# Public Project with Minimum Expected Release Delay

Guanhua Wang and Mingyu Guo*

School of Computer Science, University of Adelaide, Adelaide, Australia
{guanhua.wang, mingyu.guo}@adelaide.edu.au

**Abstract.** We study the excludable public project model where the decision is binary (build or not build). In a classic excludable and binary public project model, an agent either consumes the project in its whole or is completely excluded. We study a setting where the mechanism can set different project release time for different agents, in the sense that high-paying agents can consume the project earlier than the low-paying agents. The mechanism design objective is to minimize the expected maximum release delay and the expected total release delay. We propose the single deadline mechanisms. We show that the optimal single deadline mechanism is asymptotically optimal for both objectives, regardless of the prior distributions. For small number of agents, we propose the sequential unanimous mechanisms by extending the largest unanimous mechanisms from Ohseto [8]. We propose an automated mechanism design approach via evolutionary computation to optimize within the sequential unanimous mechanisms.

**Keywords:** Automated Mechanism Design · Public Project · Cost Sharing

## 1 Introduction

The public project problem is a fundamental mechanism design model with many applications in multiagent systems. The public project problem involves multiple agents, who need to decide whether or not to build a public project. The project can be **nonexcludable** (*i.e.*, if the project is built, then every agent gets to consume the project, including the non-paying agents/free riders) or **excludable** (*i.e.*, the setting makes it possible to exclude some agents from consuming the project) [8]. A public project can be **indivisible/binary** or **divisible** [7]. A binary public project is either built or not built (*i.e.*, there is only one level of provision). In a divisible public project, there are multiple levels of provision (*i.e.*, build a project with adjustable quality).

In this paper, we study an excludable public project model that is "divisible" in a different sense. In the model, the level of provision is binary, but an agent's consumption is divisible. The mechanism specifies when an agent can start consuming the project. High-paying agents can consume the project earlier, and the free riders need to wait. The waiting time is also called an agent's **delay**. The delay is there to incentivize payments. The model was proposed by Guo *et al.* [6]. The authors studied the following mechanism design scenario. A group of agents come together to crowd-fund a piece of security information. No agent is able to afford the information by herself.[1] Based

* Corresponding author

[1] Zero-day exploits are very expensive [5, 4].

on the agents' valuations on the information, the mechanism decides whether or not to crowd-fund this piece of information (*i.e.*, purchase it from the security consulting firm that is selling this piece of information). If we are able to raise enough payments to cover the cost of the security information, then *ideally* we would like to share it to all agents, including the free riders, in order to maximizes the *overall protection of the community*. However, if all agents receive the information regardless of their payments, then no agents are incentivized to pay. To address this, the mechanism releases the information only to high-paying agents in the beginning and the non-paying/low-paying agents need to wait for a delayed release. The mechanism design goal is to minimize the delay as long as the delay is long enough to incentivize enough payments to cover the cost of the information. Guo *et al.* [6] proposed two design objectives. One is to minimize the *max-delay* (*i.e.*, the maximum waiting time of the agents) and the other is to minimize the *sum-delay* (*i.e.*, the total waiting time of the agents). The authors focused on *worst-case mechanism design* and proposed a mechanism that has a constant approximation ratio compared to the optimal mechanism. The authors also briefly touched upon *expected* delay. The authors used simulation to show that compared to their worst-case competitive mechanism, the *serial cost sharing mechanism* [7] has much lower expected *max-delay* and *sum-delay* under various distributions.

In this paper, we focus on minimizing the expected *max-delay* and the expected *sum-delay*. We propose a mechanism family called the **single deadline mechanisms**. For both objectives, under minor technical assumptions, we prove that there exists a single deadline mechanism that is *near optimal* when the number of agents is large, *regardless of the prior distribution*. Furthermore, when the number of agents approaches infinity, the optimal single deadline mechanism approaches optimality asymptotically. For small number of agents, the single deadline mechanism is not optimal. We extend the single deadline mechanisms to multiple deadline mechanisms. We also propose a genetic algorithm based automated mechanism design approach. We use a sequence of offers to represent a mechanism and we evolve the sequences. By simulating mechanisms using multiple distributions, we show that our genetic algorithm successfully identifies better performing mechanisms for small number of agents.

