Robust Reward Placement under Uncertainty

Petros Petsinis[1] , **Kaichen Zhang**[2] , **Andreas Pavlogiannis**[1] , **Jingbo Zhou**[3] and **Panagiotis Karras**[4,1]
[1]Department of Computer Science, Aarhus University
[2]Artificial Intelligence Thrust, Hong Kong University of Science and Technology (Guangzhou)
[3]Business Intelligence Lab, Baidu Research
[4]Department of Computer Science, University of Copenhagen
petsinis@cs.au.dk , kzhangbi@connect.ust.hk , pavlogiannis@cs.au.dk , zhoujingbo@baidu.com , piekarras@gmail.com

Abstract

We consider a problem of placing generators of rewards to be collected by randomly moving agents in a network. In many settings, the precise mobility pattern may be one of several possible, based on parameters outside our control, such as weather conditions. The placement should be *robust* to this uncertainty, to gain a competent total reward across possible networks. To study such scenarios, we introduce the *Robust Reward Placement* problem (RRP). Agents move randomly by a Markovian Mobility Model with a predetermined set of locations whose connectivity is chosen adversarially from a known set Π of candidates. We aim to select a set of reward states within a budget that maximizes the minimum ratio, among all candidates in Π, of the collected total reward over the optimal collectable reward under the same candidate. We prove that RRP is NP-hard and inapproximable, and develop Ψ-Saturate, a pseudo-polynomial time algorithm that achieves an ϵ-additive approximation by exceeding the budget constraint by a factor that scales as $\mathcal{O}(\ln |\Pi|/\epsilon)$. In addition, we present several heuristics, most prominently one inspired by a dynamic programming algorithm for the $\max$–$\min$ 0–1 KNAPSACK problem. We corroborate our theoretical analysis with an experimental evaluation on synthetic and real data.

1 Introduction

In many graph optimization problems, a stakeholder has to select locations in a network, such as a road, transportation, infrastructure, communication, or web network, where to place reward-generating facilities such as stores, ads, sensors, or utilities to best service a population of moving agents such as customers, autonomous vehicles, or bots [Zhang and Vorobeychik, 2016; Ostachowicz *et al.*, 2019; Zhang *et al.*, 2020; Rosenfeld and Globerson, 2016; Amelkin and Singh, 2019]. Such problems are intricate due to the uncertainty surrounding agent mobility [Krause *et al.*, 2008; Chen *et al.*, 2016; He and Kempe, 2016; Horčík *et al.*, 2022].

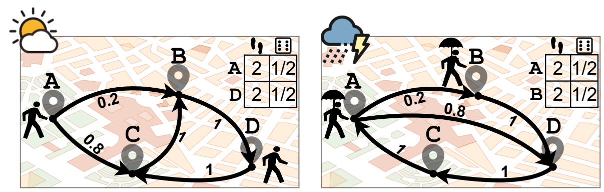

Figure 1: Moving agent under two settings; *sunny* and *rainy*; tables show numbers of steps and initial probabilities.

For instance, consider outdoor ad placement. We represent the road map as a probabilistic network in which agents move. If every agent follows the same movement pattern regardless of environmental conditions, then the problem of placing ads to maximize the expected number of ad views admits a greedy algorithm with an approximation ratio [Zhang *et al.*, 2020]. Still, the problem becomes more involved under malleable environmental conditions that alter movement patterns. As a toy example, Figure 1 shows a probabilistic network. An agent randomly starts from an initial location and takes two steps by the probabilities shown on edges representing street segments, under two environmental settings, *sunny* and *rainy*. Assume a stakeholder has a budget to place an ad-billboard at a *single location*. Under the *sunny* setting, the best choice of placement is B, as the agent certainly passes by that point regardless of its starting position; under the *rainy* setting, the agent necessarily passes by D within two steps, hence that is most preferable. However, under the *rainy* setting B yields expected reward 0.6, and so does D under the *sunny* one. Due to such uncertainty, a risk-averse stakeholder would prefer the location that yields, in the worst case, the highest ratio of the collected to best feasible reward, i.e., in this case, C, which yields expected reward 0.9 under both settings.

In this paper, we introduce the problem of robust reward placement (RRP) in a network, under uncertainty about the environment whereby an agent is moving according to any of several probabilistic mobility settings. We express each such setting by a Markov Mobility Model (MMM) $\pi \in \Pi$. The cumulative reward a stakeholder receives grows whenever the agent passes by one of the *reward states* $\mathcal{S_R}$. RRP seeks to select a set of such states $\mathcal{S}^*_{\mathcal{R}}$ within a budget, that maximizes the worst-case ratio, across all settings Π, of the collected reward $F(\mathcal{S}_\mathcal{R}|\pi)$ over the highest reward that can be collected under the same setting $F(\mathcal{S}^*_\pi|\pi)$, i.e., $\mathcal{S}^*_\mathcal{R} = \arg\max_{\mathcal{S}_\mathcal{R}} \min_{\pi\in\Pi} \frac{F(\mathcal{S}_\mathcal{R}|\pi)}{F(\mathcal{S}^*_\pi|\pi)}$. This max-min ratio objective is used in risk-averse portfolio optimization and advertising [Ordentlich and Cover, 1998; Li and Yang, 2020].

Our Contribution. Our contributions stand as follows:

1. We introduce the problem of Robust Reward Placement (RRP) over a set of Markov Mobility Models, that has real-world applications across various domains.
2. We study the properties of RRP and show that it is **NP**-hard (Theorem 1). Due to the additivity and monotonicity properties of the reward function (Lemma 3), it admits an optimal solution in pseudo-polynomial time under a single setting, i.e. $|\Pi| = 1$ (Lemma 4), yet it is inapproximable when $|\Pi| > 1$ unless we exceed the budget constraint by a factor $\mathcal{O}(\ln|\Pi|)$ (Theorem 2).
3. We adopt techniques from robust influence maximization to develop Ψ-Saturate, a pseudo-polynomial time algorithm that finds a solution within ϵ distance of the optimal, i.e. OPT $-\ \epsilon$, while exceeding the budget constraint by an $\mathcal{O}(\ln {|\Pi|}/{\epsilon})$ factor (Lemma 6).
4. We present several heuristics as alternative solutions, most prominently one based on a dynamic programming algorithm for the max–min 0–1 KNAPSACK problem, to which RRP can be reduced (Lemma 5).

We corroborate our analysis with an experimental evaluation on synthetic and real data. Due to space constraints, we relegate some proofs to the full version [Petsinis *et al.*, 2024].

2 Related Work

The Robust Reward Placement problem relates to robust maximization of spread in a network, with some distinctive characteristics. Some works [Du *et al.*, 2013; He and Kempe, 2016; Chen *et al.*, 2016; Logins *et al.*, 2020; Logins *et al.*, 2022] study problems of selecting a seed set of nodes that robustly maximize the expected spread of a diffusion process over a network. However, in those models [Kempe *et al.*, 2003] the diffusion process is *generative*, whereby an item propagates in the network by producing unlimited replicas of itself. On the other hand, we study a *non-generative* spread function, whereby the goal is to reach as many as possible out of a population of network-resident agents. Our spread function is similar to the one studied in the problem of Geodemographic Influence Maximization [Zhang *et al.*, 2020], yet thereby the goal is to select a set of network locations that achieves high spread over a mobile population under a single environmental setting. We study the more challenging problem of achieving competitive spread in the worst case under uncertainty regarding the environment.

