Traffic Lights Protocols for Smart Cars

Joshua Grinberg, Amr Mohamed, Joseph Shayani

December 7, 2016

1 Introduction

Many car manufacturers are beginning to install a new technology in cars called VANET (Vehicle Ad hoc Network), which allows cars to communicate wirelessly with other nearby VANET-equipped cars. The benefits of this technology include safer driving and more efficient routing. Cars in communication can alert nearby vehicles that another car is expected to run a red light, and they can help coordinate traffic to reduce congestion. An estimated 1.2 million lives are lost in car accidents worldwide annually, and congestion in the United States alone is estimated to exceed $100 billion annually.

Our report investigates how VANET can be used to improve transportation efficiency by coordinating traffic so that it favors cars that are in a rush or in an emergency (i.e. "high-priority" cars) over cars with time to spare (i.e. "low-priority" cars). Existing routing protocols used include stop signs or traffic lights. Unfortunately, these protocols may not be efficient because they do not favor high-priority cars over low-priority cars. In our report, we define a Greedy protocol for settling "conflicts" between cars at intersections given information about cars' priorities, and we design a mechanism for eliciting priority information from cars that incentives cars to reveal their priority levels truthfully. We evaluate our mechanisms using a simulation.[1]

2 Model of the Travel Problem

We start by describing the setup of the "travel problem."

2.1 Networks, roads, and intersections

Cars $1, \ldots, C$ drive on a network of nodes and directed edges. Each node in the network is either a "road" or an "intersection." Roads are nodes with degree 1 or 2, and intersections are nodes with degree greater than 2.

[1] Video: `https://drive.google.com/drive/folders/0B_usRrdgsVFganpNeC1TYzg4dDQ?usp=sharing`

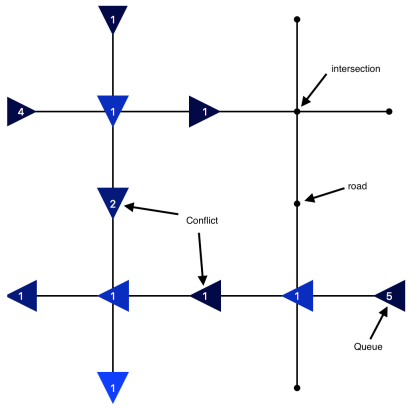

Figure 1: A basic network of cars

2.2 Trips

Each car makes N trips. Each trip is a fixed path in the network, i.e., a sequence of nodes in which every consecutive pair of nodes is connected by a properly oriented edge. We are not considering the problem of choosing optimal routes, so the route of each trip is fixed and cannot be changed mid-trip.

2.3 Moving through the network

At each time step $t = 0, 1, 2, \ldots$, each car advances one node in its trip path, if possible. A road may be occupied by an arbitrarily long queue of cars, but an intersection may only have one car at a time. A car is eligible to advance if

- the next node in its path is a road; or
- the next node in its path is an intersection and the car is at the front of the queue (at its current road).

When multiple cars are eligible to advance to the same intersection, we have a "conflict." A protocol is a set of rules that can be used systematically to settle conflicts. Protocols will be the main object of study of this paper.

2.4 Priorities and costs

Each car is either late or not late. We model late cars as high-priority and the other cars as low-priority. At the start of each round, every car is high-priority with probability $p > 0$, independent of all other cars and priorities in previous rounds, and low-priority with probability $1 - p$. Each time step that a car is on the road, it incurs a cost of 1 if low-priority and $H > 1$ if high-priority.

2.5 Goal of the paper

Once a protocol (see the Section 2.3) is chosen, we can consider running the cars through the network over the N trips and summing up total incurred costs by all the cars. The goal of the paper is to explore how to choose a protocol to minimize this total cost.

3 Applying Mechanism Design to the Travel Problem

3.1 Motivating example

Consider a simple instance of the travel problem described in Section 2: 2 roads each with 1 car, 1 intersection, and $N = 1$ trip.