## 2 Related Research

Ohseto [8] characterized all strategy-proof and individually rational mechanisms for the binary public project model (both excludable and nonexcludable), under minor technical assumptions. Deb and Razzolini [2] further showed that on top of Ohseto's characterization, if we require *equal treatment of equals* (*i.e.*, if two agents have the same type, then they should be treated the same), then the only strategy-proof and individually rational mechanisms are the *conservative equal cost mechanism* (nonexcludable) and the *serial cost sharing mechanism* (excludable), which were both proposed by Moulin [7]. It should be noted that Ohseto's characterization involves *exponential* number of parameters, so knowing the characterization does not mean it is easy to locate good mechanisms. Wang *et al.* [11] proposed a neural network based approach for optimizing within Ohseto's characterization family. The authors studied two objectives: maximizing the number of consumers and maximizing the social welfare. It should be

noted that Ohseto's characterization does not apply to the model in this paper, as our model has an additional spin that is the release delay. In this paper, we propose a family of mechanisms called the sequential unanimous mechanisms, which is motivated by Ohseto's characterization. We apply a genetic algorithm for tuning the sequential unanimous mechanisms. Mechanism design via evolutionary computation [9] and mechanism design via other computational means (such as linear programming [1] and neural networks [3, 10, 11]) have long been shown to be effective for many design settings.

## 3 Model Description

There are $n$ agents who decide whether or not to build a public project. The project is binary (build or not build) and nonrivalrous (the cost of the project does not depend on how many agents are consuming it). We normalize the project cost to 1. Agent $i$'s type $v_i \in [0, 1]$ represents her private valuation for the project. We use $\vec{v} = (v_1, v_2, \ldots, v_n)$ to denote the type profile. We assume that the $v_i$ are drawn *i.i.d.* from a known prior distribution, where $f$ is the probability density function. For technical reasons, we assume $f$ is *positive* and *Lipschitz continuous over* $[0, 1]$.

We assume that the public project has value over a time period $[0, 1]$. For example, the project could be a piece of security information that is discovered at time 0 and the corresponding exploit expires at time 1. We assume the setting allows the mechanism to specify each agent's release time for the project, so that some agents can consume the project earlier than the others. Given a type profile, a mechanism outcome consists of two vectors: $(t_1, t_2, \ldots, t_n)$ and $(p_1, p_2, \ldots, p_n)$. ***I.e.***, agent $i$ starts consuming the project at time $t_i \in [0, 1]$ and pays $p_i \geq 0$. $t_i = 0$ means agent $i$ gets to consume the public project right from the beginning and $t_i = 1$ means agent $i$ does not get to consume the public project. We call $t_i$ agent $i$'s *release time*. We assume the agents' valuations over the time period is uniform. That is, agent $i$'s valuation equals $v_i(1-t_i)$, as she enjoys the time interval $[t_i, 1]$, which has length $1 - t_i$. Agent $i$'s utility is then $v_i(1 - t_i) - p_i$. We impose the following mechanism design constraints:

- Strategy-proofness: We use $t_i$ and $p_i$ to denote agent $i$'s release time and payment when she reports her true value $v_i$. We use $t'_i$ and $p'_i$ to denote agent $i$'s release time and payment when she reports a false value $v'_i$. We should have
$$v_i(1 - t_i) - p_i \geq v_i(1 - t'_i) - p'_i$$
- Individual rationality: $v_i(1 - t_i) - p_i \geq 0$
- Ex post budget balance:
  *If the project is not built*, then no agent can consume the project and no agent pays. That is, we must have $t_i = 1$ and $p_i = 0$ for all $i$.
  *If the project is built*, then the agents' total payment must cover exactly the project cost. That is, $\sum_i p_i = 1$.

