cost route can be designed that is realistically supported by the data; the simultaneous formulation finds such a solution.

We design the failure cost in two ways, where either can be used for the sequential and the simultaneous formulations. The first failure cost is proportional to the sum (over nodes) of the expected number of failures at each node. The second failure cost considers, for each node, the probability that the first failure is before the repair crew's visit to the node. The first cost applies when the failure probability of a node does not change until it is visited by the crew, regardless of whether a failure already occurred at that node, and the second cost applies when the node is completely repaired after the first failure, or when it is visited by the repair crew, whichever comes first. In either case, the failure cost reduces to a weighted *traveling repairman problem* (TRP) objective [1].

The ML&TRP relates to literature on both machine learning and optimization (time-dependent traveling salesman problems). In machine learning, the use of unlabeled data has been explored extensively in the semi-supervised learning literature [2]. The ML&TRP does not fall under the umbrella of semi-supervised learning, since the incorporation of unlabeled data is used solely for determining the failure cost, and is not used to provide additional *distributional* information. Our work is slightly closer to work on graph-based regularization [3–5], but their goal is to obtain probability estimates that are smoothed on a graph with suitably designed edge weights. On the other hand, our goal is to obtain, in addition to probability estimates, a low-cost route for traversing a very different graph with edge weights that are physical distances. Our work contributes to the literature on the TRP and related problems by adding the new dimension of probabilistic estimation at the nodes. We adapt techniques from [6–8] within our work for solving the TRP part of the ML&TRP.

One particularly motivating application for the ML&TRP is smart grid maintenance. Since 2004, many power utility companies are implementing new inspection and repair programs for preemptive maintenance, where in the past, repair work was mainly made reactively [9]. Con Edison, which is New York City's power company, services tens of thousands of manholes (access points to underground grid) through new inspection and repair programs. The scheduling of manhole inspection and repair in Manhattan, Brooklyn and the Bronx is assisted by a statistical model [10]. This model does not take into account the route of the repair crew. This leaves open the possibility that, for this and for many other domains, estimating failure probabilities with knowledge of the repair crew's route could lead to an improvement in operations.

In Section 2, we provide the two formulations and the two ways of modeling failure cost. In Section 3, we describe mixed-integer nonlinear programs (MINLP) and algorithms for solving the ML&TRP. Section 4 gives an example and some experiments on data from the NYC power grid. Section 5 states a generalization result, and Section 6 concludes the paper.

2 ML&TRP Formulations

Consider two sets of instances, $\{x_i\}_{i=1}^m$, $\{\tilde{x}_i\}_{i=1}^M$, with $x_i \in \mathcal{X}$, $\tilde{x}_i \in \mathcal{X}$ that are feature vectors with $\mathcal{X} \subset \mathbb{R}^d$. Let the x_i^j indicate the j-th coordinate of the feature vector x_i. For the first set of instances, we are also given labels $\{y_i\}_{i=1}^m$, $y_i \in \{-1, 1\}$. These instances and their labels are the set of training examples. For the maintenance application, each of the $\{x_i\}_{i=1}^m$ encode manhole information (e.g., number and types of cables, number and types of previous events, etc.) and the labels $\{y_i\}_{i=1}^m$ encode whether the manhole failed ($y_i = 1$) or not ($y_i = -1$). More details about the features and labels can be found in Section 4. The other instances $\{\tilde{x}_i\}_{i=1}^M$ (with M unrelated to m), are unlabeled data that are each associated with a node on a graph G. The nodes of the graph G indexed by $i = 1, ..., M$ represent manholes on which we want to design a route. We are also given physical distances $d_{i,j} \in \mathbb{R}_+$ between all pairs of nodes i and j. A route on G is represented by a permutation π of the node indices $1, \dots, M$. Let Π be the set of all permutations of $\{1, ..., M\}$. A set of failure probabilities will be estimated at the nodes and these estimates will be based on a function of the form $f_\lambda(x) = \lambda \cdot x$. The class of functions $\mathcal{F}$ is chosen to be:

$$\mathcal{F} := \{f_\lambda : \lambda \in \mathbb{R}^d, ||\lambda||_2 \leq M_1\}, \tag{1}$$

where M_1 is a fixed positive real number.

The sequential formulation has a machine learning step and a traveling repairman step, whereas the simultaneous formulation has both ML and TRP together in the first step, and the second step uses the route from the first step.

Sequential Formulation
Step 1. (ML) Compute the values $f_\lambda^*(\tilde{x}_i)$:

$$f_\lambda^* \in \text{argmin}_{f_\lambda \in \mathcal{F}} \text{TrainingError}(f_\lambda, \{x_i, y_i\}_{i=1}^m).$$

Step 2. (TRP) Compute a route using estimated scores on $\{\tilde{x}_i\}_{i=1}^M$:

$$\pi^* \in \text{argmin}_{\pi \in \Pi} \text{FailureCost}(\pi, f_\lambda^*, \{\tilde{x}_i\}_{i=1}^M, \{d_{i,j}\}_{i,j=1}^M).$$

We will define the TrainingError and FailureCost shortly.

