# Stable Matching for Dynamic Ride-Sharing Systems

Tom Manderson, Iain Rudge

October 25, 2017

## 1 Problem Definition

Ride sharing allows people to share transport to reduce the usage of urban transport, typically this is done on short notice over a rolling horizon. Algorithms can then be applied to match riders and drivers in a way that will reduce the cost for each rider and driver while also reducing the total distance driven by vehicles. Therefore the two goals of the optimization is a reduction in cost for each participant, and a reduction in distance traveled system wide.

While most cars can transport up to 5 people, for the ease of modeling we will only consider one rider matched with a single driver, each rider and driver will have attributes that will effect how they are matched. Simplifications in the data, outputs and matching must be made for solutions to be found in a reasonable time. Therefore issues such as variable waiting times, chaining riders or drivers and rerouting due to traffic will all be ignored. Instead it is assumed that one driver can carry one passenger and will not pick up anyone else afterwards, similarly one rider will only travel with a single driver and not rematch afterwards. This assumption is made as otherwise the problem will become too complex and the notions of stability and cost will have to be refined, it would also increase the solution space of the model, requiring more arcs to be generated, which will massively affect the run time and complexity of the problem.

### 1.1 Motivation

The reduction of overall distance driven in a system is an important field of study, reduction in road usage will also address congestion, road wear and environmental factors. With such benefits potentially available to major urban areas it becomes apparent that carpooling is desirable from the perspective of a transport system. However by compensating a driver partially for distance driven this encourages the use of ride sharing by both driver and rider, through the potential to save money on the cost of transport it is better for all participants involved. The clear benefits to both a transport system and an individual are obvious, however these two benefits are not exactly identical. The solution for a system wide reduction in distance will not the be same as a solution that saves each rider the most amount of money possible, as such there is almost an optimization with two potential solutions, a system optimal one and one that is optimal for participants. However any tendency towards a system optimal solution will still be better than what is currently practiced as it will make some impact on the reduction of both pollution and road damage.

### 1.2 Aims

The aim of this report is to replicate and improve on the methods and models used in the original paper, while also exploring the effects of stability on the solution.

### 1.3 Problem Formulation

#### 1.3.1 Sets

$$P \quad \text{The set of all locations } p \in P \tag{1}$$

Note that each location $p \in P$ must have a distance from all other locations, that is $dist(p_m, p_n)$ must exist $\forall p_m, p_n \in P$ this distance is needed to calculate the total distance savings and individual cost savings for both individuals and the overall system.

$$D \quad \text{The set of all Drivers } d \in D \tag{2}$$
$$R \quad \text{The set of all Riders } r \in R \tag{3}$$

Here each participant $s \in R \cup D$will have the following data:

$$o_s \quad \text{Origin of participant } s \tag{4}$$
$$d_s \quad \text{Destination of participant } s \tag{5}$$
$$td_s \quad \text{Earliest departure time of participant } s \tag{6}$$
$$ta_s \quad \text{Latest arrival time of participant } s \tag{7}$$

The above data is required for the matching and costing algorithms used in the formulation, when a rider or passenger is introduced to the system the data is generated and associated with them. This is vital for the creation of the set of Arcs $A$ seen at 8

$$A \quad \text{The set of All Arcs } a \in A \tag{8}$$
$$A_{i,j} \quad \text{The Arc connecting Rider } i \text{ to Driver } j \tag{9}$$
$$\text{iff this pairing is feasible} \quad \forall (i,j) \in (R, D) \tag{10}$$

The set of Arcs 8 is the description of matching between riders and drivers, each arc represents a feasible pairing between a rider and a driver, where feasible is defined as being allowable in time, and that a saving of some form must be achieved for both parties. These arcs are what is used to solve the optimisation problem. As such Arc generation should be seen to be a vital part of the model.

### 1.3.2 Variables

Due to the use of arcs in the model, which are generated apriori prior to the IP solution, the only variable required for the problem is a binary value such that it is 1 if the arc is used and 0 otherwise.

$$x_{i,j} = \begin{cases} 1 & \text{if } A_{i,j} \text{ is used} \\ 0 & \text{otherwise} \end{cases} \tag{11}$$

### 1.3.3 Objective

The objective function will remain constant for most of the models discussed in the paper, however may be altered or added to in order to better understand stability as shown in section 2.4.2. In which the objective function is altered to introduce stability into the solution. The objective function discussed below in equation 12 is used in formulations found in sections 2.2,2.3 and 2.4.1

$$\max \sum_{(i,j) \in A} x_{i,j} \left( dist(o(j), d(j)) - dist(o(j), o(i)) - dist(d(i), d(j)) \right) \tag{12}$$

That is the objective of the optimisation is to maximise the savings generated per pairing for every arc. The maximization of savings is the same thing as minimising the total amount of distance traveled. As only savings from feasible and used arcs are considered this objective function accurately portrays the predominant aim of the task, to reduce total distance traveled system wide. However the selection of arcs in order to fulfill the optimal financial savings per participant is not considered in this formulation, and instead is dependent on both constraints and the definition of stability, as further explored in section 1.4.