Our aim is to design mechanisms that minimize the following design objectives:

- Expected *Max-Delay*: $E_{v_i \sim f}\left(\max\{t_1, t_2, \ldots, t_n\}\right)$
- Expected *Sum-Delay*: $E_{v_i \sim f}\left(\sum_i t_i\right)$

## 4 Single Deadline Mechanisms

We first describe the *serial cost sharing mechanism (SCS)* proposed by Moulin [7]. Under SCS, an agent's release time is either 0 or 1.[2]

Let $\vec{v}$ be the type profile. We first define the following functions:

$$I(\vec{v}) = \begin{cases} 1 & \exists k \in \{1, 2, \ldots, n\}, k \leq |\{v_i | v_i \geq \frac{1}{k}\}| \\ 0 & \text{otherwise} \end{cases}$$

$I(\vec{v})$ equals 1 if and only if there exist at least $k$ values among $\vec{v}$ that are at least $\frac{1}{k}$, where $k$ is an integer from 1 to $n$.

$$K(\vec{v}) = \begin{cases} \max\{k | k \leq |\{v_i | v_i \geq \frac{1}{k}\}|, k \in \{1, 2, \ldots, n\}\} & I(\vec{v}) = 1 \\ 0 & I(\vec{v}) = 0 \end{cases}$$

Given $\vec{v}$, there could be multiple values for $k$, where there exist at least $k$ values among $\vec{v}$ that are at least $\frac{1}{k}$. $K(\vec{v})$ is the largest value for $k$. If such a $k$ value does not exist, then $K(\vec{v})$ is set to 0.

**Definition 1 (Serial Cost Sharing Mechanism [7]).** *Given $\vec{v}$, let $k = K(\vec{v})$.*

- *If $k > 0$, then agents with the highest $k$ values are the consumers. The consumers pay $\frac{1}{k}$. The non-consumers do not pay.*
- *If $k = 0$, then there are no consumers and no agents pay.*

Essentially, the serial cost sharing mechanism finds the largest $k$ where $k$ agents are willing to equally split the cost. If such a $k$ exists, then we say *the cost share is successful* and these $k$ agents are *joining the cost share*. If such a $k$ does not exist, then we say *the cost share failed*.

Next we introduce a new mechanism family called the single deadline mechanisms.

**Definition 2 (Single Deadline Mechanisms).**

*A single deadline mechanism is characterized by one parameter $d \in [0, 1]$. $d$ is called the mechanism's* **deadline***. We use $M(d)$ to denote the single deadline mechanism with deadline $d$.*

*The time interval before the deadline $[0, d]$ is called the* **non-free** *part. The time interval after the deadline $[d, 1]$ is called the* **free** *part.*

*We run the serial cost sharing mechanism on the non-free part as follows. For the non-free part, the agents' valuations are $d\vec{v} = (dv_1, \ldots, dv_n)$. Let $k = K(d\vec{v})$. Agents with the highest $k$ values get to consume the non-free part, and each needs to pay $\frac{1}{k}$.*

*The free part is allocated to the agents for free. However, we cannot give out the free part if the public project is not built.*

*If we give out the free part if and only if $I(d\vec{v}) = 1$, then the mechanism is not strategy-proof, because the free parts change the agents' strategies.*[3] *Instead, we give*

[2] Because the concept of release time does not exist in the classic binary excludable public project model.

[3] For example, an agent may over-report to turn an unsuccessful cost share into a successful cost share, in order to claim the free part.

*agent $i$ her free part if and only if $I(dv_{-i}) = 1$. That is, agent $i$ gets her free part if and only if the other agents can successfully cost share the non-free part without $i$.*

*If an agent receives both the non-free part and the free part, then her release time is $0$. If an agent only receives the free part, then her release time is $d$. If an agent does not receive either part, then her release time is $1$. Lastly, if an agent only receives the non-free part, then her release time is $1 - d$, because such an agent's consumption interval should have length $d$ (*i.e.*, $[1 - d, 1]$).*

**Proposition 1.** *The single deadline mechanisms are strategy-proof, individually rational, and ex post budget balanced.*