Several robust discrete optimization problems [Kouvelis and Yu, 1997] address uncertainty in decision-making by optimizing a max–min or min–max function under constraints. The robust MINIMUM STEINER TREE problem [Johnson *et al.*, 2000] seeks to minimize the worst-case cost of a tree that spans a graph; the min–max and min–max regret versions of the KNAPSACK problem [Aissi *et al.*, 2009] have a modular function as a budget constraint; other works examine the robust version of submodular functions [Krause and Golovin, 2014; He and Kempe, 2016] that describe several diffusion processes [Adiga *et al.*, 2014; Krause *et al.*, 2008]. To our knowledge, no prior work considers the objective of maximizing the worst-case ratio of an additive function over its optimal value subject to a knapsack budget constraint.

3 Preliminaries

Markov Mobility Model (MMM). We denote a discrete-time MMM as $\pi = (\mathcal{S}, \mathcal{I}, \mathcal{T}, \mathcal{M})$, where $\mathcal{S}$ is a set of n states, $\mathcal{I}$ is a vector of n elements in $[0, 1]$ expressing an initial probability distribution over states in $\mathcal{S}$, $\mathcal{T}$ is an $n \times n$ right-stochastic matrix, where $\mathcal{T}[s, s']$ is the probability of transition from state $s \in \mathcal{S}$ to another state $s' \in \mathcal{S}$, and $\mathcal{M}$ is an $n \times K$ matrix with elements in $[0, 1]$, where K is the maximum number of steps and $\mathcal{M}[s, k]$ expresses the cumulative probability that an agent starting from state $s \in \mathcal{S}$ takes $k' \in [k, K]$ steps. Remarkably, an MMM describes multiple agents and movements, whose starting positions are expressed via initial distribution $\mathcal{I}$ and their step-sizes via $\mathcal{M}$.

Rewards. Given an MMM, we select a set of states to be *reward states*. We use *a reward vector* $\mathcal{R} \in \{0, 1\}^n$ to indicate whether state $s \in \mathcal{S}$ is a *reward state* and denote the *set* of reward states as $\mathcal{S}_\mathcal{R} = \{s \in \mathcal{S}|\mathcal{R}[s] = 1\}$. In each timestamp t, an agent at state s moves to state s' and retrieves reward $\mathcal{R}[s']$. For a set of reward states $\mathcal{S}_\mathcal{R}$, and a given MMM π, the *cumulative reward* $F(\mathcal{S}_\mathcal{R}|\pi)$ of an agent equals:

$$F(\mathcal{S}_\mathcal{R}|\pi) = \sum_{k\in[K]} F_\pi(\mathcal{S}_\mathcal{R}|k) \quad (1)$$

$$F_\pi(\mathcal{S}_\mathcal{R}|k) = \mathcal{R}^\top \left(\mathcal{T}^k(\mathcal{I} \circ \mathcal{M}_k)\right), \quad (2)$$

where $F_\pi(\mathcal{S}_\mathcal{R}|k)$ is the expected reward at the k^{th} step, $\mathcal{M}_k$ is the k^{th} column of $\mathcal{M}$, and $\circ$ denotes the Hadamard product.

Connection to Pagerank. The Pagerank algorithm [Brin and Page, 1998], widely used in recommendation systems, computes the stationary probability distribution of a random walker in a network. The Pagerank scores are efficiently computed via power-iteration method [von Mises and Pollaczek-Geiringer, 1929]. Let $\mathbf{PR}$ be an $N \times 1$ column-vector of the Pagerank probability scores, initialized as $\mathbf{PR}(0)$, $\mathbf{T}$ is an $N \times N$ matrix featuring the transition probabilities of walker, and $\mathbf{1}$ be the all-ones vector. For a damping factor a, the power method computes the scores in iterations as:

$$\mathbf{PR}(t) = a \cdot \mathbf{T} \cdot \mathbf{PR}(t-1) + \frac{1-a}{N}\mathbf{1}. \quad (3)$$

We repeat this process until convergence, i.e., until $|\mathbf{PR}(t) - \mathbf{PR}(t-1)| \leq \epsilon$ for a small $\epsilon \geq 0$. We denote the PageRank score at the i^{th} node as $\mathbf{PR}[i]$. For a sufficiently large number of steps K for each state with $\mathcal{M}_k = \mathbf{1}\, \forall k \in [K]$, Equation (2) becomes $F_\pi(k) = \mathcal{R}^\top \left(\mathcal{T}^k\mathcal{I}\right)$. Likewise, for damping factor $a = 1$, Equation (3) becomes $\mathbf{PR}(t) = \mathbf{T}^t\mathbf{PR}(0)$, thus the two equations are rendered analogous with $\mathbf{T} = \mathcal{T}$ and $\mathbf{PR}(0) = \mathcal{I}$. Then, considering that the iteration converges from step $\hat{k}$ onward, the expected reward from reward state s_i per step $k \geq \hat{k}$, $F_\pi(\{s_i\}|k)$, is the PageRank score of the i^{th} node, that is $\mathbf{PR}[i]$. To see this, let $\mathcal{R}_i = \mathbf{1}_i$ be the reward vector when $s_i \in \mathcal{S}$ is the *only* reward state; then it holds that $\mathbf{PR}[i] = \mathbf{1}_i^\top \left(\mathbf{T}^k\mathbf{PR}(0)\right) = \mathcal{R}_i^\top \left(\mathcal{T}^k\mathcal{I}\right) = F_\pi(\{s_i\}|k)$.

4 Problem Formulation

In this section we model the uncertain environment where individuals navigate and introduce the Robust Reward Placement (RRP) problem over a set of Markov Mobility Models (MMMs), extracted from real movement data, that express the behavior of individuals under different settings.

Setting. Many applications generate data on the point-to-point movements of agents over a network, along with a distribution and their total number of steps. Using aggregate statistics on this information, we formulate, *without loss of generality*, the movement of a population by a single agent moving probabilistically over the states of an MMM $\pi = (\mathcal{S}, \mathcal{I}, \mathcal{T}, \mathcal{M})$. Due to environment uncertainty, the agent may follow any of $|\Pi|$ different settings[1] $\Pi = \{\pi_1, \pi_2, \dots, \pi_{|\Pi|}\}$.

Robust Reward Placement Problem. Several resource allocation problems can be formulated as optimization problems over an MMM π, where reward states $\mathcal{S}_\mathcal{R}$ correspond to the placement of resources. Given a budget L and a cost function $c : \mathcal{S} \to \mathbb{N}^+$, the Reward Placement (RP) problem seeks a set of reward states $\mathcal{S}^*_\mathcal{R} \subseteq \mathcal{S}$ that maximizes the cumulative reward $F(\mathcal{S}^*_\mathcal{R}|\pi)$ obtained by an agent, that is:

$$\mathcal{S}^*_\mathcal{R} = \arg\max_{\mathcal{S}_\mathcal{R}} F(\mathcal{S}_\mathcal{R}|\pi) \quad \text{s.t.} \sum_{s \in \mathcal{S}_\mathcal{R}} c[s] \le L.$$