Simultaneous Formulation
Step 1. Compute the values $f_\lambda^*(\tilde{x}_i)$:

$$f_\lambda^* \in \text{argmin}_{f_\lambda \in \mathcal{F}} \Big[\text{TrainingError}(f_\lambda, \{x_i, y_i\}_{i=1}^m) \\ + C_1 \min_{\pi \in \Pi} \text{FailureCost}\left(\pi, f_\lambda, \{\tilde{x}_i\}_{i=1}^M, \{d_{i,j}\}_{i,j=1}^M\right) \Big].$$

Step 2. Compute a route corresponding to the scores:

$$\pi^* \in \text{argmin}_{\pi \in \Pi} \text{FailureCost}(\pi, f_\lambda^*, \{\tilde{x}_i\}_{i=1}^M, \{d_{i,j}\}_{i,j=1}^M).$$

A transformation of $f_\lambda^*(x)$ yields an estimate of the probability of failure $P(y = 1|x)$ (we discuss this later, see (2)). In Step 1, f_λ^* is chosen to yield probability estimates that agree with the training data, but at the same time, yield lower failure costs. The user-defined constant C_1 is a tradeoff parameter, moving from "oblivious" estimation models to cost-aware estimation models. When C_1 is small, the algorithm essentially becomes sequential, ignoring the FailureCost. When it is large, the algorithm is highly biased towards low FailureCost solutions. One might want to choose C_1 large when there is a lot of uncertainty in the estimates and a strong belief that a very low cost solution exists. Or, one could choose a large C_1 to determine what policy would be chosen when the cost is underestimated. A small C_1 is appropriate when the number of training examples is large enough so that there is little flexibility (uncertainty) in the choice of model f_λ^*. Or one would choose low C_1 when we wish to choose, among equally good solutions, the one with the lowest cost. We now define the TrainingError and two options for the FailureCost.

TrainingError. In learning, the unregularized error is a sum (or average) of losses over the training examples: $\sum_{i=1}^m l(f_\lambda(x_i), y_i)$, where the loss function $l(\cdot, \cdot)$ can be any monotonic smooth function bounded below by zero. We choose the logistic loss: $l(f_\lambda(x), y) := \ln\left(1 + e^{-yf_\lambda(x)}\right)$ so that the probability of failure $P(y = 1|x)$, is estimated as in logistic regression by

$$P(y = 1|x) \text{ or } p(x) := \frac{1}{1 + e^{-f_\lambda(x)}}. \qquad (2)$$

The negative log likelihood is:

$$\sum_{i=1}^m - \ln\left[p(x_i)^{(1+y_i)/2}(1 - p(x_i))^{(1-y_i)/2}\right] = \sum_{i=1}^m \ln\left(1 + e^{-y_i f_\lambda(x_i)}\right).$$

We then add an ℓ_2 penalty over the parameters λ (with coefficient C_2) to get

$$\text{TrainingError}(f_\lambda, \{x_i, y_i\}_{i=1}^m) := \sum_{i=1}^m \ln\left(1 + e^{-y_i f_\lambda(x_i)}\right) + C_2||\lambda||_2^2. \tag{3}$$

The coefficient C_2 is inversely related to the constant M_1 in (1) and both represent the same constraint on the function class. C_2 is useful for algorithm implementations whereas M_1 is useful for analysis.

Two Options for FailureCost. In the first option (denoted as Cost 1), for each node there is a cost for (possibly repeated) failures prior to a visit by the repair crew. In the second option (denoted as Cost 2), for each node, there is a cost for the first failure prior to visiting it. There is a natural interpretation of the failures as being generated by a continuous random process at each of the nodes. When discretized in time, this is approximated by a Bernoulli process with parameter $p(\tilde{x}_i)$. Both Cost 1 and Cost 2 are appropriate for power grid applications. Cost 2 is also appropriate for delivery truck routing applications, where perishable items can fail (once an item has spoiled, it cannot spoil again). For many applications, neither of these two costs apply, in which case, it is possible to design a more appropriate or specialized cost and use that in place of the two we present here, using the same general idea of combining this cost with the training error to produce an algorithm.

Without loss of generality, we assume that after the repair crew visits all the nodes, it returns to the starting node (node 1) which is fixed beforehand. Scenarios where one is not interested in beginning from or returning to the starting node would be modeled slightly differently (the computational complexity remains the same).

Let a route be represented by $\pi : \{1, ..., M\} \mapsto \{1, ..., M\}$, ($\pi(i)$ is the i^{th} node visited). Let the distances be such that a unit of distance is traversed in a unit of time. Given a route, the *latency* of a node $\pi(i)$ is the time (or equivalently distance) from the start at which node $\pi(i)$ is visited

$$L_\pi(\pi(i)) := \begin{cases} \sum_{k=1}^M d_{\pi(k)\pi(k+1)} \mathbf{1}_{[k<i]} & i = 2, ..., M \\ \sum_{k=1}^M d_{\pi(k)\pi(k+1)} & i = 1, \end{cases} \tag{4}$$

where we let $d_{\pi(M)\pi(M+1)} = d_{\pi(M)\pi(1)}$.