## 5 Max-Delay: Asymptotic Optimality

**Theorem 1.** *The optimal single deadline mechanism's expected max-delay approaches $0$ when the number of agents approaches infinity.*

*Proof.* We consider a single deadline mechanism $M(d)$. Every agent's valuation is drawn *i.i.d.* from a distribution with PDF $f$. Let $V_i$ be the random variable representing agent $i$'s valuation. Since $f$ is positive and Lipschitz continuous, we have that $\forall d, \exists k, P(dV_i \geq \frac{1}{k}) > 0$. That is, for any deadline $d$, there always exists an integer $k$, where the probability that an agent is willing to pay $\frac{1}{k}$ for the non-free part is positive. Let $p = P(dV_i \geq \frac{1}{k})$. We define the following Bernoulli random variable:

$$B_i = \begin{cases} 1 & dV_i \geq \frac{1}{k} \\ 0 & \text{otherwise} \end{cases}$$

$B_i$ equals $1$ with probability $p$. It equals $1$ if and only if agent $i$ can afford $\frac{1}{k}$ for the non-free part. The total number of agents in $\vec{v}$ who can afford $\frac{1}{k}$ for the non-free part then follows a Binomial distribution $B(n, p)$. We use $B$ to denote this Binomial variable. If $B \geq k + 1$, then every agent receives the free part, because agent $i$ receives the free part if excluding herself, there are at least $k$ agents who are willing to pay $\frac{1}{k}$ for the non-free part. The probability that the max-delay is higher than $d$ is therefore bounded above by $P(B \leq k)$. According to Hoeffding's inequality, when $k < np$, $P(B \leq k) \leq e^{-2n\left(p - \frac{k}{n}\right)^2}$. We immediately have that when $n$ approaches infinity, the probability that the max-delay is higher than $d$ is approaching $0$. Since $d$ is arbitrary, we have that asymptotically, the single deadline mechanism's expected max-delay is approaching $0$.

Next, we use an example to show that when $n = 500$, the optimal single deadline mechanism's expected max-delay is close to $0.01$. We reuse all notation defined in the proof of Theorem 1. We make use of the Chernoff bound. When $k < np$, we have $P(B \leq k) \leq e^{-nD\left(\frac{k}{n}||p\right)}$, where $D\left(a||p\right) = a \ln \frac{a}{p} + (1 - a) \ln \frac{1-a}{1-p}$.

When all agents receive the free part, the max-delay is at most $d$. Otherwise, the max-delay is at most $1$. The expected max-delay is at most

$$P(B \leq k) + d(1 - P(B \leq k)) \leq P(B \leq k) + d$$

*Example 1.* Let us consider a case where $n = 500$. We set $d = 0.01$ and $k = 250$.

- $f$ is the uniform distribution $U(0, 1)$: We have $p = 0.6$ and $P(B \leq 250) \leq 3.69\text{e}-5$. $M(0.01)$'s expected max-delay is then bounded above by $0.01 + 3.69\text{e} - 5$.
- $f$ is the normal distribution $N(0.5, 0.1)$ restricted to $[0, 1]$: We have $p = 0.84$ and $P(B \leq 250) \leq 7.45\text{e} - 69$. $M(0.01)$'s expected max-delay is then bounded above by $0.01 + 7.45\text{e} - 69$.

On the contrary, the expected max-delay of the serial cost sharing mechanism is not approaching 0 asymptotically. For example, when $n = 500$, under $U(0, 1)$, the expected max-delay of the serial cost sharing mechanism equals $0.632$.

## 6 Sum-Delay: Asymptotic Optimality

**Theorem 2.** *When the number of agents approaches infinity, the optimal single deadline mechanism is optimal among all mechanisms in terms of expected sum-delay.*

Theorem 2 can be proved by combining Proposition 4 and Proposition 5.

**Proposition 2.** *The optimal expected sum-delay is finite regardless of the distribution.*

*Proof.* We consider the following mechanism: Pick an arbitrary integer $k > 1$. We offer $\frac{1}{k}$ to the agents one by one. An agent gets the whole interval $[0, 1]$ if she agrees to pay $\frac{1}{k}$ and if the project is built. Otherwise, she gets nothing. We build the project only when $k$ agents agree. Since we approach the agents one by one, after $k$ agents agree to pay $\frac{1}{k}$, all future agents receive the whole interval for free. This mechanism's expected sum-delay is bounded above by a constant. The constant only depends on the distribution.