### 1.3.4 Constraints

The constraints in this section are consistent in every formulation of the model, other constraints may be added for other models however these are the fundamental constraints on the system.

$$\sum_{j \in D} x_{i,j} \leq 1 \quad \forall i \in R \tag{13}$$

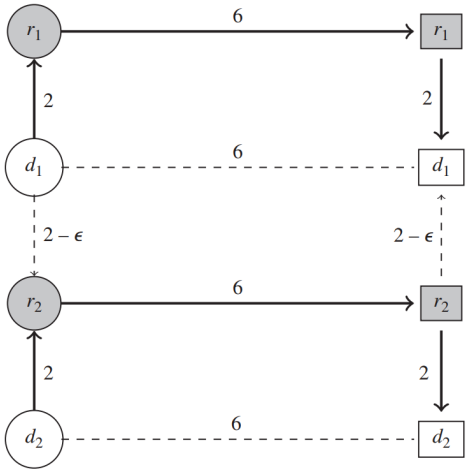
$$\sum_{i \in R} x_{i,j} \leq 1 \quad \forall j \in D \tag{14}$$

Equations 13,14 both add constraints that a driver can match with at most one rider, and a rider can match with at most one driver. This does not force all drivers or riders to match as some may have no feasible pairings, however these constraints will later contribute to the need to define stability, as discussed in section 1.4.

## 1.4 Stability Definition

Stability is the key concept that this paper and model is based upon. Drivers and riders do not need to accept a match that is proposed by the model, even if the match presents some saving to both parties. Riders or drivers may reject a match if they believe that they could find a better savings. Game Theory can be used to define a notion of stability, that is a system is stable if no rider and driver pair that is currently matched to others or unmatched would prefer to be matched with each other.
A blocking pair is any pair in the solution where both participants prefer to be matched with each other over their current matches (or lack thereof). Any solution that contains a blocking pair will no longer be a stable solution.

Figure 1: Diagram for stability explanation




Consider the example found in figure 1, if $0 < \epsilon < 1$ then the system optimal solution is to assign $r_1$ to $d_1$ an $r_2$ to $d_2$.This results in a system wide saving of 4 and a saving of 1 for each participant.
However $r_2$ and $d_1$ would prefer to be matched to one another rather than their current partners. This will increase the individual savings by $\epsilon$ and reduce the system wide savings to $2 + 2\epsilon$ in the worst case this will 2. Therefore $r_2, d_1$ is a blocking pair and means the system is not stable.

However for $1 < \epsilon \leq 2$ the system optimal assignment in Figure 1 is to match $r_2$ to $d_1$ and this assignment is stable. We may consider $\epsilon$ to be the price of stability. Later in the formulation we will discuss how it is used to determine marginally stable solutions.

# 2 Prior Art

The paper this work is based on defines 3 types of problems that could be considered in the overall model, arranged in order of difficulty they are; the system optimal formulation, the stable formulation and the unstable formulation. As mentioned in section 1.4 stability plays a major role in solutions of problems. While the system optimal solution provides the maximum vehicle-distance savings across the whole system, the stable formulation satisfies individual participant preferences first and foremost, and the unstable solutions provide tradeoffs between these two extremes.

## 2.1 Arc Generation

Arc generation is the underlying algorithm to the modeling of the data, without accurate arc generation the formulation would be impossible to solve. As arc generation is consistent across all formulations it is vital that it is efficient and well designed. The paper suggests a greedy algorithm approach to arc generation, that is it will check every rider against every driver, whether they are feasible or not. Once this is done a check is performed to determine if the rider can be matched with the driver, after this is performed a reduction in travel is calculated. All potential arcs for the problem are returned from this algorithm and are used in the model as the set $A$.