It is obvious that in order to minimize cost, the protocol must allow the high-priority car to advance before the low-priority car, whenever the 2 cars have different priority. Consider, as an alternative, the protocol that chooses randomly between the two cars, i.e., 50-50. About $p(1-p)$ fraction of the time, the "high-priority first" protocol will result in lower total cost compared to the "50-50" protocol. Indeed, the two cars have different priority levels with probability $2p(1-p)$, and then with probability .5, the 50-50 protocol will allow the low-priority car to advance first.

However, information about the priority of a car's trip is known only to the driver of the car. If the administrator of the high-priority first protocol were to ask cars to report their priority levels, a car with low-priority has an incentive to report high-priority. Hence, we have identified the first incentives issue in the travel problem:

Observation 1. *In some instances of the travel problem, a protocol can use information about the priority levels of cars to reduce total cost. However, if we naively ask cars to report their priority levels, there is an incentive for cars to always overstate their priority levels.*

Now, the problem of choosing a protocol becomes a problem of mechanism design. If in some protocol, for example, cars always report high-priority, then the protocol does not have any better information than the 50-50 protocol. We need to design rules that create an incentive for cars to report their priorities truthfully (at least most of the time).

3.2 Mechanisms, signals, and priorities

In our setting, a mechanism M is a function that maps signals from cars to an assignment of priority levels $\mathcal{P} \in \{l, h\}^C$, where C is the number of cars. To begin, we let the mechanism operate on one signal per car, per trip. At trip n, we allow the mechanism rules to operate on the full history of signals $\mathcal{S}[1:n]$. See Algorithm 1 for more details.

3.3 Using priorities greedily

Suppose that for some trip, we have priorities levels $\mathcal{P} \in \{l, h\}^C$. We might try to resolve conflicts between competing eligible cars greedily, i.e., by selecting the front car from the

Algorithm 1 A mechanism maps car signals to car priorities

function GetPriorities(Mechanism M, Signals $\mathcal{S}$, Trip n)
  for each car $c = 1, \ldots, C$ **do**
    Elicit signal σ from c.
    $\mathcal{S}[n](c) \leftarrow \sigma$
  return Priorities output by mechanism $M(\mathcal{S}[1:n])$

road with the highest total priority (to be defined more precisely later in this section). Such approach generalizes the high-priority first protocol from the simple example in Section 3.1.

Before we define the greedy protocol (see Algorithm 2), we need some notation. Let I be the set of all intersections, and for intersection i, let $R(i)$ be the set of all road nodes r with a directed node from r to i. Let $l(r, t, \mathcal{P})$ denote the number of low-priority cars at road r at time t, and let $h(r, t, \mathcal{P})$ denote the number of high-priority cars, according to the priorities $\mathcal{P}$.

Algorithm 2 A greedy protocol for settling intersection conflicts

procedure Greedy(Mechanism M)
  Initialize signals $\mathcal{S}$ to the empty list of n C-tuples
  for Trip $n = 1, \ldots, N$ **do**
    $t \leftarrow 0$
    $\mathcal{P} \leftarrow$ GetPriorities$(M, \mathcal{S}, n)$
    while Some cars have not yet reached destination **do**
      for $i \in I$ **do**
        for $r \in R(i)$ **do**
          Calculate total cost of cars $c(r) \leftarrow l(r, t, \mathcal{P}) + H * h(r, t, \mathcal{P})$
        Let $\hat{R} \leftarrow \arg\max_{r \in R(i)} \{c(r)\}$ be the set of roads with highest total cost
        if $|\hat{R}| = 1$ **then**
          Pick unique road and let car at front of queue advance
        else
          Pick one road in $\hat{R}$ uniformly at random and let front car advance
      $t \leftarrow t + 1$

3.4 The Button Mechanism: Using latency to truthful reports

Our goal is to design a mechanism that will assign priorities that are close to cars' true priority. As a benchmark for our analysis, we define the ideal mechanism:

Algorithm 3 The ideal, mind-reading mechanism

function IdealMechanism(Signals $\mathcal{S}$)
  return True priorities of each car, independent of signals $\mathcal{S}$

We now work to define a mechanism that comes close to IdealMechanism. Let the signal space Σ consist of two signals, low and high, i.e., $\Sigma = \{l, h\}$. When we elicit signals, we simply ask a car to report its priority. In order to prevent cars from reporting h every trip, we design a mechanism ButtonMechanism that uses latency periods: Whenever car c signals high-priority, ButtonMechanism will assign c low-priority for the next T trips, regardless of c's signal. See Algorithm 4 below.