The following proposition follows from Proposition 2.

**Proposition 3.** *Given a mechanism $M$ and the number of agents $n$, let $Fail(n)$ be the probability of not building under $M$. We only need to consider $M$ that satisfies $Fail(n) = O(1/n)$.*

We then propose a relaxed version of the ex post budget balance constraint, and use it to calculate the delay lower bound.

**Definition 3 (Ex ante budget balance).** *Mechanism $M$ is ex ante budget balanced if and only if the expected total payment from the agents equals the probability of building (times project cost 1).*

**Proposition 4.** *Let $Fail(n)$ be the probability of not building the project when there are $n$ agents. We consider what happens when we offer $o$ for the whole interval $[0, 1]$ to an individual agent. If the agent accepts $o$ then she pays $o$ and gets the whole interval. Otherwise, the agent pays 0 and receives nothing.*

*We define the* delay versus payment ratio $r(o)$ *as follows:*

$$r(o) = \frac{\int_0^o f(x)dx}{o\int_o^1 f(x)dx}$$

*$r$ is continuous on $(0,1)$. Due to $f$ being Lipschitz continuous, we have $\lim_{o\to 0} r(o) = f(0)$ and $\lim_{o\to 1} r(o) = \infty$.*[4] *We could simply set $r(0) = f(0)$, then $r$ is continuous on $[0,1)$. We define the* optimal delay versus payment ratio *$r^* = \min_{o\in[0,1)} r(o)$.*

*The expected sum-delay is bounded below by $r^*(1 - Fail(n))$, which approaches $r^*$ asymptotically according to Proposition 3.*

**Proposition 5.** *Let $o^*$ be the optimal offer that leads to the optimal delay versus payment ratio $r^*$.*[5]

$$o^* = \arg\min_{o\in[0,1)} r(o)$$

*Let $\epsilon > 0$ be an arbitrarily small constant. The following single deadline mechanism's expected sum delay approaches $r^*(1+\epsilon)$ asymptotically.*

$$M(\frac{1+\epsilon}{no^* \int_{o^*}^1 f(x)dx})$$

We then use an example to show that when $n = 500$, under different distributions, the optimal single deadline mechanism's expected sum-delay is close to optimality.

*Example 2.* We consider $n = 500$ which is the same as Example 1. Simulations are based on $100,000$ random draws.

- $f$ is the uniform distribution $U(0,1)$: The single deadline mechanism $M(1)$ (essentially the serial cost sharing mechanism) has an expected sum-delay of $1.006$, which is calculated via numerical simulation. $Fail(500)$ is then at most $0.002$. $r^* = 1$. The lower bound is $0.998$, which is close to our achieved sum-delay $1.006$.
- $f$ is the normal distribution $N(0.5, 0.1)$ restricted to $[0,1]$: The single deadline mechanism $M(1)$'s expected sum-delay equals $2.3\text{e}-4$ in simulation, which is obviously close to optimality.
- $f$ is the beta distribution $Beta(0.5, 0.5)$: The single deadline mechanism $M(0.01)$'s expected sum-delay equals $1.935$ in simulation. $Fail(500)$ is then at most $0.00387$. $r^* = 1.927$. The lower bound equals $(1 - 0.00387) * r^* = 1.920$, which is very close to the achieved sum-delay of $1.935$. The serial cost sharing mechanism $M(1)$ is far away from optimality in this example. The expected sum-delay of the serial cost sharing mechanism is much larger at $14.48$.

## 7 Automated Mechanism Design for Smaller Number of Agents

For smaller number of agents, the single deadline mechanism family no longer contains a near optimal mechanism. We propose two numerical methods for identifying better mechanisms for smaller number of agents. One is by extending the single deadline mechanism family and the other is via evolutionary computation.