## 2.2 System optimal solution

The system optimal solution is the most basic formulation, there is no additional constraints. Therefore the model is the maximiasation of distance saved while obeying simple constraints. That is one driver can match with one rider and most 14, one Rider can match with one driver at most 13 and that each arc represents a feasible solution where at least some saving is made.
This formulation is the most basic however is used to benchmark the quality of a solution. The system optimal solution is the best possible solution and does not consider the savings or preferences of an individual. As such while this solution is ideal it is unlikely that it would ever be used as many participants will not use the system unless some substantial saving is made.

## 2.3 Stable solution

The Simple stable solution is similar to the system optimal solution, however now there also exists a sub optimisation for participant savings. The constraint below is added to the system and will act in a way such that the optimal savings of each driver and rider are used where possible. This will create a gap between the system optimal solution and the stable solution.

$$\sum_{j`\geq j_i} x_{i,j`} + \sum_{i`\geq i_j} x_{i`,j} + x_{ij} \geq 1 \quad \forall (i,j) \in A \tag{15}$$

That is if arc $(i, j)$ is used then it must present the best savings for both the rider and driver involved. However this presents an issue, if some driver $d_1$ prefers to match with a rider $r_1$ then they must match according to this constraint, however if rider $r_1$ prefers to match with another driver $d_2$ then this constraint cannot be satisfied for both cases without violating constraints 13,14. This issue is inherent in the data of the problem, there may exist cases where a stable formulation is viable, however most commonly this is not the case, as such a formulation for the unstable model should be considered.

## 2.4 Unstable solution

The unstable formulation is required when no stable solution can be found, relying on the definition of a blocking pair as described in section 1.4 The solutions in this section seek to apply a price to the cost of stability. Here we refer to the price of stability as the cost associated with allowing a blocking pair to exist, for a blocking pair to exist they must generate a substantial amount of savings more than if they were otherwise matched in a system optimal formulation. This cost of stability here will be represented by $\epsilon$. The two below formulations use similar a similar principle of the cost of stability

however implement it in different ways, the first through constraints and the other through a changed objective function.

#### 2.4.1 Stability as a constraint

This model seeks to add a constraint on the cost of stability, however a pair may not be able to notice a small change in savings, or the blocking pair is causing drastic reduction is solution quality. Both of these are not ideal and as such the value $\epsilon$ is used where a pair is not matched unless the savings of both Rider and driver is $\epsilon$ greater than a stable solution. This is implemented in the following three constraints.

$$S_{i,j} > S_{i,u(j)} + \epsilon \tag{16}$$

$$S_{i,j} > S_{u(i),j} + \epsilon \tag{17}$$

$$\sum_{j' \geq j(\epsilon,i)} X_{i,j'} + \sum_{i' \geq i(\epsilon,j)} X_{i',j} + X_{i,j} \geq 1 \tag{18}$$

That is, Rider i will not change pairs unless this new pairing is $\epsilon$ better than its previous match and similarly with the drivers. This constraint adds a tendency towards a marginally stable solution, where only some participants are not in there preferred pairing.

There is an especially useful property of this formulation: when there are no ties in preferences, the optimal solution is exactly equivalent to the linear relaxation of the problem.

#### 2.4.2 Stability as an objective

This formulation is similar to that found in the above section, however instead of including the cost of stability as a constraint, the objective function is altered to minimize the cost associated with the cost of stability, this is implemented as seen below.

$$S_{i,j} > S_{i,u(j)} + \epsilon \tag{19}$$

$$S_{i,j} > S_{u(i),j} + \epsilon \tag{20}$$

$$min \sum_{i \in R, j \in D} Y_{ij} \tag{21}$$

$$Y_{ij} \geq 1 - \left( \sum_{j' \geq j(\epsilon,i)} X_{i,j'} + \sum_{i' \geq i(\epsilon,j)} X_{i',j} + X_{i,j} \right) \quad \forall (i,j) \in A \tag{22}$$

# 3 Student Formulations

The formulations below are a combination of reformulations of the papers work and formulations and techniques developed throughout the process of the assignment. The main hurdles that were met during this were a proper understanding and definition of stability and data generation to properly model the systems. The four formulations discussed in the paper were reimplemented and found to have drastically faster run times than that of the original work, this may be in part to using Gurobi over Cplex, or through a more efficient arc generation method. The formulations developed can be found in the below sections.