Algorithm 4 A mechanism that uses a latency period

function ButtonMechanism(Signals $\mathcal{S}$)
  $\mathcal{P} \leftarrow$ empty C-tuple
  $n \leftarrow$ index of last nonempty entry in $\mathcal{S}$
  for each car $c = 1, \ldots, C$ **do**
    if $\mathcal{S}[n](c) = l$, or $\mathcal{S}[m](c) = h$ for any trip $m = n - T, \ldots, n - 1$ **then**
      $\mathcal{P}(c) \leftarrow l$
    else
      $\mathcal{P}(c) \leftarrow h$
  return $\mathcal{P}$

We call the above mechanism ButtonMechanism because we imagine that it could be implemented in the following way: There is a button in the car that the driver can press to guarantee himself high-priority treatment, but he cannot press the button more than once in any window of T trips. We set $T = \lfloor 1/p \rfloor - 1$, so that if the driver presses the button as often as possible (and if $\lfloor 1/p \rfloor = 1/p$), he will press it p-fraction of the time.

Now that we have defined a mechanism, it is time to consider strategies or policies for a car c in the travel problem. For simplicity, adjust the model of the travel problem slightly so that c travels an infinite number of trips. Thus, as long as c has no information about other cars in the network or its own priority-levels in future trips, the car's decision problem, i.e., the problem of deciding whether to signal high or low at the beginning of the trip, is Markovian.

If we also assume that the network is fixed over all of c's trips and that c takes the same path in the network every trip, then the only relevant information for car c's decision is its priority level and the current latency period.

Call the (priority, latency) pair a "state", denoted (π, ℓ). A "policy" Π is a map from states to signals $\{(\pi, \ell) : \pi = l, h; \ell = 0, 1, \ldots, T\} \to \{l, h\}$. We make one more key assumption for our initial analysis:

Assumption 1. *The trip for car c using the* Greedy *protocol is always at least as short if c is assigned high-priority compared to if c is assigned low-priority (when all other cars' trips and priorities are fixed), and sometimes strictly shorter.*

Thus, in the scope of a single trip, a car always prefers to be assigned high-priority. Assumption 1 is certainly true when there is 1 intersection and 2 cars, as in Section 3.1.

What policy is optimal for c? Suppose the paths and priorities of other cars are drawn (independently each trip) from some distribution, which may or may not be known to c. Other cars follows some policy to give signals to ButtonMechanism. Before the start of a

trip n, we can consider the expected duration of c's trip, $E_c(\rho)$, over the resolution of other cars' data and the randomness in the GREEDY protocol, given that c is assigned priority ρ by BUTTONMECHANISM. By Assumption 1, $E_c(h) < E_c(l)$.

We define an "optimal" policy Π as a policy that minimizes average expected cost as the number of trips approaches infinity:

$$A(\Pi) := \lim_{N\to\infty} \frac{1}{N} \sum_{n=1}^{N} \mathbf{E}(\text{Cost of trip } n; \Pi)$$

Here the expectation is over the resolution of other cars' data, the randomness in the GREEDY protocol, and the randomness of car c's priorities $\{\pi_n\}_{n=1}^N$. Because other cars' data and the randomness of GREEDY are independent of $\{\pi_n\}_{n=1}^N$, we can write

$$A(\Pi) = \lim_{N\to\infty} \frac{1}{N} \sum_{n=1}^{N} \sum_{\pi,\rho\in\{l,h\}} \mathbf{P}(\pi_n = \pi, \rho_n = \rho; \Pi) E_c(\rho) K(\pi), \tag{1}$$

where $E_c(\rho)$ is as defined above, and where $K(l) = 1$ and $K(h) = H$ are the costs of low and high priority cars. Here we have averaged over the resolution of other cars' data and the randomness in GREEDY.