Cost 1 (*Cost is Proportional to Expected Number of Failures Before the Visit*). Up to the time that node $\pi(i)$ is visited, there is a probability $p(\tilde{x}_{\pi(i)})$ that a failure will occur in each unit time interval. This failure is determined by a Bernoulli random variable with parameter $p(\tilde{x}_{\pi(i)})$. Thus, in a time interval of length $L_\pi(\pi(i))$ units, the number of node failures follows a binomial distribution. For each node, we associate a cost proportional to the expected number of failures before the repair crew's visit:

$$\begin{aligned} \text{Cost of node } \pi(i) &\propto E(\text{number failures in } L_\pi(\pi(i)) \text{ time units}) \\ &= \text{mean of Bin}(L_\pi(\pi(i)), p(\tilde{x}_{\pi(i)})) = p(\tilde{x}_{\pi(i)})L_\pi(\pi(i)). \end{aligned} \tag{5}$$

If the failure probability for node $\pi(i)$ is small, we can afford to visit it later on in the route (the latency $L_\pi(\pi(i))$ is larger). If $p(\tilde{x}_{\pi(i)})$ is large, we visit node $\pi(i)$ earlier to keep our cost low. The failure cost for a route π is

$$\text{FailureCost}(\pi, f_\lambda, \{\tilde{x}_i\}_{i=1}^M, \{d_{i,j}\}_{i,j=1}^M) = \sum_{i=1}^M p(\tilde{x}_{\pi(i)}) L_\pi(\pi(i)).$$

Substituting the definition of $L_\pi(\pi(i))$ from (4):

$$\text{FailureCost}(\pi, f_\lambda, \{\tilde{x}_i\}_{i=1}^M, \{d_{i,j}\}_{i,j=1}^M) =$$
$$\sum_{i=2}^M p(\tilde{x}_{\pi(i)}) \sum_{k=1}^M d_{\pi(k)\pi(k+1)} \mathbf{1}_{[k<i]} + p(\tilde{x}_{\pi(1)}) \sum_{k=1}^M d_{\pi(k)\pi(k+1)}, \tag{6}$$

where $p(\tilde{x}_{\pi(i)})$ is given in (2). In a more general setting (explored in a longer version of this work [11]), we could relax the assumption of setting $p(\tilde{x}_{\pi(i)}) = 0$ after the visit as we have implicitly done here. Note that since the cost is a sum of M terms, it is invariant to ordering or indexing (caused by π) and we can rewrite it as

$$\text{FailureCost}(\pi, f_\lambda, \{\tilde{x}_i\}_{i=1}^M, \{d_{i,j}\}_{i,j=1}^M) = \sum_{i=1}^M p(\tilde{x}_i) L_\pi(i).$$

Cost 2 (*Cost is Proportional to Probability that First Failure is Before the Visit*). This cost reflects the penalty for not visiting a node before the first failure occurs there. The model is governed by the geometric distribution: the probability that the first failure for node $\pi(i)$ occurs at time $L_\pi(\pi(i))$ is $p(\tilde{x}_{\pi(i)})(1 - p(\tilde{x}_{\pi(i)}))^{L_\pi(\pi(i))-1}$, and the cost of visiting node $\pi(i)$ is proportional to:

$$P\Big(\text{first failure occurs before } L_\pi(\pi(i))\Big) = 1 - (1 - p(\tilde{x}_{\pi(i)}))^{L_\pi(\pi(i))}$$
$$= 1 - \left(1 - \frac{1}{1 + e^{-f_\lambda(\tilde{x}_{\pi(i)})}}\right)^{L_\pi(\pi(i))} = 1 - \left(1 + e^{f_\lambda(\tilde{x}_{\pi(i)})}\right)^{-L_\pi(\pi(i))}. \tag{7}$$

Similarly to Cost 1, $L_\pi(\pi(i))$ influences the cost at each node. If we visit a node early in the route, then the cost incurred is small because the node is less likely to fail before we reach it. Similarly, if we schedule a visit later on in the tour, the cost is higher because the node has a higher chance of failing prior to the repair crew's visit. The total failure cost is

$$\sum_{i=1}^M \left(1 - \left(1 + e^{f_\lambda(\tilde{x}_{\pi(i)})}\right)^{-L_\pi(\pi(i))}\right). \tag{8}$$

This cost is not directly related to a weighted TRP cost in its present form, but building on this, we will derive a cost that is the same as a weighted TRP. Before doing so in Section 3, we formulate the integer program for the simultaneous formulation for Cost 1.

3 Optimization

Mixed-Integer Optimization for Cost 1. For both the sequential and simultaneous formulations, we need to solve the TRP subproblem:

$$\begin{aligned}\pi^* &\in \text{argmin}_{\pi\in\Pi}\text{FailureCost}(\pi, f^*_\lambda, \{\tilde{x}_i\}_{i=1}^M, \{d_{i,j}\}_{i,j=1}^M),\\ &= \text{argmin}_{\pi\in\Pi}\sum_{i=2}^M p(\tilde{x}_{\pi(i)})\sum_{k=1}^M d_{\pi(k)\pi(k+1)}\mathbf{1}_{[k<i]} + p(\tilde{x}_{\pi(1)})\sum_{k=1}^M d_{\pi(k)\pi(k+1)} \quad (9)\end{aligned}$$

The standard TRP objective is a special case of weighted TRP (9) when $\forall i = 1,...,M,\ p(\tilde{x}_i) = p$. The TRP is different from the traveling salesman problem (TSP); the goal of the TSP is to minimize the total traversal time (in this case, this is the same as the distance traveled) needed to visit all nodes once, whereas the goal of the TRP is to minimize the sum of the waiting times to visit each node. Both the problems are known to be NP-complete in the general case [12].