## 3.1 Arc Generation

The generations of Arcs was the largest bottleneck found in the problem formulation, as such it became the first section of the code to be reformulated, while the original paper implements a naive greedy algorithm we have instead opted to implement a method of arc generation that should be faster and more sensible. As we have ordered all of our participants by the earliest departure time , a rolling

**Algorithm 1** Arc Generation Algoritm

```
for i ∈ R do
    for j ∈ D do
        if d_i > a_j then
            Continue
        if d_j > a_i then
            Break
        if Pickup + Dropoff > d trip then
            Continue
        Overlap = min(a_i, d_i) − max(rider pickup, d_j)
        if r trip × time/km > Overlap then
            Continue
        if d trip − (pickup+Dropoff) > 0 then
            Continue
        MakeArc(i, j)
```

window can be used to dramatically reduce the amount of computation required in the Arc generation algorithm, this method can be seen below.

As can be seen the reduction in potential arcs checked as well as a reduction in potential arcs will dramatically increase the run time of the model. This is magnified when this algorithm is parallelised, which can be done so trivially by executing iterations of the outer loop in parallel.

## 3.2 Stable Solution

The stable solution used in the student formulation was the same as suggested in the paper, no additional reformulation was required as it was a capable method that solved very quickly, additionally to this the method did not suit itself to lazy constraints as the three constraints required are all called often enough that lazy constraints would not be a feasible speed up.

## 3.3 Unstable Solutions

In addition to reimplementing the paper's constraint-based epsilon stability and objective-based stability maximisation, we implemented and tested three other approaches to the tradeoff between stability and system optimality.

### 3.3.1 Lazy Constraints of Stability

We explored two different approaches using lazy constraints, which we deemed "upper" and "lower" constraints. Both applied the standard formulation of stability constraints (Equation 2.4.1) lazily, only differing in the conditions in which the constraint was added.

In the "upper" variant, stability constraints were applied on all arcs that were selected (i.e. their variables had the "upper" value) in the intermediate solution. This effectively produced a formulation that was identical to the original paper's stable solution, but with the constraints applied lazily. This method significantly increased solve runtime as Gurobi was unable to take advantage of stability constraints in presolve, and in addition to the fact that the result was not novel, we discarded this approach.

In the "lower" variant, stability constraints were applied on any unselected arcs (variables with the "lower" value) in the intermediate solution. By only adding stability constraints to unselected arcs, we effectively ensure that any intermediate MIP solution provides a lower bound on stability, so we can only select arcs that are more likely to be preferred. This is a novel approach to the tradeoff between stability and system optimality, but provides less control than other methods tested.

For the rest of this report, the term "lazy constraint approach" refers only to the "lower" variant.

### 3.3.2 Epsilon-Objective Stability

In the original paper, the authors explored a formulation that added an additional set of variables, $Y_{ij}$, to decide whether or not to consider the stability constraint for an arc $(i, j)$. This formulation is covered in Section 2.4.2.

We replicated the constraint used in the paper's formulation (Equation 22) and used this as our stability constraint, but rather than altering the objective to only consider the $Y_{ij}$ as in Equation 21, we modified the original objective by adding a constraint pricing term.

$$\max \sum_{(i,j)\in A} X_{ij}c_{ij} - \epsilon Y_{ij} \tag{23}$$

By including our $Y$ variables as a negative term in our maximisation objective, we ensure that the optimisation attempts to branch these variables to 0, in which case the stability constraint for the given variable (Equation 22) behaves in the same way as the simple stable case. By adjusting our $\epsilon$ we then set a "price on stability".

### 3.3.3 Value-dependent Constraint Pricing

We extended the above epsilon-objective stability formulation by changing how we priced our $Y$ terms in our objective. Instead of having a fixed cost $\epsilon$ for each $Y_{ij}$, we instead made the price depend on the savings from the given arc:

$$\max \sum_{(i,j)\in A} c_{ij}\left(X_{ij} - \alpha Y_{ij}\right) \tag{24}$$

We refer to this as "value-dependent constraint pricing" as it effectively puts a price on whether or not to include a stability constraint, and this price is determined based on the value of the given arc in terms of vehicle-mile savings.

The $\alpha$ term in this formulation's objective, Equation 23, is used as a multiplier to reduce the cost of $Y_{ij}$ variables. We found that an $\alpha$ value of 0.1 gave consistently good results.

# 4 Data

## 4.1 Generation

Despite reaching out to the original paper authors, we were unable to obtain the data used in the paper, and as such were forced to generate our own. Our data generation depended on a set of several parameters:

- Location count, `LOCATION_COUNT`
- Maximum X/Y distance, `MAX_XY`
- Announcement count, `ANNOUNCEMENT_COUNT`
- Maximum simulation time, `MAX_TIME`
- Time flexibility, `FLEXIBILITY` (always 20)

Each location in the set of locations, $P$, was generated by selecting a uniform random $x$ and $y$ coordinate from the set $[0, \texttt{MAX\_XY}]$. By using 2D coordinates it becomes very easy to determine the distance between each location. This was repeated until `LOCATION_COUNT` locations were generated.