However, in reality the *agent's* movements follow an unknown distribution sampled from a set of settings $\Pi = \{\pi_1, \pi_2, \dots, \pi_{|\Pi|}\}$ represented as different MMMs. Under this uncertainty, the Robust Reward Placement (RRP) problem seeks a set of reward states $\mathcal{S}_\mathcal{R}$, within a budget, that maximizes the worst-case ratio of agent's cumulative reward over the optimal one, when the model $\pi \in \Pi$ is *unknown*. Formally, we seek a reward placement $\mathcal{S}^*_\mathcal{R} \subseteq \mathcal{S}$ such that:

$$\mathcal{S}^*_\mathcal{R} = \arg\max_{\mathcal{S}_\mathcal{R}} \min_{\pi \in \Pi} \frac{F(\mathcal{S}_\mathcal{R}|\pi)}{F(\mathcal{S}^*_\pi|\pi)} \quad \text{s.t.} \sum_{s \in \mathcal{S}_\mathcal{R}} c[s] \le L, \tag{4}$$

where $\mathcal{S}^*_\pi = \arg\max_{\mathcal{S}_\mathcal{R}} F(\mathcal{S}_\mathcal{R}|\pi)$ is the optimal reward placement for a given model $\pi \in \Pi$ within budget L. This formulation is equivalent to minimizing the maximum *regret ratio* of $F(\mathcal{S}_\mathcal{R}|\pi)$, i.e., $1 - \frac{F(\mathcal{S}_\mathcal{R}|\pi)}{F(\mathcal{S}^*_\pi|\pi)}$. The motivation arises from the fact that stakeholders are prone to compare what they achieve with what they could optimally achieve. The solution may also be seen as the optimal placement when the model $\pi \in \Pi$ in which agents are moving is chosen by an *omniscient* adversary, i.e. an adversary who chooses the setting π after observing the set of reward states $\mathcal{S}_\mathcal{R}$.

5 Hardness and Inapproximability Results

In this section we examine the optimization problem of RRP and we show that is **NP**-hard in general. First, in Theorem 1 we prove that even for a single model ($|\Pi| = 1$) the optimal solution cannot be found in polynomial time, due to a reduction from the 0–1 KNAPSACK problem [Karp, 1972].

Theorem 1. *The* RRP *problem is* **NP***-hard even for a single model, that is* $|\Pi| = 1$.

[1] We use the terms 'setting' and 'model' interchangeably.

Proof. In the 0–1 KNAPSACK problem [Karp, 1972] we are given a set of items U, each item $u \in U$ having a cost $c(u)$ and, wlog, an *integer* value $F(u)$ and seek a subset $V \subseteq U$ that has total cost $\sum_{v \in V} c(v)$ no more than a given budget L and maximum total value $\sum_{v \in V} F(v)$. In order to reduce 0–1 KNAPSACK to RRP, we set a distinct state $s \in \mathcal{S}$ for each item $u \in U$ with the same cost, i.e., $\mathcal{S} = U$, assign to each state a self-loop with transition probability 1, let each state be a reward state, and set a uniform initial distribution of agents over states equal to $1/|\mathcal{S}|$ and steps probability equal to $\mathcal{M}[s, k] = 1$, $\forall k \in [1, \dots, F(u)]$. For a single setting, an optimal solution to the RRP problem of Equation (4) is also optimal for the **NP**-hard 0–1 KNAPSACK problem. □

Theorem 2 proves that RRP is inapproximable in polynomial time within constant factor, by a reduction from the HITTING SET problem, unless we exceed the budget constraint.

Theorem 2. *Given a budget L and set of models Π, it is* **NP***-hard to approximate the optimal solution to* RRP *within a factor of $\Omega(1/n^{1-\epsilon})$, for any constant $\epsilon > 0$, unless the cost of the solution is at least βL, with $\beta \ge \ln |\Pi|$.*

Proof. We reduce the HITTING SET problem [Karp, 1972] to RRP and show that an approximation algorithm for RRP implies one for HITTING SET. In the HITTING SET problem, given a collection of X items, $C = \{c_1, c_2, \dots, c_X\}$ and a set of M subsets thereof, $B_i \subseteq C$, $i \in \{1, \dots, M\}$, we seek a *hitting set* $C' \subseteq C$ such that $B_i \cap C' \neq \emptyset \ \forall i \in \{1, \dots, M\}$.

Given an instance of HITTING SET, we reduce it to RRP as follows. For each subset B_i we set a state $s^l_i \in \mathcal{S}^l$ and for each item c_i we set a state $s^r_i \in \mathcal{S}^r$. Aslo, for each subset B_i we set an MMM π_i ($|\Pi| = M$) over the same set of states $\mathcal{S} = \mathcal{S}^l \cup \mathcal{S}^r$ with $\mathcal{S}^l \cap \mathcal{S}^r = \emptyset$. We set the initial probabilities $\mathcal{I}$ as uniform for all states in $\mathcal{S}^l$, equal to $1/|\mathcal{S}^l|$ for all models. Each model $\pi_i \in \Pi$ features transition probabilities 1 from each state s^l_j to state s^l_i, with $i \neq j$, and uniform transition probabilities from s^l_i to each state s^r_j if and only if $c_j \in B_i$. States in $\mathcal{S}^r$ are *absorbing*, i.e., each state has a self-loop with probability 1. Figure 2 shows a small example of a HITTING SET instance and its RRP equivalent. We set the cost for absorbing states in $\mathcal{S}^r$ to 1 and let each node in $\mathcal{S}^l$ have a cost exceeding L. By this construction, if the reward placement $\mathcal{S}_\mathcal{R}$ does not form a *hitting set*, then there exists at least one subset B_i, such that $B_i \cap \mathcal{S}_\mathcal{R} = \emptyset$, hence $\min_\pi \frac{F(\mathcal{S}_\mathcal{R}|\pi)}{F(\mathcal{S}^*_\pi|\pi)} = 0$. In reverse, if $\mathcal{S}_\mathcal{R}$ forms a hitting set, it holds that $\min_\pi \frac{F(\mathcal{S}_\mathcal{R}|\pi)}{F(\mathcal{S}^*_\pi|\pi)} \ge \frac{1}{|\mathcal{S}^r|} > 0$. Thus, a hitting set exists if and only if $\min_\pi \frac{F(\mathcal{S}_\mathcal{R}|\pi)}{F(\mathcal{S}^*_\pi|\pi)} > 0$. In effect, if we

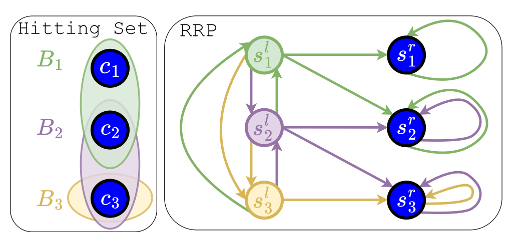

Figure 2: HITTING SET (left) and RRP reduction (right).

obtained an approximation algorithm for RRP by increasing the budget to βL, for $\beta > 1$, then we would also approximate, with a budget increased by a factor of β, the HITTING SET problem, which is **NP**-hard for $\beta < (1-\delta)\ln|\Pi|$ and $\delta > 0$ [Dinur and Steurer, 2014]. □

6 Connections to Knapsack Problems

In this section, we establish connections between RRP and KNAPSACK problems, which are useful in our solutions.

Monotonicity and Additivity. Lemma 3 establishes that the cumulative reward function $F(\mathcal{S}_\mathcal{R}|\pi)$ is monotone and additive with respect to $\mathcal{S}_\mathcal{R}$. These properties are vital in evaluating $F(\mathcal{S}_\mathcal{R}|\pi)$ while exploiting pre-computations.