We extend the integer linear program (ILP) of [6] to include "unequal flow values" in (9). For interpretation, consider the sum $\sum_{i=1}^M \bar{p}(\tilde{x}_i)$ as the total "flow" through a route where $\bar{p}(\tilde{x}_i)$ will be chosen later according to either Cost 1 or Cost 2. At the beginning of the tour, the repair crew has flow $\sum_{i=1}^M \bar{p}(\tilde{x}_i)$. Along the tour, flow of the amount $\bar{p}(\tilde{x}_i)$ is dropped when the repair crew visits node $\pi(i)$ at latency $L_\pi(\pi(i))$. We introduce two sets of variables $\{z_{i,j}\}_{i,j}$ and $\{y_{i,j}\}_{i,j}$ which can together represent a route (instead of the π notation). Let $z_{i,j}$ represent the flow on edge (i,j) and let a binary variable $y_{i,j}$ represent whether there exists a flow on edge (i,j). Then the mixed ILP is:

$$\min_{z,y}\sum_{i=1}^M\sum_{j=1}^M d_{i,j}z_{i,j} \quad \text{s.t.} \quad (10)$$

$$\text{No flow from node } i \text{ to itself: } z_{i,i} = 0 \quad \forall i = 1,...,M \quad (11)$$

$$\text{No edge from node } i \text{ to itself: } y_{i,i} = 0 \quad \forall i = 1,...,M \quad (12)$$

$$\text{Exactly one edge into each node: } \sum_{i=1}^M y_{i,j} = 1 \quad \forall j = 1,...,M \quad (13)$$

$$\text{Exactly one edge out from each node: } \sum_{j=1}^M y_{i,j} = 1 \quad \forall i = 1,...,M \quad (14)$$

$$\text{Flow coming back at the end of the loop is } \bar{p}(\tilde{x}_1)\text{: } \sum_{i=1}^M z_{i,1} = \bar{p}(\tilde{x}_1) \quad (15)$$

Change of flow after crossing node k is:

$$\sum_{i=1}^M z_{i,k} - \sum_{j=1}^M z_{k,j} = \begin{cases}\bar{p}(\tilde{x}_1) - \sum_{i=1}^M \bar{p}(\tilde{x}_i) & k = 1\\ \bar{p}(\tilde{x}_k) & k = 2,...,M\end{cases} \quad (16)$$

$$\text{Connects flows } z \text{ to indicators of edge } y\text{: } z_{i,j} \le r_{i,j}y_{i,j} \quad (17)$$

$$\text{where } r_{i,j} = \begin{cases} \bar{p}(\tilde{x}_1) & j = 1 \\ \sum_{i=1}^{M} \bar{p}(\tilde{x}_i) & i = 1 \\ \sum_{i=2}^{M} \bar{p}(\tilde{x}_i) & \text{otherwise.} \end{cases}$$

Constraints (11) and (12) restrict self-loops from forming. Constraints (13) and (14) impose that every node should have exactly one edge coming in and one going out. Constraint (15) represents the flow on the last edge coming back to the starting node. Constraint (16) quantifies the flow change after traversing a node k. Constraint (17) represents an upper bound on $z_{i,j}$ relating it to the corresponding binary variable $y_{i,j}$.

Mixed-Integer Optimization for Cost 2. By applying the log function to the cost of each node (7) (and subtracting a constant), we can minimize a more tractable cost objective:

$$\text{FailureCost} = \min_{\pi} \sum_{i=1}^{M} L_{\pi}(\pi(i)) \log\left(1 + e^{f_{\lambda}(\tilde{x}_{\pi(i)})}\right).$$

This failure cost term is now a weighted sum of latencies where the weights are of the form $\log\left(1 + e^{f_{\lambda}(\tilde{x}_{\pi(i)})}\right)$. We can thus reuse the mixed ILP (10)-(17) where the weights are redefined as $\bar{p}(\tilde{x}_i) := \log\left(1 + e^{\lambda \cdot \tilde{x}_i}\right)$.

We have thus shown how to solve the weighted TRP subproblem, and we will now present ways to solve the full ML&TRP.

Mixed-Integer Nonlinear Programs (MINLPs) for Simultaneous Formulation. The full objective using Cost 1 is:

$$\min_{\lambda} \left(\sum_{i=1}^{m} \ln\left(1 + e^{-y_i f_{\lambda}(x_i)}\right) + C_2 ||\lambda||_2^2 + C_1 \min_{\{z_{i,j}, y_{i,j}\}} \sum_{i=1}^{M} \sum_{j=1}^{M} d_{i,j} z_{i,j} \right) \tag{18}$$

$$\text{such that constraints (11) to (17) hold, where } \bar{p}(\tilde{x}_i) = \frac{1}{1 + e^{-\lambda \cdot \tilde{x}_i}}.$$

The full objective using the modified version of Cost 2 is:

$$\min_{\lambda} \left(\sum_{i=1}^{m} \ln\left(1 + e^{-y_i f_{\lambda}(x_i)}\right) + C_2 ||\lambda||_2^2 + C_1 \min_{\{z_{i,j}, y_{i,j}\}} \sum_{i=1}^{M} \sum_{j=1}^{M} d_{i,j} z_{i,j} \right) \tag{19}$$

$$\text{such that constraints (11) to (17) hold, where } \bar{p}(\tilde{x}_i) = \log\left(1 + e^{\lambda \cdot \tilde{x}_i}\right).$$

If we have an algorithm for solving (18), then the same scheme can be used to solve (19). There are multiple ways of solving (or approximately solving) a mixed integer nonlinear optimization problem of the form (18) or (19). We consider three methods. The first method is to directly use a generic *mixed integer non-linear programming* (MINLP) solver. The second and third methods (called