Each announcement in the set of announcements, $S$, was generated by selecting a start and end location randomly from $P$, and uniformly selecting a earliest departure time from $[0, \texttt{MAX\_TIME}]$. To that earliest departure time, we add the time flexibility, `FLEXIBILITY`, and the time taken (in minutes) to travel from origin location to destination location (assuming 60km/h), resulting in our latest arrival time. If this latest arrival time is greater than `MAX_TIME`, the result is discarded and generating the result begins again. We then randomly decided whether the announcement is a rider or driver with equal probability. This process is repeated until `ANNOUNCEMENT_COUNT` announcements are generated.

### 4.2 Datasets

We generated 10 datasets for testing our various implementations on. Their names and generation parameters are listed below. Filenames have `.pickle` appended.

| Name | Locations | Max X/Y | Announcements | Max Time |
|---|---|---|---|---|
| small5k | 100 | 200 | 5000 | 1000 |
| small15k | 100 | 200 | 15000 | 1000 |
| medium1k | 200 | 200 | 1000 | 1000 |
| medium10k | 200 | 200 | 10000 | 1000 |
| medium20k | 200 | 200 | 20000 | 1000 |
| loc300a5k | 300 | 300 | 5000 | 2000 |
| loc500small15k | 500 | 200 | 15000 | 1000 |
| large15k | 500 | 1000 | 15000 | 4000 |
| large20k | 500 | 1000 | 20000 | 2000 |
| realistic | 1000 | 20 | 12000 | 300 |

## 5 Results

We examine our solutions in terms of two main metrics: total system savings, which is our main optimization goal, and blocking pairs, which serves as a metric for stability with which we can examine the tradeoffs that different formulations provide. Additionally, for our very large “realistic" dataset we also examine solution time.

### 5.1 Realistic Dataset

The “realistic” dataset is called that due to the model parameters used - the area considered is 20km by 20km, and we consider 12000 rider/driver announcements over a 5 hour window - far closer to realistic situations than the other datasets. As such, 383721 arcs are generated (an order of magnitude more than any other dataset) and both model declaration and optimization take a significant amount of time to complete. Because of this, we have included more information about this dataset than other datasets, which are listed together on the following page.

| Solution Method | Model build time (s) | Solve time (s) | Savings | Blocking pairs |
|---|---|---|---|---|
| System Optimal | 16.58 | 8.93 | 43198.54 | 677 |
| Simple Stable | 729.43 | 40.78 | 41263.04 | 0 |
| Constraint Epsilon | 1360.29 | 555.33 | 43198.54 | 677 |
| Paper Objective | 806.25 | TIMEOUT: 1800 | - | - |
| Objective Epsilon | 671.56 | 337.81 | 41552.55 | 26 |
| Value Pricing | 826.76 | 483.03 | 42755.02 | 298 |
| Lazy | 15.65 | TIMEOUT: 1800 | - | - |

## 5.2 Small Datasets

Nine of the ten datasets tested on are small enough that solves are near-instantaneous. These datasets have their results presented without solution times for this reason. Due to the number of implementations tried, this table is split in two parts: prior art, and novel approaches

Prior art:

| Dataset | Arcs | System Optimal | | Simple Stable | | Constraint Epsilon | | Paper Objective | |
|---|---|---|---|---|---|---|---|---|---|
| | | Savings | Blocking | Savings | Blocking | Savings | Blocking | Savings | Blocking |
| small5k | 2494 | 66062.66 | 37 | 64984.89 | 0 | 65611.20 | 23 | 64984.89 | 0 |
| small15k | 21887 | 335828.90 | 459 | 323218.99 | 0 | 331638.97 | 306 | 322527.03 | 0 |
| large15k | 862 | 189042.04 | 3 | 188796.24 | 0 | 189035.71 | 1 | 188796.24 | 0 |
| large20k | 3479 | 644497.27 | 52 | 637466.47 | 0 | 641281.33 | 32 | 637466.47 | 0 |
| medium1k | 97 | 4291.66 | 0 | 4291.66 | 0 | 4291.66 | 0 | 4291.66 | 0 |
| medium10k | 8984 | 154893.49 | 177 | 149903.23 | 0 | 153285.20 | 117 | 149881.50 | 0 |
| medium20k | 34266 | 424292.57 | 692 | 405308.23 | 0 | 418230.73 | 463 | 404878.92 | 0 |
| loc300a5k | 692 | 36296.58 | 4 | 36049.00 | 0 | 36144.02 | 1 | 36049.00 | 0 |
| loc500small15k | 18382 | 263388.36 | 394 | 252702.08 | 0 | 260195.41 | 272 | 252702.08 | 0 |