[4] When $o$ approaches 0, $r(o)$'s numerator is approaching $of(0)$ while the denominator is approaching $o$.

[5] If $o^* = 0$, then we replace it with an infinitesimally small $\gamma > 0$. The achieved sum-delay is then approaching $r(\gamma)(1+\epsilon)$ asymptotically. When $\gamma$ approaches 0, $r(\gamma)$ approaches $r^*$.

**Definition 4 (Multiple Deadline Mechanisms).** *A multiple deadline mechanism $M(d_1, \ldots, d_n)$ is characterized by $n$ different deadlines. Agent $i$'s non-free part is $[0, d_i]$ and her free part is $[d_i, 1]$. The mechanism's rules are otherwise identical to the single deadline mechanisms.*

We simply use exhaustive search to find the best set of deadlines. Obviously, this approach only works when the number of agents is tiny. We then present an Automated Mechanism Design approach based on evolutionary computation.

Ohseto [8] characterized all strategy-proof and individually rational mechanisms for the binary public project model (under several minor technical assumptions). We summarize the author's characterization as follows:

- *Unanimous mechanisms* (characterization for the nonexcludable model): Under an unanimous mechanism, there is a cost share vector $(c_1, c_2, \ldots, c_n)$ with $c_i \geq 0$ and $\sum_i c_i = 1$. The project is built if and only if all agents accept this cost share vector.
- *Largest unanimous mechanisms* (characterization for the excludable model): Under a largest unanimous mechanism, for every subset/coalition of the agents, there is a constant cost share vector. The agents initially face the cost share vector corresponding to the grand coalition. If some agents do not accept the current cost share vector, then they are forever excluded. The remaining agents face a different cost share vector based on who are left. If at some point, all remaining agents accept, then we build the project. Otherwise, the project is not built.

We extend the largest unanimous mechanisms by adding the *release time* element.

**Definition 5 (Sequential unanimous mechanisms).** *A cost share vector under a sequential unanimous mechanism includes both the payments and the release time:*

$$T_1, B_1, \quad T_2, B_2, \quad \ldots, \quad T_n, B_n$$

*Agent $i$ accepts the above cost share vector if and only if her utility* based on her reported valuation *is nonnegative when paying $B_i$ for the time interval $[T_i, 1]$. That is, agent $i$ accepts the above cost share vector if and only if* her reported valuation *is at least $\frac{B_i}{1-T_i}$. $\frac{B_i}{1-T_i}$ is called the* unit price *agent $i$ faces. We require $B_i \geq 0$ and $\sum_i B_i = 1$.*

*A sequential unanimous mechanism contains $m$ cost share vectors in a sequence. The mechanism goes through the sequence and stops at the first vector that is accepted by all agents. The project is built and the agents' release time and payments are determined by the unanimously accepted cost share vector. If all cost share vectors in the sequence are rejected, then the decision is not to build.*

The largest unanimous mechanisms (can be interpreted as special cases with binary $T_i$) form a subset of the sequential unanimous mechanisms. The sequential unanimous mechanisms' structure makes it suitable for genetic algorithms — we treat the cost share vectors as the *genes* and treat the sequences of cost share vectors as the *gene sequences*.

The sequential unanimous mechanisms are generally not strategy-proof. However, they can be easily proved to be strategy-proof in two scenarios:

- A sequential unanimous mechanism is strategy-proof when *the sequence contains only one cost share vector* (an agent faces a take-it-or-leave-it offer). This observation makes it easy to generate an initial population of strategy-proof mechanisms.
- If for every agent, as we go through the cost share vector sequence, the unit price an agent faces is *nondecreasing* and her release time is also *nondecreasing*, then the mechanism is strategy-proof. Essentially, when the above is satisfied, all agents prefer earlier cost share vectors. All agents are incentivized to report truthfully, as doing so enables them to secure the earliest possible cost share vector.