Claim 2. *If Π is an optimal policy, then*

1. *$\Pi(\pi, \ell) = l$ for any $\ell > 0$, and*
2. *$\Pi(h, 0) = h$.*

Proof Sketch. Observe, the inner sum of (1) consists of 4 terms. By Assumption 1 and the fact that $C(h) > C(l)$, we know that

$$E_c(h)C(l) \leqslant E_c(l)C(l), E_c(h)C(h) \leqslant E_c(l)C(h)$$

By Assumption 1 and the fact that $C(h) > C(l)$, we have the following 4 inequalities:

$$E_c(h)C(l) < E_c(l)C(l) \tag{2}$$
$$E_c(h)C(l) < E_c(h)C(h) \tag{3}$$
$$E_c(h)C(h) < E_c(l)C(h) \tag{4}$$
$$E_c(l)C(l) < E_c(l)C(h) \tag{5}$$

We can use these inequalities and consider how changing the policy Π increases or decreases $\mathbf{P}(\pi_n = \pi, \rho_n = \rho; \tilde{\Pi})$ for different values of π and ρ to reason about the change in $A(\Pi)$.

Proceed in the following order:

1. Show that optimal policy Π must set some $\Pi(\pi, 0) = h$ for some π. Otherwise $\mathbf{P}(\pi_n = \pi, \rho_n = h; \tilde{\Pi}) = 0$ for both $\pi = l, h$. We can definitely decrease total cost by increasing $\mathbf{P}(\pi_n = l, \rho_n = h; \tilde{\Pi})$ if signal h when $\pi = l$. Here we use inequality (5).

2. Show that if $\Pi(\pi, \ell) = h$ for some π and some $\ell > 0$, since we are signaling wastefully, we can increase $\mathbf{P}(\pi_n = \pi, \rho_n = h; \tilde{\Pi})$ by setting $\Pi(\pi, \ell) = l$ for all $\ell > 0$. We use inequality (3). This proves the first claim.

3. It remains to rule out the policy that signals high only at state $(l, 0)$. Use inequality (4) to show that switching to policy Π that signals at both $(l, 0)$ and $(h, 0)$ decreases total cost.

□

Corollary 3. *The following are the only two possible optimal policies:*

1. ***"Truthful:"*** $\Pi_1(h, 0) = h$ *and* $\Pi_1(\pi, \ell) = l$ *for all other* (π, ℓ)
2. ***"Aggressive:"*** $\Pi_2(\pi, 0) = h$ *and* $\Pi_2(\pi, \ell) = l$ *for if* $\ell > 0$*, regardless of* π*.*

Policy Π_1 is close to "truthful" in the sense that the car signals high-priority only when it is truly on a highipriority trip and as often as possible when it is high-priority, subject to the constraint that it cannot (usefully) signal high-priority during the latency period. Policy Π_2 is "aggressive" in the sense that is signals high priority as soon as the latency period is over, regardless of the car's true priority.

If all cars use Truthful, then the priorities assigned to cars by ButtonMechanism should be close to cars' true priorities. Then we are hopeful that, if Greedy performs well given true priorities, we can achieve low total cost (for all cars) using Greedy with ButtonMechanism. First, however, we need to see if it is in the best interest of cars to use the Truthful policy.

3.5 Testing the incentive compatibility of the Button Mechanism

In this subsection, we show empirically that ButtonMechanism is incentive compatible. That is, we demonstrate that under some assumptions about the set up of the road network and the priority structure, the Truthful policy from Corollary 3 is optimal, regardless of whether other cars are playing Truthful or Aggressive.

We construct a network with a single intersection and $C = 10$ cars. We set the probability of being high priority $p = .3$. We follow a single car c for $N = 10000$ trips. First we assume all other cars use the Truthful policy, and then we calculate the average cost incurred by c when c uses the Truthful policy and the Aggressive policy. Next we assume all other cars use the Aggressive policy, and we again compare the average cost incurred by c when c uses the Truthful policy and the Aggressive policy. We repeat the experiment over various values of high cost H in the range $[2, 10]$.