Novel approaches:

| Dataset | Arcs | Objective Epsilon | | Value Pricing | | Lazy Matching | |
|---|---|---|---|---|---|---|---|
| | | Savings | Blocking | Savings | Blocking | Savings | Blocking |
| small5k | 2494 | 66032.97 | 22 | 66027.35 | 21 | 64984.89 | 0 |
| small15k | 21887 | 335537.01 | 309 | 334999.10 | 251 | 323218.99 | 0 |
| large15k | 862 | 189035.71 | 1 | 189035.71 | 1 | 188796.24 | 0 |
| large20k | 3479 | 644476.00 | 40 | 644340.88 | 30 | 637466.47 | 0 |
| medium1k | 97 | 4291.66 | 0 | 4291.66 | 0 | INFEASIBLE | |
| medium10k | 8984 | 154753.74 | 117 | 154684.43 | 111 | 149903.23 | 0 |
| medium20k | 34266 | 423743.95 | 427 | 423344.64 | 380 | 405308.23 | 0 |
| loc300a5k | 692 | 36292.06 | 3 | 36292.06 | 3 | INFEASIBLE | |
| loc500small15k | 18382 | 263094.38 | 258 | 262900.29 | 238 | 252702.08 | 0 |

# 6 Discussion

When comparing and contrasting approaches to solving the ride-sharing problem, our solutions should be examined in terms of the system optimal and simple stable solutions - these provide a bound on vehicle-distance savings and blocking pairs. If an approach selects more blocking pairs than our system optimal solution, it is making poor decisions and should be discarded, but we cannot save more than our system optimal solution. If an approach has less savings than our simple stable solution, it is trivial to see that the simple stable solution will satisfy individual preferences at least as well (as simple stable always has 0 blocking pairs), while also performing better system-wide. One additional fact not listed in the results table is that the simple stable solution was solved in Gurobi's presolve step for all but the "realistic" dataset.

With this in mind, we can discard the lazy constraints approach to the stability-savings tradeoff. In all situations, the lazy constraints performs at best as well as our simple stable solution, and at worst creates an infeasible model or times out. As the lazy constraints approach is simply selectively applying the simple stability constraints, this points more towards an implementation error than an inherently flawed approach.

We can also discard the approach labeled "Paper Objective" - this refers to the approach listed in Section 2.4.2. This approach is effectively trying to minimise blocking pairs selected, which will guarantee us a stable solution. Unfortunately, as the objective in this formulation does not include total savings, we lose the guarantee that the stable solution found by this formulation is the best possible, so our simple stable solution will perform better. We additionally lose the power gained by the relationship between the simple stable formulation and its linear relaxation, which leads to large performance benefits. Interestingly, the timeout for solving this formulation occurred while still attempting to find the root node of the problem.

This leaves the value pricing, constraint epsilon, and objective epsilon formulations. On the small datasets constraint pricing was consistently more stable but less optimal, objective epsilon more optimal but less stable, and value pricing providing a middle ground in both stability and optimality. This is not replicated on the "realistic" dataset, where the constraint epsilon approach ends up producing the system optimal solution. This most likely occurs as the epsilon value for constraint epsilon was tuned on the small datasets, and ends up producing incredibly loose constraints on this dataset.

Despite constraint-epsilon changing behavior, both value pricing and objective epsilon approaches seem to still perform well, and provide different and useful tradeoffs between stability and savings while also keeping solve time down to a reasonable level. Both approaches also have model parameters that could be tuned differently for the realistic approach, but as value pricing provides a scaling factor on savings rather than an absolute contribution to the objective, it is not as necessary to examine the value pricing $\alpha$ parameter.

# 7 Conclusion

We examined four different approaches from our original paper, successfully implementing and benchmarking them after significant challenges gaining a comprehensive understanding of the concept of stability. Additionally, we developed three novel approaches to the problem, two of which performed well in comparison to the prior art approaches and provided new options when examining the tradeoff between stability and total system-wide savings.