Nelder-Mead and *Alternating Minimization*, denoted NM and AM respectively) are iterative schemes over the λ parameter space. At every iteration of these algorithms, we will need to evaluate the objective function. This evaluation involves solving an instance of the weighted TRP subproblem. For the AM algorithm, define Obj as follows:

$$\begin{aligned}\mathrm{Obj}(\lambda,\pi) = &\ \mathrm{TrainingError}(f_\lambda, \{x_i, y_i\}_{i=1}^m) \\ &+ C_1 \mathrm{FailureCost}\left(\pi, f_\lambda, \{\tilde{x}_i\}_{i=1}^M, \{d_{i,j}\}_{i,j=1}^M\right). \qquad (20)\end{aligned}$$

Starting from an initial vector λ_0, Obj is minimized alternately with respect to λ and then with respect to π, as shown in Algorithm 1.1.

```
Inputs: $\{x_i, y_i\}_1^m, \{\tilde{x}_i\}_1^M, \{d_{ij}\}_{ij}, C_1, C_2, T$ and initial vector $\lambda_0$.
for t=1:T do
  Compute $\pi_t \in \mathrm{argmin}_{\pi \in \Pi} \mathrm{Obj}(\lambda_{t-1}, \pi)$ (mixed ILP).
  Compute $\lambda_t \in \mathrm{argmin}_{\lambda \in \mathbb{R}^d} \mathrm{Obj}(\lambda, \pi_t)$ (Gradient descent).
end for
Output: $\pi_T$.
```

Algorithm 1.1. AM: Alternating minimization algorithm

4 Experiments

We have now defined two formulations (sequential and simultaneous), each with two possible definitions for the failure cost (Cost 1 and Cost 2), and three algorithms for the simultaneous formulation (MINLP solver, NM, and AM). In what follows, we will highlight the advantage of the simultaneous method over the less general sequential method through two experiments. The first involves a very simple synthetic dataset, designed to show differences between the two methods. The second experiment involves a real dataset, designed as part of a collaboration with NYC's power company, Con Edison (see [10] for a more detailed description of these data). In each experiment, we solve the simultaneous formulation over a range of values of C_1 and compare the routes and failure estimates obtained over this range. Our goal for this section is to illustrate that incorporating the routing cost into the machine learning model can produce lower cost solutions in at least some scenarios, without harming prediction accuracy. For both experiments, we have a fixed training set and separate test set to evaluate predictions of the model, and the unlabeled set of nodes with distances. In both experiments, there is a lot of uncertainty in the estimates for the unlabeled set. In the toy example, the unlabeled set is in a low density region, so the probabilities could reasonably change without substantially affecting prediction ability. In the second experiment, the data are very imbalanced (the positive class is very rare), so there is a lot of uncertainty in the estimates, and further, there is a prior belief that a low-cost route exists. In particular, we have reason to believe that some of the probabilities are overestimated in this particular experiment using the particular unlabeled set we chose, and that knowing the repair route can help to determine these probabilities; this is because there are underground electrical cables traversing each linear stretch of the repair route.

Toy Example. We illustrate how the simultaneous formulation takes advantage of uncertainty; it is because a small change in the probabilities can give a completely different route and cost. Consider the graph G shown in Figure 1(a) and Figure 1(b). Figure 1(c) shows unlabeled points $\{\tilde{x}_i\}_{i=1}^4 \in \mathbb{R}^2$ along with the training instances (represented by two gray clusters). The sequential formulation produces a function f_λ^* whose 0.5-probability level set is shown as a black line here. The route corresponding to that solution is given in Figure 1(a), which is $\pi^* = 1-3-2-4-1$. If we were to move the 0.5-probability level set slightly, for instance to the dashed line in Figure 1(c) by using an appropriate tradeoff parameter C_1 in the simultaneous formulation, the probability estimates on the finite training set change only slightly, but the cost and the corresponding route change entirely (Figure 1(b)). The new route is $\pi^* = 1-3-4-2-1$, and yields a lower value of Cost 1 (a decrease of ∼16.4%). In both cases, the probability estimators have very similar validation performance, but the solutions on the graph are different.

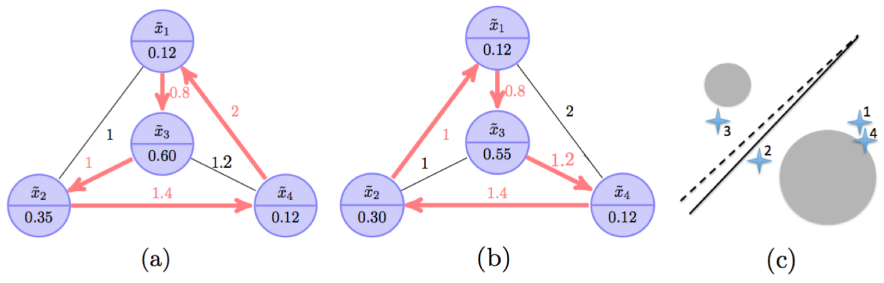

Fig. 1. For the above graphs, the numbers in the nodes indicate their probability of failures and the numbers on the edges indicate distances. (a) Route as determined by sequential formulation (*highlighted*). (b) Route determined by the simultaneous formulation. (c) The feature space.