The results (first plot in Figure 3.5) show that regardless of what policy other cars use, the single car c is better off using the Truthful policy. The figure also illustrates that the spread between Truthful and Aggressive increases as H increases. The intuition behind the increasing spread is that, as H increases, it is relatively less beneficial to be aggressive and signal high when c is low-priority.

The dominance of Truthful is not a special case in the choice of parameters described above. Rather, at least in the case of a single intersection, the results holds more generally.

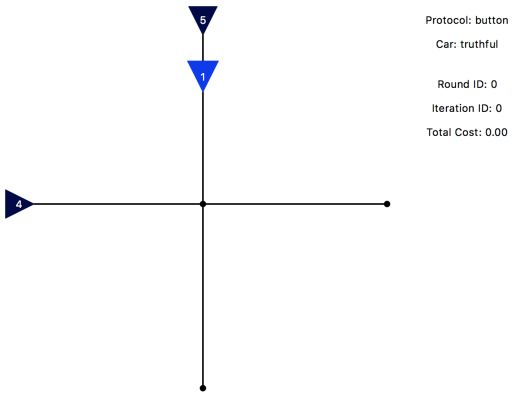

Figure 2: A single-intersection network with many cars.

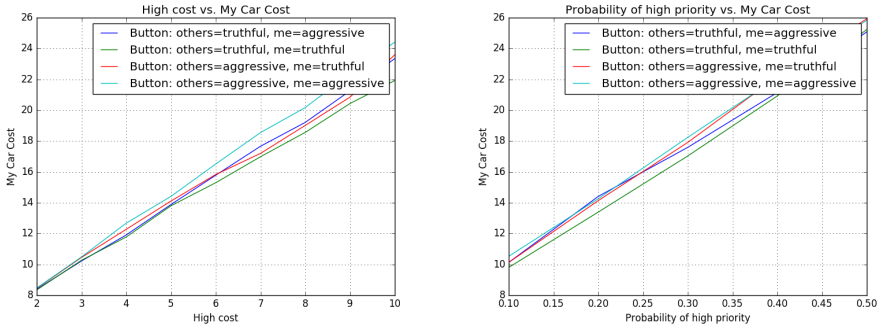

Figure 3: Truthful is dominant, and the dominance is robust to changes in p.

The second plot in Figure 3.5 shows that as we vary the probability of high priority p in $[.1, .5]$, Truthful remains dominant. Of course, after $p > .5$, the latency period $T(p) = 0$, so the ButtonMechanism is degenerate.

3.6 Performance in single-intersection networks

Now that we have shown that ButtonMechanism is incentive compatible, we can inquire about the total societal cost of the Greedy protocol using ButtonMechanism under the assumption that all cars play Truthful.

Greedy(ButtonMechanism) performs almost as well as Greedy(IdealMechanism) in single intersection networks. As expected, it performs much better than the generalization of the 50-50 protocol from Section 3.1. In fact, Greedy(ButtonMechanism) is much closer to the "oracle" protocol Greedy(IdealMechanism) than the "baseline" protocol 50-50. See Figure 3.6.

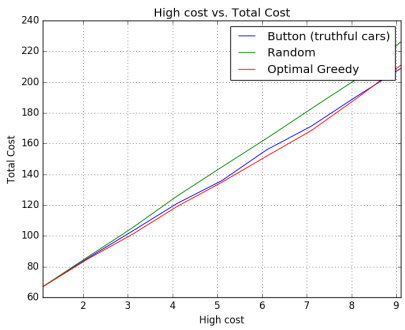

Figure 4: GREEDY runs almost as well as possible when implemented with BUTTONMECHANISM and cars use the Truthful policy.