Lemma 3. *The cumulative reward $F(\mathcal{S}_\mathcal{R}|\pi)$ in Equation* (1) *is a monotone and additive function of reward states $\mathcal{S}_\mathcal{R}$.*

Proof. By Equation (1) we obtain the monotonicity property of the cumulative reward function $F(\cdot|\pi)$. Given a model $\pi \in \Pi$ and two sets of reward states $\mathcal{A} \subseteq \mathcal{B} \subseteq \mathcal{S}$ every term of $F(\mathcal{A}|\pi)$ is no less than its corresponding term of $F(\mathcal{B}|\pi)$ due to Equation (2). For the *additivity* property it suffices to show that any two sets of reward states $\mathcal{A}, \mathcal{B} \subseteq \mathcal{S}$ satisfy:

$$F(\mathcal{A}|\pi) + F(\mathcal{B}|\pi) = F(\mathcal{A} \cup \mathcal{B}|\pi) + F(\mathcal{A} \cap \mathcal{B}|\pi).$$

Assume w.l.o.g. that the equality holds at time t, i.e. $r^t_\mathcal{A} + r^t_\mathcal{B} = r^t_{\mathcal{A}\cap\mathcal{B}} + r^t_{\mathcal{A}\cup\mathcal{B}}$, $r^t_\mathcal{X}$ being the cumulative reward at time t for reward states $\mathcal{X}$. It suffices to prove that the additivity property holds for $t+1$. At timestamp $t+1$, the agent at state $s \in \mathcal{S}$ moves to $s' \in \mathcal{S}$. We distinguish cases as follows:

1. If $s' \notin \mathcal{A} \cup \mathcal{B}$ then $s' \notin \mathcal{A} \cap \mathcal{B}$, $s' \notin \mathcal{A}$ and $s' \notin \mathcal{B}$, thus additivity holds.
2. If $s' \in \mathcal{A} \cup \mathcal{B}$ and $s' \notin \mathcal{A} \cap \mathcal{B}$ then either $s' \in \mathcal{A}$ or $s' \in \mathcal{B}$. Assume wlog that $s' \in \mathcal{A}$, then it holds that: $r^{t+1}_\mathcal{A} = r^t_\mathcal{A} + \mathcal{T}[s,s']$, $r^{t+1}_{\mathcal{A}\cup\mathcal{B}} = r^t_{\mathcal{A}\cup\mathcal{B}} + \mathcal{T}[s,s']$, $r^{t+1}_\mathcal{B} = r^t_\mathcal{B}$ and $r^{t+1}_{\mathcal{A}\cap\mathcal{B}} = r^t_{\mathcal{A}\cap\mathcal{B}}$.
3. If $s' \in \mathcal{A} \cap \mathcal{B}$ then $s' \in \mathcal{A}$ and $s' \in \mathcal{B}$. Then, it holds that: $r^{t+1}_\mathcal{A} = r^t_\mathcal{A} + \mathcal{T}[s,s']$, $r^{t+1}_\mathcal{B} = r^t_\mathcal{B} + \mathcal{T}[s,s']$, $r^{t+1}_{\mathcal{A}\cup\mathcal{B}} = r^t_{\mathcal{A}\cup\mathcal{B}} + \mathcal{T}[s,s']$, and $r^{t+1}_{\mathcal{A}\cap\mathcal{B}} = r^t_{\mathcal{A}\cap\mathcal{B}} + \mathcal{T}[s,s']$.

In all cases the cumulative reward function is additive. □

Next, Lemma 4 states that RRP under a single model π ($|\Pi| = 1$), i.e., the maximization of $F(\mathcal{S}_\mathcal{R}|\pi)$ within a budget L, is solved in pseudo-polynomial time thanks to the additivity property in Lemma 3 and a reduction from the 0–1 KNAPSACK problem [Karp, 1972]. Lemma 4 also implies that we can find the optimal reward placement with the maximum expected reward by using a single expected setting π.

Lemma 4. *For a single model π ($|\Pi| = 1$) and a budget L, there is an optimal solution for* RRP *that runs in pseudo-polynomial time $\mathcal{O}(Ln)$.*

Proof. For each state $s_i \in \mathcal{S}$ we set an item $u_i \in U$ with cost $c(u_i) = c[s_i]$ and value $F(u_i) = F(\{s_i\}|\pi)$. Since the reward function is additive (Lemma 3), it holds that $F(\mathcal{S}_\mathcal{R}|\pi) = \sum_{s_i \in \mathcal{S}_\mathcal{R}} F(\{s_i\}|\pi) = \sum_{u_i \in U} F(u_i)$. Thus, we can optimally solve single setting RRP in pseudo-polynomial time by using the dynamic programming solution for 0–1 KNAPSACK [Martello and Toth, 1987]. □

In the MAX–MIN 0–1 KNAPSACK problem (MNK), given a set of items U, each item $u \in U$ having a cost $c(u)$, and a collection of scenarios X, each scenario $x \in X$ having a value $F_x(u)$, we aim to determine a subset $V \subseteq U$, with total cost no more than L, and maximizes the minimum total value across scenarios, i.e., $\arg_V \max \min_x \sum_{u \in V} F_x(u)$. The following lemma reduces the RRP problem to MAX–MIN 0–1 KNAPSACK [Yu, 1996] in pseudo-polynomial time.

Lemma 5. RRP *is reducible to* MAX–MIN 0–1 KNAPSACK *in $\mathcal{O}(|\Pi|Ln)$ time.*

7 Approximation Algorithm

Here, we introduce Ψ-Saturate,[2] a pseudo-polynomial time binary-search algorithm based on the Greedy-Saturate method [He and Kempe, 2016]. For any $\epsilon > 0$, Ψ-Saturate returns an ϵ-additive approximation of the optimal solution by exceeding the budget constraint by a factor $\mathcal{O}(\ln|\Pi|/\epsilon)$.

Algorithm 1 Ψ-Saturate Algorithm

Input: MMMs Π, steps K, budget L, precision ϵ, param. β.
Output: Reward Placement $\mathcal{S}^*_\mathcal{R}$ of cost at most βL.

1: **for** $\pi \in \Pi$ **do**
2: $\quad \mathcal{S}^*_\pi \leftarrow$ Knapsack(π, L)
3: **end for**
4: $\eta_{min} \leftarrow 0, \eta_{max} \leftarrow 1, \mathcal{S}^*_\mathcal{R} \leftarrow \emptyset$
5: **while** $(\eta_{min} - \eta_{max}) \geq \epsilon$ **do**
6: $\quad \eta \leftarrow (\eta_{max} + \eta_{min})/2$
7: $\quad \mathcal{S}_\mathcal{R} \leftarrow \emptyset$
8: $\quad$ **while** $\sum_{\pi\in\Pi} \min\left(\eta, \frac{F(\mathcal{S}_\mathcal{R}|\pi)}{F(\mathcal{S}^*_\pi|\pi)}\right) < (\eta \cdot |\Pi| - \eta \cdot \epsilon/3)$ **do**
9: $\quad\quad s \leftarrow \arg\max_{s \in \mathcal{S}\setminus\mathcal{S}_\mathcal{R}} \sum_{\pi\in\Pi} \frac{1}{c(s)} \left(\min\left(\eta, \frac{F(\mathcal{S}_\mathcal{R}\cup\{s\}|\pi)}{F(\mathcal{S}^*_\pi|\pi)}\right) - \min\left(\eta, \frac{F(\mathcal{S}_\mathcal{R}|\pi)}{F(\mathcal{S}^*_\pi|\pi)}\right)\right)$
10: $\quad\quad \mathcal{S}_\mathcal{R} \leftarrow \mathcal{S}_\mathcal{R} \cup \{s\}$
11: $\quad$ **end while**
12: $\quad$ **if** $\sum_{s\in\mathcal{S}_\mathcal{R}} c[s] > \beta L$ **then**
13: $\quad\quad \eta_{max} \leftarrow \eta$
14: $\quad$ **else**
15: $\quad\quad \eta_{min} \leftarrow \eta \cdot (1 - \epsilon/3)$
16: $\quad\quad \mathcal{S}^*_\mathcal{R} \leftarrow \mathcal{S}_\mathcal{R}$
17: $\quad$ **end if**
18: **end while**
19: **return** $\mathcal{S}^*_\mathcal{R}$