The NYC Power Grid. We have information related to manholes from the Bronx, NYC (∼23K manholes). Each manhole is represented by (4 dimensional) features that encode the number and type of electrical cables entering the manhole and the number and type of past events involving the manhole. The training features encode events prior to 2008, and the training labels are 1 if the manhole was the source of a serious event (fire, explosion, smoking manhole) during 2008.

The prediction task is to predict events in 2009. The test set (for evaluating the performance of the predictive model) consists of features derived from the time period before 2009, and labels from 2009. Predicting manhole events can be a difficult task for machine learning, because one cannot necessarily predict an event using the available data. The operational task is to design a route for a repair crew that is fixing seven manholes in 2009 on which we want the cost of failures to be low. Because of the large class imbalance, the misclassification error is almost always the size of the whole positive class. Because of this, we evaluate the quality of the predictions from f_{λ^*} using the area under the ROC curve (AUC), for both training and test.

We solve (18) and (19) using an appropriate range of values for the regularization parameter C_1, with the goal of seeing whether for the same level of estimation performance, we can get a reduction in the cost of failures. Note that the uncertainty in the estimation of failure probabilities is due to the finite number of examples in the training set. The other regularization parameter C_2 is kept fixed throughout (in practice one might use cross-validation if C_2 is allowed to vary). The evaluation metric AUC is a measure of ranking quality; it is sensitive to the rank-ordering of the nodes in terms of their probability to fail, and it is not as sensitive to changes in the values of these probabilities. This means that as the parameter C_1 increases, the estimated probability values will tend to decrease, and thus the failure cost will decrease; it may be possible for this to happen without impacting the prediction quality as measured by the AUC, but this depends on the routes and it is not guaranteed. In our experiments, for both training and test we had a large sample ($\sim$23K examples). The test AUC values for the simultaneous method were all within 1% of the values obtained by the sequential method; this is true for both Cost 1 and Cost 2, for each of the AM, NM, and MINLP solvers, see Figures 3(a) and 3(b). The variation in TrainingError across the methods was also small, about 2%, see Figure 3(c). So, changing C_1 did not dramatically impact the prediction quality as measured by the AUC. On the other hand, the failure costs varied widely over the different methods and settings of C_1, as a result of the decrease in the probability estimates, as shown in Figure 3(d). As C_1 was increased from 0.05 to 0.5, Cost 1 went from 27.5 units to 3.2 units, which is over eight times smaller. This means that with a 1-2% variation in the predictive model's AUC, the failure cost can decrease a lot, potentially yielding a more cost-effective route for inspection and/or repair work. The reason for an order of magnitude change in the failure cost is because the probability estimates are reducing by an order of magnitude due to uncertainty; yet our model still maintained the same level of AUC performance on training and test sets. Figure 2(a) shows the route provided by the sequential formulation. For the simultaneous formulation, there are changes in the cost and the route as the coefficient C_1 increases. When the failure cost term starts influencing the optimal solution of the objective (18), we get a new route as shown in Figure 2(b). This demonstration on data from the Bronx illustrates that it is possible to take advantage of uncertainty in modeling, in order to create a much more cost-effective solution.

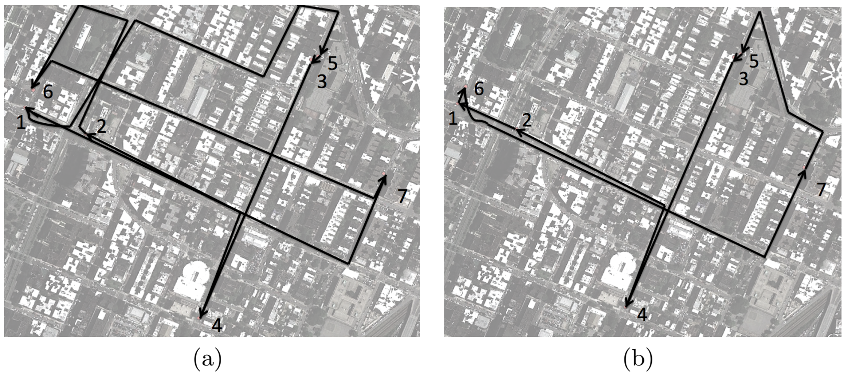

Fig. 2. (a) Sequential formulation route: 1-5-3-4-2-6-7-1. (b) Simultaneous formulation route ($C_1 = 0.5$): 1-6-7-5-3-4-2-1.

5 Generalization Bound

We initially introduced the failure cost regularization term in order to find scenarios where the data would support low-cost (more actionable) repair routes. From another point of view, incorporating regularization increases bias and reduces variance, and may thus allow us to obtain better prediction guarantees as we increase C_1. Any type of bias can either help or hurt the quality of the a statistical model, depending on whether the "prior belief" associated the bias is correct (this relates to "approximation error"). At the same time, incorporating bias helps to reduce the variance of the solution, reducing the difference between the training error we measure and the true error on the full population ("generalization error"). This difference is what we discuss in this section.

The "hypothesis space" is the set of models that an algorithm can choose from. When C_1 is large, it means we are only allowing models that yield low-cost solutions. This restriction on the hypothesis space (to the set of low-cost solutions) is a reduction in the size of this space. In statistical learning theory, the size of the hypothesis space is recognized as one of the most important quantities in the learning process, and this idea is formalized through probabilistic guarantees, *i.e.,* bounds on the generalization error. The bound we provide below shows how the TRP cost term (using Cost 1) reduces the size of the hypothesis space by removing a spherical cap, and how this could affect the generalization ability of the ML&TRP algorithms.

