

Audit Games

Jeremiah Blocki, Nicolas Christin, Anupam Datta, Ariel D. Procaccia, Arunesh Sinha

Carnegie Mellon University, Pittsburgh, USA

{jblocki@cs., nicolasc@, danupam@, arielpro@cs., aruneshs@}cmu.edu

Abstract

Effective enforcement of laws and policies requires expending resources to prevent and detect offenders, as well as appropriate punishment schemes to deter violators. In particular, enforcement of privacy laws and policies in modern organizations that hold large volumes of personal information (e.g., hospitals, banks, and Web services providers) relies heavily on internal audit mechanisms. We study economic considerations in the design of these mechanisms, focusing in particular on effective resource allocation and appropriate punishment schemes. We present an audit game model that is a natural generalization of a standard security game model for resource allocation with an additional punishment parameter. Computing the Stackelberg equilibrium for this game is challenging because it involves solving an optimization problem with non-convex quadratic constraints. We present an additive FPTAS that efficiently computes a solution that is arbitrarily close to the optimal solution.

1 Introduction

In a seminal paper, Gary Becker [1968] presented a compelling economic treatment of crime and punishment. He demonstrated that effective law enforcement involves optimal resource allocation to prevent and detect violations, coupled with appropriate punishments for offenders. He described how to optimize resource allocation by balancing the societal cost of crime and the cost incurred by prevention, detection and punishment schemes. While Becker focused on crime and punishment in society, similar economic considerations guide enforcement of a wide range of policies. In this paper, we study effective enforcement mechanisms for this broader set of policies. Our study differs from Becker’s in two significant ways—our model accounts for *strategic interaction* between the enforcer (or defender) and the adversary; and we design efficient algorithms for *computing* the optimal resource allocation for prevention or detection measures as well as punishments. At the same time, our model is significantly less nuanced than Becker’s, thus enabling the algorithmic development and raising interesting questions for further work.

A motivating application for our work is auditing, which typically involves detection and punishment of policy violators. In particular, enforcement of privacy laws and policies in modern organizations that hold large volumes of personal information (e.g., hospitals, banks, and Web services providers like Google and Facebook) relies heavily on internal audit mechanisms. Audits are also common in the financial sector (e.g., to identify fraudulent transactions), in internal revenue services (e.g., to detect tax evasion), and in traditional law enforcement (e.g., to catch speed limit violators).

The audit process is an interaction between two agents: a defender (auditor) and an adversary (auditee). As an example, consider a hospital (defender) auditing its employee (adversary) to detect privacy violations committed by the employee when accessing personal health records of patients. While privacy violations are costly for the hospital as they result in reputation loss and require expensive measures (such as privacy breach notifications), audit inspections also cost money (e.g., the cost of the human auditor involved in the investigation). Moreover, the number and type of privacy violations depend on the actions of the *rational* auditee—employees commit violations that benefit them.

1.1 Our Model

We model the audit process as a game between a defender (e.g. a hospital) and an adversary (e.g., an employee). The defender audits a given set of targets (e.g., health record accesses) and the adversary chooses a target to attack. The defender’s action space in the audit game includes two components. First, the allocation of its inspection resources to targets; this component also exists in a standard model of security games [Tambe, 2011]. Second, we introduce a continuous punishment rate parameter that the defender employs to deter the adversary from committing violations. However, punishments are not free and the defender incurs a cost for choosing a high punishment level. For instance, a negative work environment in a hospital with high fines for violations can lead to a loss of productivity (see [Becker, 1968] for a similar account of the cost of punishment). The adversary’s utility includes the benefit from committing violations and the loss from being punished if caught by the defender. Our model is parametric in the utility functions. Thus, depending on the application, we can instantiate the model to either allocate resources for detecting violations or preventing them.

This generality implies that our model can be used to study all the applications previously described in the security games literature [Tambe, 2011].

To analyze the audit game, we use the Stackelberg equilibrium solution concept [von Stackelberg, 1934] in which the defender commits to a strategy, and the adversary plays an optimal response to that strategy. This concept captures situations in which the adversary learns the defender’s audit strategy through surveillance or the defender publishes its audit algorithm. In addition to yielding a better payoff for the defender than any Nash equilibrium, the Stackelberg equilibrium makes the choice for the adversary simple, which leads to a more predictable outcome of the game. Furthermore, this equilibrium concept respects the computer security principle of avoiding “security through obscurity”—audit mechanisms like cryptographic algorithms should provide security despite being publicly known.