The Ψ-Saturate Algorithm. Algorithm 1 presents the pseudocode of Ψ-Saturate. As a first step, in Lines 1–2, the algorithm finds the optimal reward placement $\mathcal{S}^*_\pi$ for each model $\pi \in \Pi$; this is needed for evaluating the denominator of the RRP objective value in Equation (4). By Lemma 4, $\mathcal{S}^*_\pi$ is computed in pseudo-polynomial time using the dynamic programming algorithm for the KNAPSACK problem. Then, in Lines 5–18 the algorithm executes a binary search in the range of the min–max objective ratio (Line 4). In each iteration, the algorithm makes a guess η of the optimal min–max

[2] Ψ for 'pseudo-', from Greek 'ψευδής'.

objective value (Line 6), and then seek a set of reward states $\mathcal{S}_\mathcal{R}$ (Line 7), of minimum cost, with score at least η (Line 8), within distance $\epsilon > 0$. Finding $\mathcal{S}_\mathcal{R}$ of the minimum cost, implies an optimal solution for the NP-hard RRP problem. Thus, in Lines 9–10, Ψ-Saturate approximates this solution by using the Greedy algorithm in [Wolsey, 1982] for function $\min\left(\eta, \frac{F(\mathcal{S}_\mathcal{R}|\pi)}{F(\mathcal{S}^*_\pi|\pi)}\right)$ which, for fixed π and η, is monotone and submodular.[3] If the formed solution exceeds the budget constraint, the algorithm decreases the upper bound of the search scope (Lines 12–13), otherwise it increases the lower bound and updates the optimal solution $\mathcal{S}^*_\mathcal{R}$ (Lines 14–16). Finally, it returns the optimal solution found (Line 19).

In Lemma 6 we prove that by setting $\beta = 1 + \ln \frac{3|\Pi|}{\epsilon}$, Ψ-Saturate approximates the optimal value within distance ϵ.

Lemma 6. *For any constant $\epsilon > 0$, let $\beta = 1 + \ln \frac{3|\Pi|}{\epsilon}$. Ψ-Saturate finds a reward placement $\mathcal{S}_\mathcal{R}$ of cost at most βL with $\min_\pi \frac{F(\mathcal{S}_\mathcal{R}|\pi)}{F(\mathcal{S}^*_\pi|\pi)} \geq \min_\pi \frac{F(\mathcal{S}^*_\mathcal{R}|\pi)}{F(\mathcal{S}^*_\pi|\pi)} - \epsilon = OPT - \epsilon$, and $\mathcal{S}^*_\mathcal{R} = \arg_{\mathcal{S}_\mathcal{R}} \max \min_\pi \frac{F(\mathcal{S}_\mathcal{R}|\pi)}{F(\mathcal{S}^*_\pi|\pi)}$ s.t. $\sum_{s\in\mathcal{S}_\mathcal{R}} c[s] \leq L$.*

Proof. We seek to solve a max–min regret optimization problem of an *additive function under a knapsack constraint*. While finding the optimal score in the denominator of the max–min ratio is **NP**-hard due to Theorem 1, in Line 2 we evaluate it by a pseudo-polynomial time Knapsack algorithm, as Lemma 4 allows. In Lines 5–18, we perform a binary search to find a reward placement of cost at most βL. By the analysis in [He and Kempe, 2016], the Ψ-Saturate algorithm provides an $(\beta, \mathrm{OPT} - \epsilon)$ bicriteria approximation for RRP, where OPT is the optimal objective ratio score. □

Unlike the pseudo-polynomial-time dynamic programming algorithm (Knapsack, Line 2) we employ, the Greedy-Saturate algorithm [He and Kempe, 2016] uses the Greedy[4] algorithm to *approximate* the optimal reward placement $\mathcal{S}^*_\pi$ (Lines 1–2), which provides an 1/2-approximation of the optimal solution for a monotone additive function over a knapsack constraint [Garey and Johnson, 1979]. As our reward function is monotone and additive (Lemma 3), Greedy-Saturate offers an $(\frac{1}{2}\mathrm{OPT} - \epsilon, \beta L)$ bicriteria approximation.

Notably, for $\beta = 1$, Ψ-Saturate returns an non-constant approximation of the optimal solution within the budget constraint L. In particular, the next corollary holds.

Corollary 7. *For any constant $\epsilon > 0$, let $\gamma = 1+\ln \frac{3|\Pi|}{\epsilon}$. For $\beta = 1$, Ψ-Saturate satisfies the budget constraint and returns an $\frac{1}{\gamma}(OPT' - \epsilon)$ approximation factor of the optimal solution, with $OPT' = \max_{\mathcal{S}_\mathcal{R}} \min_\pi \frac{F(\mathcal{S}_\mathcal{R}|\pi)}{F(\mathcal{S}^*_\pi|\pi)}$ s.t. $\sum_{s\in\mathcal{S}_\mathcal{R}} c[s] \leq \frac{L}{\gamma}$.*

We stress that the approximation in Corollary 7 is non-constant and can be arbitrarily small, as implied by the inapproximability result of Theorem 2.

[3] The minimum of a constant function (η) and a monotone additive function $\left(\frac{F(\mathcal{S}_\mathcal{R}|\pi)}{F(\mathcal{S}^*_\pi|\pi)}, \text{Lemma 3}\right)$ is monotone and submodular. The term $F(\mathcal{S}^*_\pi|\pi)$ is constant as it has been computed in Line 2.

[4] The algorithm iteratively selects the element, within the budget, that offers the maximal marginal gain divided by its cost.

8 Heuristic Solutions

Inspired from previous works on node selection in networks [He and Kempe, 2016; Zhang *et al.*, 2020] and the connection of RRP with Knapsack problems, we propose four heuristic methods. For a single model ($|\Pi| = 1$) and under uniform costs ($c[s] = c\ \forall s \in \mathcal{S}$), these four heuristics find an optimal solution. However, contrary to Ψ-Saturate algorithm (Lemma 6), they may perform arbitrarily badly in the general multi-model case, even by exceeding the budget constraint. To accelerate the selection process, we use the *Lazy Greedy* technique that updates values selectively [Minoux, 1978] in all heuristics, except the one using dynamic programming.

All Greedy. The All Greedy method optimally solves the RRP problem for each model $\pi \in \Pi$ separately using the Knapsack dynamic programming algorithm (Lemma 4) and then picks, among the collected solutions, the one yielding the best value of the objective in Equation (4). All Greedy is optimal for a single model with an arbitrary cost function.

Myopic. A greedy algorithm that iteratively chooses the reward state $s^* \in \mathcal{S}$, within the budget, that offers the maximal marginal gain ratio to the RRP objective divided by the cost, that is $s^* = \arg\max_{s\in\mathcal{S}\setminus\mathcal{S}_\mathcal{R}} \min_{\pi\in\Pi} \left(\frac{1}{c[s]} \frac{F(\mathcal{S}_\mathcal{R}\cup\{s\}|\pi) - F(\mathcal{S}_\mathcal{R}|\pi)}{F(\mathcal{S}^*_\pi|\pi)}\right)$.