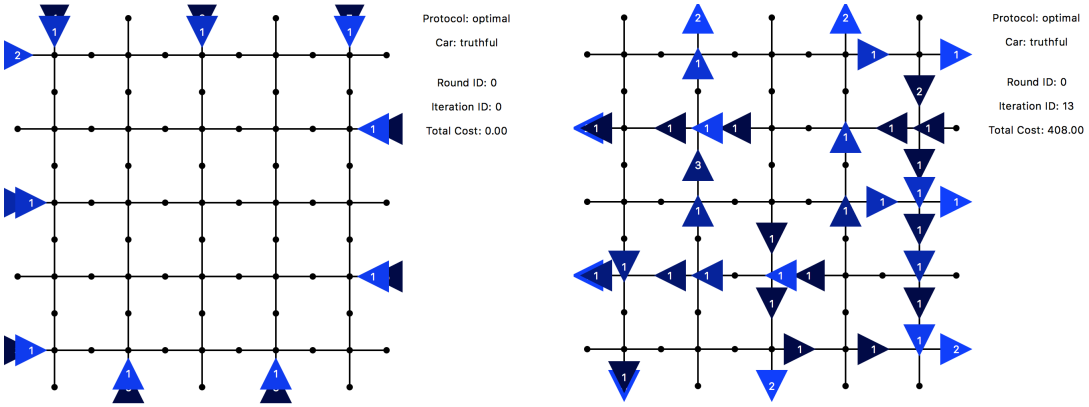

Figure 5: A network with many cars and intersections.

4 Limitations of the Greedy protocol and the Button Mechanism

4.1 The Greedy protocol performs poorly in large networks

The goal of the BUTTONMECHANISM is to elicit truthful reports about cars' priorities in order to allow GREEDY to run with accurate information. But it when the number of roads and cars in a network grows large, even with perfectly accurate information, GREEDY performs worse.

See Figure 4.1 for some intuition: GREEDY allows the high-priority cars going south to advance before the low-priority cars going east. The eastbound low-priority cars are forced to wait, even though it would be a Pareto improvement for some of them to advance first, since the southbound cars come into conflict with the westbound cars at the following intersection. RANDOM, on average, lets half of the low-priority eastbound cars advance.

GREEDY is good at resolving local conflicts and performs well in networks with a single intersection or big networks with few cars. In such examples, the "downstream" effects of local decisions are limited. However, in a big network with many intersections and many cars, there are many downstream effects, and GREEDY may perform no better or worse than

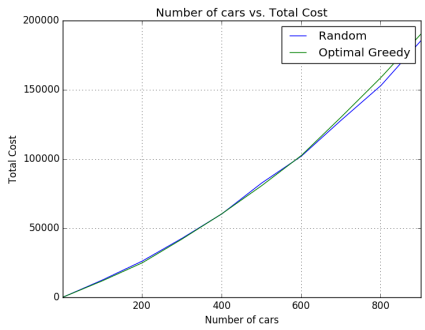

Figure 6: RANDOM outperforms GREEDY when the number of cars in the multi-intersection network grows large.

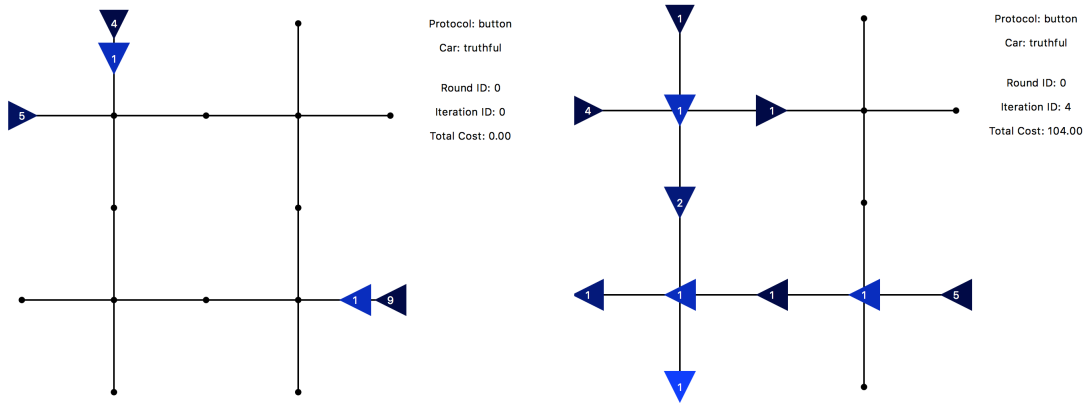

Figure 7: Downstream effects.