1.2 Our Results

Our approach to computing the Stackelberg equilibrium is based on the multiple LPs technique of Conitzer and Sandholm [2006]. However, due to the effect of the punishment rate on the adversary’s utility, the optimization problem in audit games has quadratic and non-convex constraints. The non-convexity does not allow us to use any convex optimization methods, and in general polynomial time solutions for a broad class of non-convex optimization problems are not known [Neumaier, 2004].

However, we demonstrate that we can efficiently obtain an additive approximation to our problem. Specifically, we present an additive fully polynomial time approximation scheme (FPTAS) to solve the audit game optimization problem. Our algorithm provides a K -bit precise output in time polynomial in K . Also, if the solution is rational, our algorithm provides an exact solution in polynomial time. In general, the exact solution may be irrational and may not be representable in a finite amount of time.

1.3 Related Work

Our audit game model is closely related to security games [Tambe, 2011]. There are many papers (see, e.g., [Korzhyk *et al.*, 2010; Pita *et al.*, 2011; 2008]) on security games, and as our model adds the additional continuous punishment parameter, all the variations presented in these papers can be studied in the context of audit games (see Section 4). However, the audit game solution is technically more challenging as it involves non-convex constraints.

An extensive body of work on auditing focuses on analyzing logs for detecting and explaining violations using techniques based on logic [Vaughan *et al.*, 2008; Garg *et al.*, 2011] and machine learning [Zheng *et al.*, 2006; Bodik *et al.*, 2010]. In contrast, very few papers study economic considerations in auditing strategic adversaries. Our work is inspired in part by the model proposed in one such paper [Blocki *et al.*, 2012], which also takes the point of view of commitment and Stackelberg equilibria to study auditing. However, the emphasis in that work is on developing a detailed model and using it to predict observed audit practices in industry and the effect of public policy interventions on auditing practices.

They do not present efficient algorithms for computing the optimal audit strategy. In contrast, we work with a more general and simpler model and present an efficient algorithm for computing an approximately optimal audit strategy. Furthermore, since our model is related to the security game model, it opens up the possibility to leverage existing algorithms for that model and apply the results to the interesting applications explored with security games.

Zhao and Johnson [2008] model a specific audit strategy—“break the glass”—where agents are permitted to violate an access control policy at their discretion (e.g., in an emergency situation in a hospital), but these actions are audited. They manually analyze specific utility functions and obtain closed-form solutions for the audit strategy that results in a Stackelberg equilibrium. In contrast, our results apply to any utility function and we present an efficient algorithm for computing the audit strategy.

2 The Audit Game Model

The audit game features two players: the defender (D), and the adversary (A). The defender wants to audit n targets t_1, \dots, t_n , but has limited resources which allow for auditing only one of the n targets. Thus, a pure action of the defender is to choose which target to audit. A randomized strategy is a vector of probabilities p_1, \dots, p_n of each target being audited. The adversary attacks one target such that given the defender’s strategy the adversary’s choice of violation is the best response.

Let the utility of the defender be $U_D^a(t_i)$ when audited target t_i was found to be attacked, and $U_D^u(t_i)$ when unaudited target t_i was found to be attacked. The attacks (violation) on unaudited targets are discovered by an external source (e.g. government, investigative journalists,...). Similarly, define the utility of the attacker as $U_A^a(t_i)$ when the attacked target t_i is audited, and $U_A^u(t_i)$ when attacked target t_i is not audited, excluding any punishment imposed by the defender. Attacks discovered externally are costly for the defender, thus, $U_D^a(t_i) > U_D^u(t_i)$. Similarly, attacks not discovered by internal audits are more beneficial to the attacker, and $U_A^u(t_i) > U_A^a(t_i)$.

The model presented so far is identical to security games with singleton and homogeneous schedules, and a single resource [Korzhyk *et al.*, 2010]. The additional component in audit games is punishment. The defender chooses a punishment “rate” $x \in [0, 1]$ such that if auditing detects an attack, the attacker is fined an amount x . However, punishment is not free—the defender incurs a cost for punishing, e.g., for creating a fearful environment. For ease of exposition, we model this cost as a linear function ax , where $a > 0$; however, our results directly extend to any cost function polynomial in x . Assuming $x \in [0, 1]$ is also without loss of generality as utilities can be scaled to be comparable to x . We do assume the punishment rate is fixed and deterministic; this is only natural as it must correspond to a consistent policy.