Best-Worst Search (BWS). This algorithm uses as a score the minimum, over settings, cumulative reward for a set $\mathcal{S}_\mathcal{R}$, that is $H(\mathcal{S}_\mathcal{R}) = \min_\pi F(\mathcal{S}_\mathcal{R}|\pi)$ and iteratively chooses the reward state $s^* \in \mathcal{S}$, within the budget, that offers the maximal marginal gain to that score divided by the cost, that is $s^* = \arg\max_{s\in\mathcal{S}\setminus\mathcal{S}_\mathcal{R}} \left(\frac{H(\mathcal{S}_\mathcal{R}\cup\{s\}) - H(\mathcal{S}_\mathcal{R})}{c[s]}\right)$.

Dynamic Programming (DP-RRP). In Lemma 5 we reduced RRP to MAX–MIN 0–1 KNAPSACK (MNK) in pseudo-polynomial time. While MNK admits an optimal solution using a pseudo-polynomial time dynamic programming algorithm, its running time grows exponentially with the number of settings $|\Pi|$ [Yu, 1996]. To overcome this time overhead, we propose a more efficient albeit *non-optimal* dynamic-programming algorithm for the RRP problem, noted as DP-RRP. For reward placement $\mathcal{S}_\mathcal{R}$, we denote the cumulative reward for each setting as the following $|\Pi|$-tuple: $g(\mathcal{S}_\mathcal{R}) = \left(F(\mathcal{S}_\mathcal{R}|\pi_1), F(\mathcal{S}_\mathcal{R}|\pi_2), \ldots, F(\mathcal{S}_\mathcal{R}|\pi_{|\Pi|})\right)$. We use an $(n + 1) \times (L + 1)$ matrix M whose entries are $|\Pi|$-tuples of the form $g(\cdot)$. Let $\min g(\mathcal{S}_\mathcal{R}) = \min_{\pi_i} F(\mathcal{S}_\mathcal{R}|\pi_i)$ be the minimum reward, across $|\Pi|$ settings. We define the maximum of two entries $g(\mathcal{S}_{R_1})$ and $g(\mathcal{S}_{R_2})$, as $\arg\max_{\mathcal{S}_\mathcal{R}\in\{\mathcal{S}_{R_1}, \mathcal{S}_{R_2}\}} \min g(\mathcal{S}_\mathcal{R})$, i.e. the one holding the largest minimum reward. We initialize $M[\cdot, 0] = M[0, \cdot] = (0, 0, \ldots, 0)$ and recursively compute $M[i, j]$ as follows:

$$M[i, j] = \max\{M[i-1, j], M[i-1, j-c[i]] + g(\{i\})\}, \quad (5)$$

where $M[i, j]$ stands for a solution using the first i states, by some arbitrary order, and j units of budget. In the recursion of Equation (5), the first option stands for *not* choosing state s_i as a reward state, while the latter option stands for doing so while paying cost $c[i]$ and gaining the additive reward $g(\{i\})$. We compute $M[n, L]$ as above in space and time complexity $\Theta(|\Pi|Ln)$ and backtrack over M to retrieve the selected reward states in the final solution. Note that, for a single model, i.e. $|\Pi| = 1$ and arbitrary cost function, Equation (5) returns an optimal solution.

Worst-case performance. While all heuristics[5] approach the optimal solution under a single setting, they may perform arbitrarily badly with multiple settings. In Lemma 8 we prove that this holds even when exceeding the budget constraint, contrariwise to the Ψ-Saturate algorithm (Lemma 6).

Lemma 8. *The heuristics for* RRP *may perform arbitrarily badly even when they exceed the budget constraint from L to βL, with $\beta = 1 + \ln \frac{3|\Pi|}{\epsilon}$ and $\epsilon > 0$.*

9 Experimental Analysis

In this section we evaluate the running time and performance of algorithms on synthetic and real-world data. We use different problem parameters as Table 1 shows, marking the default value of each parameter in bold. To satisfy the budget constraint for the Ψ-Saturate algorithm, we fix $\beta = 1$ as in Corollary 7 and set precision to $\epsilon = (|\Pi| \cdot 10^3)^{-1}$. We set the budget L as a percentage of the total cost $\sum_{s \in \mathcal{S}} c[s]$. To benefit from the additivity property of Lemma 3, we precompute the cumulative reward $F(\{s\}|\pi)$ for each state $s \in \mathcal{S}$ and model $\pi \in \Pi$. We implemented[6] all methods in C++ 17 and ran experiments on a 376GB server with 96 CPUs @2.6GHz.

Parameter	Values
n	2500, 5000, 7500, **10000**, 12500
$\langle d \rangle$	3,**6**,9,12
p_β	0.6, 0.7, **0.8**, 0.9
$\lvert\Pi\rvert$	2,5,**10**,15,20
K	2,4,**6**,8,10
L	10%, **25%**, 50%, 75%

Table 1: Parameter settings.

9.1 Synthetic Data

We use two different types of synthetic datasets to represent stochastic networks (i.e., MMMs). In each type, we generate a directed graph and then sample edge weights from a normal distribution to create different settings. In more detail:

Erdős-Rényi: We generate 20 directed graphs for each of the 5 sizes shown in Table 1. In all cases, we set the edge creation probability to achieve the desired average in-degree $\langle d \rangle$.

Scale-Free: We generate 20 directed scale-free graphs for each of the 5 sizes shown in Table 1. Following [Bollobás *et al.*, 2003], we use three parameters to construct the network: p_α (p_γ), the probability to add a new node connected to an existing one chosen randomly by its in-degree (out-degree), and p_β, the probability to add an edge (u, v), with u and v selected by their out-degree and in-degree respectively. In all datasets, we tune p_β and set $p_\gamma = \frac{1-p_\beta}{3}$ and $p_\alpha = 2p_\gamma$, so that $p_\alpha + p_\beta + p_\gamma = 1$.

[5]All algorithms work, without modification, with rewards of arbitrary non-negative values and when a partial solution is given.

[6]https://anonymous.4open.science/r/RRP-F6CA

Given a graph structure, we generate $|\Pi| = 20$ distinct settings, corresponding to different models. For each setting π_i, we sample the weight of edge (u, v) from a normal distribution with mean $\mu = {}^1/_{d_u}$ and standard deviation $\sigma_i = {}^i/_{10d_u}$. When we sample a negative value, we set the edge weight to zero. In each resulting directed graph, we set transition probabilities $\mathcal{T}$ as normalized edge weights. Moreover, we set the initial probabilities $\mathcal{I}$ proportionally to the sum of nodes' outgoing weights and the cost of a node as the rounded-down average number of its in-neighbors among settings.