RANDOM.

4.2 Driving around to game the system

We have identified at least one way to game BUTTONMECHANISM: In order to escape the latency period, a driver may want to make aimless trips. Aimless driving will create more traffic and hurt overall efficiency.

As long as cars require drivers and the energy necessary for driving a car is expensive, the aimless driving attack will not be attractive for most drivers. However, if cars were self-driving and the cost of fueling cars were inexpensive, the aimless driving attack could be a concern.

The aimless drive attack motivates an alternative mechanism for limiting the amount of times that cars can signal high-priority: It may be possible to use money (either real money or a scrip currency) to allow cars at intersections to participate in auctions. In order to signal high-priority, a car would have to have enough currency. We could charge a driving tax that prevents an aimless low-priority car from accruing currency, so that aimless driving is no longer useful.

We considered trying to extend the theory of the VCG mechanism to implement truth-

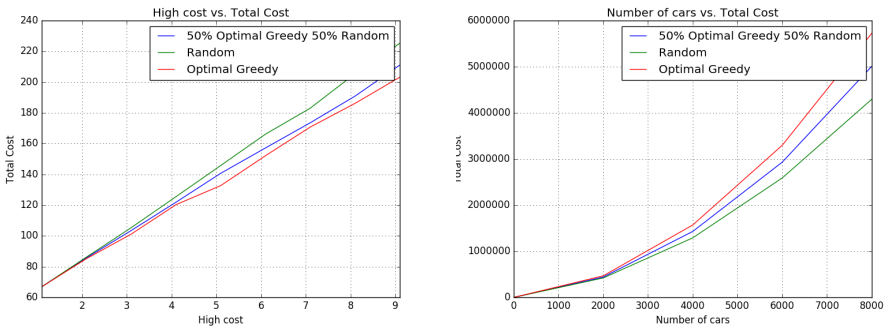

Figure 8: Mixing GREEDY and RANDOM.

ful auctions. Implementing and analyzing VCG proved challenging for several reasons. Because of the dynamic nature of the problem (new cars arriving and leaving), calculating the externality of a car is far from straightforward. In addition, using scrip currency, the value of which is endogenous to the repeated game, breaks the assumption of quasi-linearity of money that is crucial to the truthfulness of the VCG mechanism in a one-shot setting.

5 Extensions

5.1 Mixing Greedy and Random

Whereas in Section 3.6, we saw that GREEDY outperformed RANDOM in single intersection networks; in Section 4.1, we discovered that RANDOM does better in networks with many intersections and cars. These discoveries motivate combining the two protocols to get a protocol that is more robust to a wider variety of networks, i.e., by resolving conflicts randomly sometimes and greedily other times. Figure 5.1 demonstrates that the combined protocol performs nearly as well as GREEDY in the single intersection network and does better than GREEDY in large networks as the number of cars increases.

5.2 Optimal number of Roads

We observe a curious relation between the number of roads and the total cost of a network: When all other parameters, including the number of cars, are fixed, and the number of roads increases, the total cost at first decreases sharply before leveling off and then increasing. See Figure 9.

Note that the decrease in total cost, no doubt due to less congestion, occurs despite an increase in the total distance of travel for each car on each trip. The question of finding the optimal number of roads seems to motivate a problems in urban planning, i.e., find the radius of a city that minimizes cost of travel, given a fixed target population.

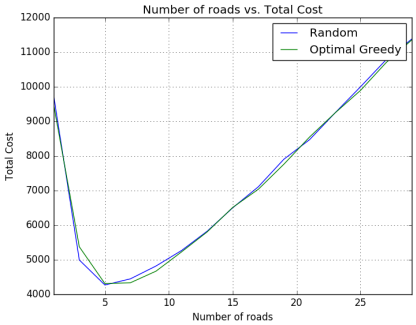

Figure 9: Varying the number of roads.

6 Acknowledgements

We are grateful for the feedback and advice we received from the staff of CS269I, specially Professor Tim Roughgarden. We would also like to thank Dr. Scott Kominers for his suggestion to consider the Button mechanism.