The sequential unanimous mechanism family *seems* to be quite expressive.[6] Our experiments show that by optimizing within the sequential unanimous mechanisms, we are able to identify mechanisms that perform better than existing mechanisms. Our approach is as follows:

- Initial population contains $200$ strategy-proof mechanisms. Every initial mechanism is a sequential unanimous mechanism with only one cost share vector. The $B_i$ and the $T_i$ are randomly generated by sampling $U(0, 1)$.
- We perform evolution for $200$ rounds. Before each round, we filter out mechanisms that are not truthful. We have two different filters:
  - Strict filter: we enforce that every agent's unit price faced and release time must be nondecreasing. With this filter, the final mechanism produced must be strategy-proof. We call this variant the *Truthful Genetic Algorithm (TGA)*.
  - Loose filter: we use simulation to check for strategy-proofness violations. In every evolution round, we generate $200$ random type profiles. For each type profile and each agent, we randomly draw one false report and we filter out a mechanism if any beneficial manipulation occurs. After finishing evolution, we use $10,000$ type profiles to filter out the untruthful mechanisms from the final population. It should be noted that, we can only claim that the remaining mechanisms are *probably* truthful. We call this variant the *Approximately Truthful Genetic Algorithm (ATGA)*.
- We perform crossover and mutations as follows:
  - Crossover: We call the top $50\%$ of the population (in terms of fitness, *i.e.*, expected max-delay or sum-delay) the *elite population*. For every elite mechanism, we randomly pick another mechanism from the whole population, and perform a crossover by randomly swapping one gene segment.
  - Mutation: For every elite mechanism, with $20\%$ chance, we randomly select one gene, modify the offer of one agent by making it worse. We insert that new cost share vector into a random position after the original position.
  - Neighbourhood Search: For every elite mechanism, with $20\%$ chance, we randomly perturb one gene uniformly (from $-10\%$ to $+10\%$).

[6] Let $M$ be a strategy-proof mechanism. There exists a sequential unanimous mechanism $M'$ (with exponential sequence length). $M'$ has an approximate equilibrium where the equilibrium outcome is arbitrarily close to $M$'s outcome. To prove this, we only need to discretize an individual agent's type space $[0, 1]$ into a finite number of grid points. The number of type profiles is exponential. We place $M$'s outcomes for all these type profiles in a sequence.

- Abandon duplication and unused genes: In every evolution round, if a cost share vector is never unanimously accepted or if two cost share vectors are within $0.0001$ in L1 distance. then we remove the duplication/unused genes.

### 7.1 Experiments

We present the expected max-delay and sum-delay for $n = 3, 5$ and for different distributions. ATGA is only approximately truthful. We recall that in our evolutionary process, in each round, we only use a very loose filter to filter out the untruthful mechanisms. After evolution finishes, we run a more rigorous filter on the final population (based on $10,000$ randomly generated type profiles). The percentage in the parenthesis is the percentage of mechanisms surviving the more rigorous test. The other mechanisms (TGA and Single/Multiple deadlines) are strategy-proof. SCS is the serial cost sharing mechanism from Moulin [7], which has the best known expected delays [6].

| ***n=3,sum-delay*** | ATGA | TGA | Single deadline | Multiple deadline | SCS |
|---|---|---|---|---|---|
| **Uniform(0,1)** | **1.605(95%)** | **1.605** | **1.605** | **1.605** | **1.605** |
| **Beta(0.5,0.5)** | **1.756(89%)** | **1.757** | **1.757** | **1.757** | **1.757** |
| **Bernoulli(0.5)** | **0.869(100%)** | **0.868** | 1.499 | 1.253 | 1.498 |
| **50% 0, 50% 0.8** | **1.699(98%)** | 1.873 | 1.873 | 1.873 | 1.873 |

| ***n=3,max-delay*** | ATGA | TGA | Single deadline | Multiple deadline | SCS |
|---|---|---|---|---|---|
| **Uniform(0,1)** | **0.705(97%)** | **0.705** | **0.705** | **0.705** | **0.705** |
| **Beta(0.5,0.5)** | **0.754(87%)** | 0.757 | 0.782 | 0.757 | 0.782 |
| **Bernoulli(0.5)** | **0.5(100%)** | **0.498** | 0.687 | **0.50** | 0.877 |
| **50% 0, 50% 0.8** | **0.676(94%)** | 0.753 | 0.749 | 0.749 | 0.877 |