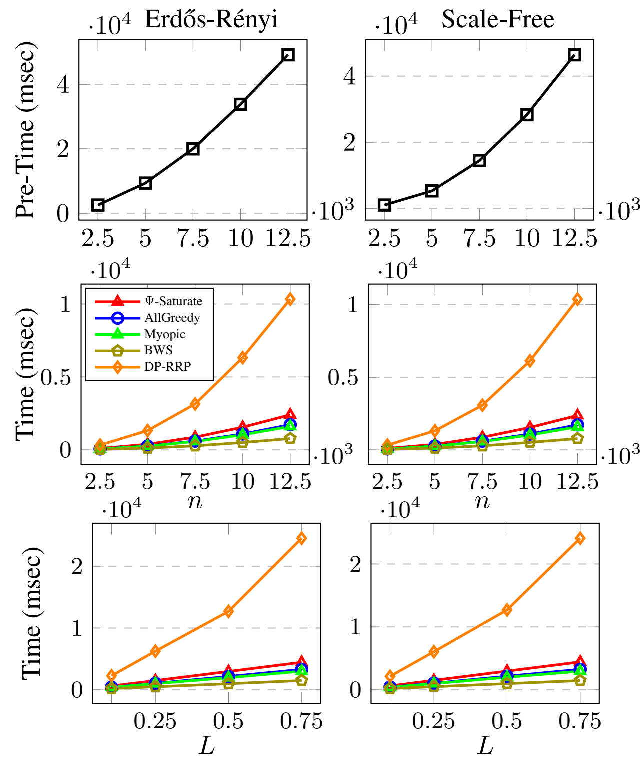

Figure 3: Preprocessing and running time vs. n, L for Erdős-Rényi (left) and Scale-Free (right) datasets.

Time efficiency. Figure 3 plots the average (over 20 graphs) preprocessing time vs. graph size and running time for all algorithms vs. graph size and budget. Notably, the precomputation takes time superlinear in graph size n, as the time complexity needed for the power iteration is $\mathcal{O}(K(n + m))$ for K steps maximum and m edges. The runtime of most algorithms grows linearly with graph size and budget, indicating their efficiency, except of DP-RRP, whose time complexity is at least quadratic in n, $\Theta(|\Pi|Ln)$ while $L = \Omega(n)$.

Reward placement robustness. Figure 4 illustrates the algorithms' average performance (over 20 graphs) in the reward placement robustness objective as several parameters vary. On the Erdős-Rényi data, Ψ-Saturate outperforms all heuristics vs. graph size n and in other measurements, while on Scale-Free data, DP-RRP performs best overall. With a single setting, i.e., when $|\Pi| = 1$, all heuristics find an almost optimal solution. However, as expected, performance

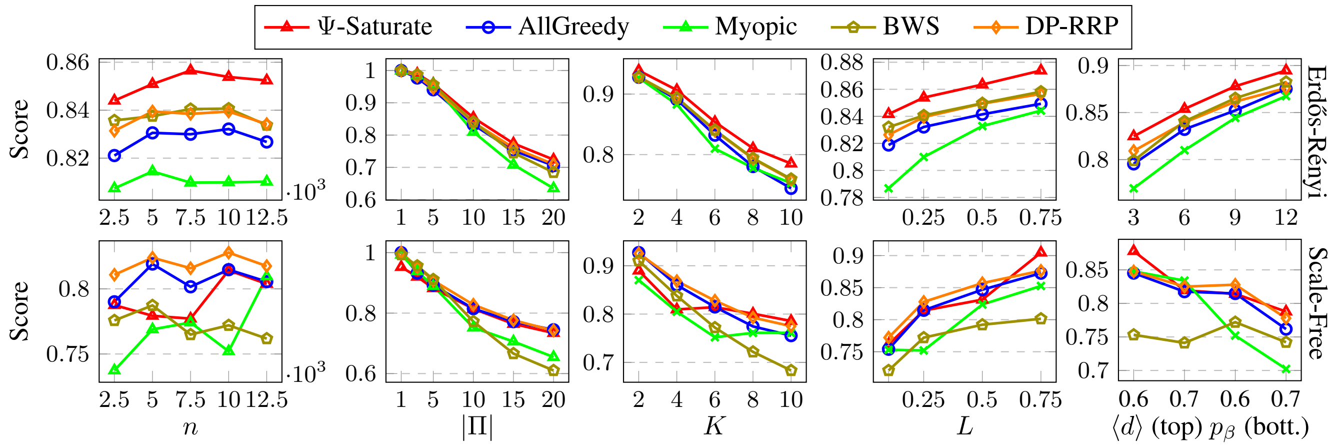

Figure 4: Reward placement robustness scores on Erdős-Rényi (top) and Scale-Free (bottom) datasets.

decreases as the number of models $|\Pi|$ grows, whence the adversary possesses a larger pool of models to select from. Similarly, as the number of steps K grows, the feasible agent movements expand, causing the optimal cumulative reward per setting to rise more than the worst-case reward in general, hence the robustness score falls. In contrast, the score of all algorithms rises with budget L. Intuitively, a higher budget offers more flexibility to hedge against worst-case outcomes, hence better robustness scores. This effect is more evident on the Scale-Free dataset, which has fewer lucrative nodes of high in-degree. The growth of robustness scores vs. $\langle d \rangle$ on Erdős-Rényi data confirms the importance of in-degree. On the other hand, the growth of p_β results in Scale-Free networks with more skewed power-law in-degree distributions, on which robustness scores suffer.

9.2 Real Data

To further validate our observations, we create graphs using real-world movement data. We gathered movement records from Baidu Map, covering Xuanwu District in Nanjing[7] from July 2019 to September 2019; these records comprise sequential Points of Interest (POIs) with timestamps, allowing us to calculate the probability of transitioning between POIs based on the Markovian assumption. Using these probabilities, we construct graphs where nodes represent POIs and edges express transition probabilities. Each graph captures a 7-day period, resulting in a total of 13 graphs. The combined dataset features a total of 51 943 different nodes. Out of practical considerations, we assign data-driven costs to POIs based on their visit frequency and a fixed value: $c[x] = \lfloor \mathsf{frequency}(x)/25 + 50 \rfloor$. The initial and steps probabilities follow the same default setup as the synthetic datasets.

Time and Performance. Preprecessing the Xuanwu dataset where $n = 51\,943$ and $|\Pi| = 13$ takes 118 seconds. Figure 5 shows how the running time and robustness scores vary as the budget grows. DP-RRP is the most time-consuming, followed by Ψ-Saturate, while BWS emerges as the most time-efficient solution. Interestingly, the robustness

[7]https://en.wikipedia.org/wiki/Nanjing

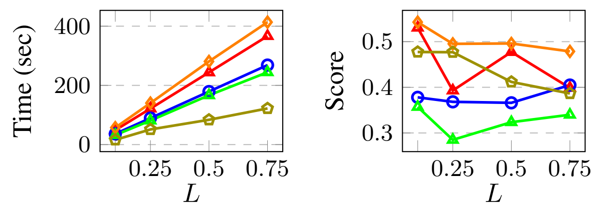

Figure 5: Time and robustness score vs. L on the Xuanwu dataset.

score does not follow a clear upward trend vs. budget with these real-world data; after all, the objective of Equation (4) is not a monotonic function of budget; while a higher budget allows for more flexibility in allocating resources, it also allows the optimal reward to grow correspondingly. Nevertheless, DP-RRP consistently outperforms all algorithms across budget values, corroborating its capacity to uncover high quality solutions even in hard scenarios, even while the performance of Ψ-Saturate and other heuristics fluctuates.

10 Conclusions

We introduced the **NP**-hard problem of Robust Reward Placement (RRP). Assuming an agent is moving on an unknown Markov Mobility Model (MMM), sampled by a set of Π candidates, RRP calls to select reward states within a budget that maximize the worst-case ratio of the expected reward (agent visits) over the optimal one. Having shown that RRP is strongly inapproximable, we propose Ψ-Saturate, an algorithm that achieves an ϵ-additive approximation by exceeding the budget constraint by a factor of $\mathcal{O}(\ln |\Pi|/\epsilon)$. We also developed heuristics, most saliently one based on a dynamic programming algorithm. Our experimental analysis on both synthetic and real-world data indicates the effectiveness of Ψ-Saturate and the dynamic-programming-based solution. In the future, we aim to examine the robust configuration of agent-based content features [Ivanov *et al.*, 2017] under a set of adversarial mobility models.