Define the true risk as the expectation of the logistic loss:

$$R(f_\lambda) := E_{(x,y)\sim\mathcal{X}\times\mathcal{Y}} l(f_\lambda(x), y) = \int \ln\left(1 + e^{-y f_\lambda(x)}\right) \partial\mu_{\mathcal{X}\times\mathcal{Y}}(x, y).$$

We bound $R(f_\lambda)$ by the empirical risk $R(f_\lambda, \{x_i, y_i\}_1^m) = \frac{1}{m}\sum_{i=1}^m \ln\left(1 + e^{-y_i f(x_i)}\right)$ plus a complexity term that depends on the geometry of where the nodes are located. Before we do this, we need to replace the Lagrange multiplier C_1 in (18)

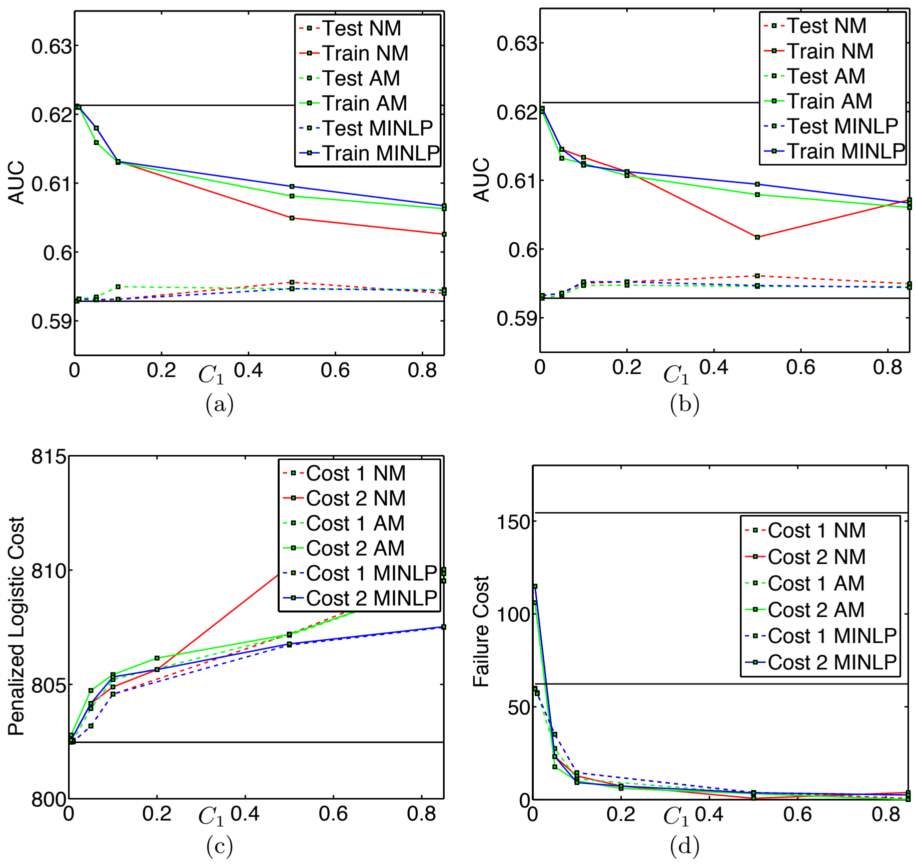

Fig. 3. For all the figures, horizontal lines represent baseline sequential formulation values for training or testing; x-axes represent values of C_1; the curves for the three algorithms (NM, AM and MINLP) are very similar to each other and the focus is on their trend with respect to C_1. (a) AUC values with Cost 1. (b) AUC values with Cost 2. (c) ℓ_2-regularized logistic loss. (d) Decreasing failure cost for both Cost 1 and 2.

with an explicit constraint, so f_λ is subject to a specific limit on the failure cost:

$$\min_{\pi} \sum_{i=1}^{M} \frac{1}{1+e^{-f_\lambda(\tilde{x}_{\pi(i)})}} L_\pi(\pi(i)) \leq C_g.$$

C_g is a constant (inversely related to C_1), and C_g will be a bias-variance tradeoff in the bound. Let $\sup_{x\in\mathcal{X}} ||x||_2 \leq M_2$, so $f_\lambda : \mathcal{X} \to [-M_1M_2, M_1M_2]$. Let us define the set of functions that are subject to a constraint on the failure cost as:

$$\mathcal{F}_0 := \left\{ f_\lambda : f_\lambda \in \mathcal{F}, \min_{\pi \in \Pi} \sum_{i=1}^{M} L_\pi(\pi(i)) \frac{1}{1+e^{-f_\lambda(\tilde{x}_{\pi(i)})}} \leq C_g \right\}.$$

Now we incorporate the geometry. Let d_i to be the shortest distance from the starting node to node i and let d_1 be the length of the shortest tour that visits all the nodes and returns to node 1. Define a vector c element-wise by:

$$c^j = \frac{\tilde{c}^j}{C_g - \tilde{c}_0} \text{ where } \tilde{c}^j = \frac{e^{M_1M_2}}{(1+e^{M_1M_2})^2} \left(\sum_i d_i \tilde{x}_i^j \right), \text{ where}$$

$$\tilde{c}_0 = \left(M_1M_2 \frac{e^{M_1M_2}}{(1+e^{M_1M_2})^2} + \frac{1}{1+e^{M_1M_2}} \right) \sum_i d_i.$$

This vector c incorporates both C_g and the d_i's that are the important ingredients in providing a generalization guarantee.