| ***n=5,sum-delay*** | ATGA | TGA | Single deadline | Multiple deadline | SCS |
|---|---|---|---|---|---|
| **Uniform(0,1)** | 1.462(95%) | 1.503 | **1.415** | **1.415** | **1.415** |
| **Beta(0.5,0.5)** | 2.279(92%) | 2.12 | **1.955** | **1.955** | **1.955** |
| **Bernoulli(0.5)** | **1.146(100%)** | 1.867 | 2.106 | 1.711 | 2.523 |
| **50% 0, 50% 0.8** | 2.432(94%) | 2.845 | 2.323 | **2.248** | 2.667 |

| ***n=5,max-delay*** | ATGA | TGA | Single deadline | Multiple deadline | SCS |
|---|---|---|---|---|---|
| **Uniform(0,1)** | 0.677(91%) | 0.677 | **0.662** | **0.662** | 0.678 |
| **Beta(0.5,0.5)** | 0.754(79%) | 0.75 | **0.73** | **0.73** | 0.827 |
| **Bernoulli(0.5)** | 0.506(100%) | **0.50** | 0.577 | **0.50** | 0.971 |
| **50% 0, 50% 0.8** | **0.666(80%)** | 0.751 | 0.736 | 0.679 | 0.968 |

**Table 1.** We see that ATGA performs well in many settings. If we focus on *provable* strategy-proof mechanisms, then TGA and the optimal multiple deadline mechanism also often perform better than the serial cost sharing mechanism.

## References


1. Conitzer, V., Sandholm, T.: Complexity of mechanism design. In: Darwiche, A., Friedman, N. (eds.) UAI '02, Proceedings of the 18th Conference in Uncertainty in Artificial Intelligence, University of Alberta, Edmonton, Alberta, Canada, August 1-4, 2002. pp. 103–110. Morgan Kaufmann (2002)
2. Deb, R., Razzolini, L.: Voluntary cost sharing for an excludable public project. Mathematical Social Sciences **37**(2), 123 – 138 (1999)
3. Dütting, P., Feng, Z., Narasimhan, H., Parkes, D., Ravindranath, S.S.: Optimal auctions through deep learning. In: International Conference on Machine Learning. pp. 1706–1715. PMLR (2019)
4. Fisher, D.: Vupen founder launches new zero-day acquisition firm zerodium (2015), july 24, 2015 online: https://threatpost.com/vupen-launches-new-zero-day-acquisition-firm-zerodium/113933/
5. Greenberg, A.: Shopping for zero-days: A price list for hackers' secret software exploits (2012), march 23, 2012 online: http://www.forbes.com/sites/andygreenberg/2012/03/23/shopping-for-zero-days-an-price-list-for-hackers-secret-software-exploits/
6. Guo, M., Yang, Y., Ali Babar, M.: Cost sharing security information with minimal release delay. In: PRIMA 2018: Principles and Practice of Multi-Agent Systems. pp. 177–193. Springer International Publishing, Cham (2018)
7. Moulin, H.: Serial cost-sharing of excludable public goods. The Review of Economic Studies **61**(2), 305–325 (1994)
8. Ohseto, S.: Characterizations of strategy-proof mechanisms for excludable versus nonexcludable public projects. Games and Economic Behavior **32**(1), 51 – 66 (2000)
9. Phelps, S., McBurney, P., Parsons, S.: Evolutionary mechanism design: a review. Autonomous agents and multi-agent systems **21**(2), 237–264 (2010)
10. Shen, W., Tang, P., Zuo, S.: Automated mechanism design via neural networks. In: Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems. pp. 215–223. AAMAS '19, International Foundation for Autonomous Agents and Multiagent Systems, Richland, SC (2019)
11. Wang, G., Guo, R., Sakurai, Y., Babar, A., Guo, M.: Mechanism design for public projects via neural networks. In: 20th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2021, online) (2021)