Acknowledgments

Work supported by grants from DFF (P.P. and P.K., 9041-00382B) and Villum Fonden (A.P., VIL42117).

References

[Adiga *et al.*, 2014] Abhijin Adiga, Chris J. Kuhlman, Henning S. Mortveit, and Anil Kumar S. Vullikanti. Sensitivity of diffusion dynamics to network uncertainty. *J. Artif. Intell. Res.*, 51:207–226, 2014.

[Aissi *et al.*, 2009] Hassene Aissi, Cristina Bazgan, and Daniel Vanderpooten. Min–max and min–max regret versions of combinatorial optimization problems: A survey. *Eur. J. Oper. Res.*, 197(2):427–438, 2009.

[Amelkin and Singh, 2019] Victor Amelkin and Ambuj K. Singh. Fighting opinion control in social networks via link recommendation. In *ACM SIGKDD*, pages 677–685, 2019.

[Bollobás *et al.*, 2003] Béla Bollobás, Christian Borgs, Jennifer T. Chayes, and Oliver Riordan. Directed scale-free graphs. In *ACM-SIAM SODA*, pages 132–139, 2003.

[Brin and Page, 1998] Sergey Brin and Lawrence Page. The anatomy of a large-scale hypertextual web search engine. *Computer networks and ISDN systems*, 30(1-7):107–117, 1998.

[Chen *et al.*, 2016] Wei Chen, Tian Lin, Zihan Tan, Mingfei Zhao, and Xuren Zhou. Robust influence maximization. In *ACM SIGKDD*, pages 795–804, 2016.

[Dinur and Steurer, 2014] Irit Dinur and David Steurer. Analytical approach to parallel repetition. In *ACM STOC*, pages 624–633, 2014.

[Du *et al.*, 2013] Nan Du, Le Song, Manuel Gomez-Rodriguez, and Hongyuan Zha. Scalable influence estimation in continuous-time diffusion networks. In *NeurIPS*, pages 3147–3155, 2013.

[Garey and Johnson, 1979] Michael R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman, 1979.

[He and Kempe, 2016] Xinran He and David Kempe. Robust influence maximization. In *ACM SIGKDD*, pages 885–894, 2016.

[Horčík *et al.*, 2022] Rostislav Horčík, Alvaro Torralba, Pavel Rytíř, Lukáš Chrpa, and Stefan Edelkamp. Optimal mixed strategies for cost-adversarial planning games. In *Intl Conf. on Autom. Plan. and Sched.*, pages 160–168, 2022.

[Ivanov *et al.*, 2017] Sergei Ivanov, Konstantinos Theocharidis, Manolis Terrovitis, and Panagiotis Karras. Content recommendation for viral social influence. In *ACM SIGIR*, pages 565–574, 2017.

[Johnson *et al.*, 2000] David S. Johnson, Maria Minkoff, and Steven Phillips. The prize collecting Steiner tree problem: theory and practice. In *ACM-SIAM SODA*, pages 760–769, 2000.

[Karp, 1972] Richard M. Karp. Reducibility among combinatorial problems. In *Complexity of Computer Computations*, pages 85–103, 1972.

[Kempe *et al.*, 2003] David Kempe, Jon M. Kleinberg, and Éva Tardos. Maximizing the spread of influence through a social network. In *ACM SIGKDD*, pages 137–146, 2003.

[Kouvelis and Yu, 1997] Panos Kouvelis and Gang Yu. *Robust Discrete Optimization and Its Applications*, volume 14 of *Nonconvex Optimization and Its Applications*. Springer, 1997.

[Krause and Golovin, 2014] Andreas Krause and Daniel Golovin. Submodular function maximization. In *Tractability: Practical Approaches to Hard Problems*, pages 71–104. Cambridge University Press, 2014.

[Krause *et al.*, 2008] Andreas Krause, H. Brendan McMahan, Carlos Guestrin, and Anupam Gupta. Robust submodular observation selection. *Journ. of Mach. Learn. Res.*, 9(93):2761–2801, 2008.

[Li and Yang, 2020] Huiran Li and Yanwu Yang. Optimal keywords grouping in sponsored search advertising under uncertain environments. *International Journal of Electronic Commerce*, 24(1):107–129, 2020.

[Logins *et al.*, 2020] Alvis Logins, Yuchen Li, and Panagiotis Karras. On the robustness of cascade diffusion under node attacks. In *The Web Conference*, pages 2711–2717, 2020.

[Logins *et al.*, 2022] Alvis Logins, Yuchen Li, and Panagiotis Karras. On the robustness of diffusion in a network under node attacks. *IEEE Trans. Knowl. Data Eng.*, 34(12):5884–5895, 2022.

[Martello and Toth, 1987] Silvano Martello and Paolo Toth. Algorithms for knapsack problems. In *Surveys in Combinatorial Optimization*, volume 132 of *North-Holland Mathematics Studies*, pages 213–257. 1987.

[Minoux, 1978] Michel Minoux. Accelerated greedy algorithms for maximizing submodular set functions. In *IFIP Conference on Optimization Techniques*, pages 234–243, 1978.

[Ordentlich and Cover, 1998] Erik Ordentlich and Thomas M. Cover. The cost of achieving the best portfolio in hindsight. *Math. Oper. Res.*, 23(4):960–982, 1998.

[Ostachowicz *et al.*, 2019] Wieslaw Ostachowicz, Rohan Soman, and Pawel Malinowski. Optimization of sensor placement for structural health monitoring: a review. *Structural Health Monitoring*, 18(3):963–988, 2019.

[Petsinis *et al.*, 2024] Petros Petsinis, Kaichen Zhang, Andreas Pavlogiannis, Jingbo Zhou, and Panagiotis Karras. Robust reward placement under uncertainty. arXiv:2405.05433, 2024.

[Rosenfeld and Globerson, 2016] Nir Rosenfeld and Amir Globerson. Optimal tagging with Markov chain optimization. In *NeurIPS*, pages 1307–1315, 2016.

[von Mises and Pollaczek-Geiringer, 1929] Richard von Mises and Hilda Pollaczek-Geiringer. Praktische verfahren der gleichungsauflösung. *ZAMM - Zeitschrift für Angewandte Mathematik und Mechanik*, 9(1):58–77, 1929.

[Wolsey, 1982] Laurence A. Wolsey. An analysis of the greedy algorithm for the submodular set covering problem. *Combinatorica*, 2(4):385–393, 1982.

[Yu, 1996] Gang Yu. On the max–min 0–1 knapsack problem with robust optimization applications. *Operations Research*, 44(2):407–415, 1996.

[Zhang and Vorobeychik, 2016] Haifeng Zhang and Yevgeniy Vorobeychik. Submodular optimization with routing constraints. In *AAAI Conf. on Artif. Intell.*, pages 819–826, 2016.

[Zhang *et al.*, 2020] Kaichen Zhang, Jingbo Zhou, Donglai Tao, Panagiotis Karras, Qing Li, and Hui Xiong. Geodemographic influence maximization. In *ACM SIGKDD*, pages 2764–2774, 2020.