Theorem 1. ***(Generalization Bound)*** *Let $\mathcal{X} = \{x \in \mathbb{R}^d : ||x||_2 \leq M_2\}$, $\mathcal{Y} = \{-1, 1\}$. Let $\mathcal{F}_0$ be defined as above with respect to $\{\tilde{x}_i\}_{i=1}^M$, $\tilde{x}_i \in \mathcal{X}$ (not necessarily random). Let $\{x_i, y_i\}_{i=1}^m$ be a sequence of m examples drawn independently according to an unknown distribution $\mu_{\mathcal{X}\times\mathcal{Y}}$. Then for any $\epsilon > 0$,*

$$P\Big(\exists f_\lambda \in \mathcal{F}_0 : |R(f_\lambda, \{x_i, y_i\}_1^m) - R(f_\lambda)| > \epsilon\Big)$$
$$\leq 4\alpha(d, C_g, c) \left(\frac{32M_1M_2}{\epsilon} + 1 \right)^d \exp\left(\frac{-m\epsilon^2}{512(M_1M_2)^2} \right),$$

where

$$\alpha(d, C_g, c) := \frac{1}{2} + \frac{||c||_2^{-1} + \frac{\epsilon}{32M_2}}{M_1 + \frac{\epsilon}{32M_2}} \frac{\Gamma\left[1+\frac{d}{2}\right]}{\sqrt{\pi}\Gamma\left[\frac{d+1}{2}\right]} {}_2F_1\left(\frac{1}{2}, \frac{1-d}{2}; \frac{3}{2}; \left(\frac{||c||_2^{-1} + \frac{\epsilon}{32M_2}}{M_1 + \frac{\epsilon}{32M_2}} \right)^2 \right) \tag{21}$$

and where ${}_2F_1(a, b; c; d)$ is the hypergeometric function.

The term $\alpha(d, C_g, c)$ comes directly from formulae for the normalized volume of a spherical cap. Our goal was to establish that generalization can depend on C_g. As C_g decreases, the norm $\|c\|_2$ increases, and thus (21) decreases, and the whole bound decreases. Decreasing C_g may thus improve generalization. The proof is lengthy and is provided in a longer version [11].

6 Conclusion

In this work, we present a machine learning algorithm that takes into account the way its recommendations will be ultimately used. This algorithm takes advantage of uncertainty in the model in order to potentially find a much more practical solution. Including these operating costs is a new way of incorporating "structure" into machine learning algorithms, and we plan to explore this in other ways in ongoing work. We discussed the tradeoff between estimation error and operating cost for the specific application to the ML&TRP. In doing so, we showed a new way in which data dependent regularization can influence an algorithm's prediction ability, formalized through generalization bounds.

Acknowledgements This work is supported by an International Fulbright Science and Technology Award, the MIT Energy Initiative, and the National Science Foundation under Grant IIS-1053407.

References

1. Jean-Claude Picard and Maurice Queyranne. The time-dependent traveling salesman problem and its application to the tardiness problem in one-machine scheduling. *Operations Research*, 26(1):86–110, January–February 1978.
2. Olivier Chapelle, Bernhard Schölkopf, and Alexander Zien, editors. *Semi-Supervised Learning.* MIT Press, Cambridge, MA, 2006.
3. Shivani Agarwal. Ranking on graph data. In *Proceedings of the 23rd International Conference on Machine Learning*, 2006.
4. Mikhail Belkin, Partha Niyogi, and Vikas Sindhwani. Manifold regularization: A geometric framework for learning from labeled and unlabeled examples. *Journal of Machine Learning Research*, 7:2399–2434, 2006.
5. Dengyong Zhou, Jason Weston, Arthur Gretton, Olivier Bousquet, and Bernhard Schölkopf. Ranking on data manifolds. In *Advances in Neural Information Processing Systems 16*, pages 169–176. MIT Press, 2004.
6. Matteo Fischetti, Gilbert Laporte, and Silvano Martello. The delivery man problem and cumulative matroids. *Oper. Res.*, 41:1055–1064, November 1993.
7. C. A. van Eijl. A polyhedral approach to the delivery man problem. Memorandum COSOR 95–19, Department of Mathematics and Computer Science, Eindhoven University of Technology, The Netherlands, 1995.
8. Miriam Lechmann. The traveling repairman problem - an overview. Diplomarbeit, Universitat Wein, 1–79, 2009.
9. Ian Urbina. Mandatory safety rules are proposed for electric utilities. *New York Times*, 08-21-2004. Late Edition, Sec B, Col 3, Metropolitan Desk, pg 2.
10. Cynthia Rudin, Rebecca Passonneau, Axinia Radeva, Haimonti Dutta, Steve Ierome, and Delfina Isaac. A process for predicting manhole events in Manhattan. *Machine Learning*, 80:1–31, 2010.
11. Theja Tulabandhula and Cynthia Rudin and Patrick Jaillet. Machine Learning and the Traveling Repairman. `arXiv:1104.5061`, 2011.
12. Avrim Blum, Prasad Chalasani, Don Coppersmith, Bill Pulleyblank, Prabhakar Raghavan, and Madhu Sudan. On the minimum latency problem. *Proceedings of the 26th ACM symposium on theory of computing*, 163–171, September 1994.