We can now define the full utility functions. Given probabilities p_1, \dots, p_n of each target being audited, the utility of the defender when target t_* is attacked is

$$p_*U_D^a(t_*) + (1 - p_*)U_D^u(t_*) - ax.$$

The defender pays a fixed cost ax regardless of the outcome. In the same scenario, the utility of the attacker when target t_* is attacked is

$$p_*(U_A^u(t_*) - x) + (1 - p_*)U_A^u(t_*).$$

The attacker suffers the punishment x only when attacking an audited target.

Equilibrium. The Stackelberg equilibrium solution involves a commitment by the defender to a strategy (with a possibly randomized allocation of the resource), followed by the best response of the adversary. The mathematical problem involves solving multiple optimization problems, one each for the case when attacking t_* is in fact the best response of the adversary. Thus, assuming t_* is the best response of the adversary, the $*^{th}$ optimization problem P_* in audit games is

$$\begin{aligned} \max_{p_i, x} \quad & p_* U_D^a(t_*) + (1 - p_*) U_D^u(t_*) - ax, \\ \text{subject to} \quad & \forall i \neq *. \ p_i (U_A^a(t_i) - x) + (1 - p_i) U_A^u(t_i) \\ & \leq p_* (U_A^a(t_*) - x) + (1 - p_*) U_A^u(t_*), \\ & \forall i. \ 0 \leq p_i \leq 1, \\ & \sum_i p_i = 1, \\ & 0 \leq x \leq 1. \end{aligned}$$

The first constraint verifies that attacking t_* is indeed a best response. The auditor then solves the n problems P_1, \dots, P_n (which correspond to the cases where the best response is t_1, \dots, t_n , respectively), and chooses the best solution among all these solutions to obtain the final strategy to be used for auditing. This is a generalization of the multiple LPs approach of Conitzer and Sandholm [2006].

Inputs. The inputs to the above problem are specified in K bit precision. Thus, the total length of all inputs is $O(nK)$. Also, the model can be made more flexible by including a dummy target for which all associated costs are zero (including punishment); such a target models the possibility that the adversary does not attack any target (no violation). Our result stays the same with such a dummy target, but, an additional edge case needs to be handled—we discuss this case in a remark at the end of Section 3.2.

3 Computing an Audit Strategy

Because the indices of the set of targets can be arbitrarily permuted, without loss of generality we focus on one optimization problem P_n ($*$ = n) from the multiple optimization problems presented in Section 2. The problem has quadratic and non-convex constraints. The non-convexity can be readily checked by writing the constraints in matrix form, with a symmetric matrix for the quadratic terms; this quadratic-term matrix is indefinite.

However, for a fixed x , the induced problem is a linear programming problem. It is therefore tempting to attempt a binary search over values of x . This naïve approach does not work, because the solution may not be single-peaked in the values of x , and hence choosing the right starting point for the binary search is a difficult problem. Another naïve approach is to discretize the interval $[0, 1]$ into steps of ϵ' , solve the resultant LP for the $1/\epsilon'$ many discrete values of x , and

then choose the best solution. As an LP can be solved in polynomial time, the running time of this approach is polynomial in $1/\epsilon'$, but the approximation factor is at least $a\epsilon'$ (due to the ax in the objective). Since a can be as large as 2^K , getting an ϵ -approximation requires ϵ' to be $2^{-K}\epsilon$, which makes the running time exponential in K . Thus, this scheme cannot yield an FPTAS.

3.1 High-Level Overview

Fortunately, the problem P_n has another property that allows for efficient methods. Let us rewrite P_n in a more compact form. Let $\Delta_{D,i} = U_D^a(t_i) - U_D^u(t_i)$, $\Delta_i = U_A^a(t_i) - U_A^u(t_i)$ and $\delta_{i,j} = U_A^u(t_i) - U_A^u(t_j)$. $\Delta_{D,i}$ and Δ_i are always positive, and P_n reduces to:

$$\begin{aligned} \max_{p_i, x} \quad & p_n \Delta_{D,n} + U_D^u(t_n) - ax, \\ \text{subject to} \quad & \forall i \neq n. \ p_i (-x - \Delta_i) + p_n (x + \Delta_n) + \delta_{i,n} \leq 0, \\ & \forall i. \ 0 \leq p_i \leq 1, \\ & \sum_i p_i = 1, \\ & 0 \leq x \leq 1. \end{aligned}$$

The main observation that allows for polynomial time approximation is that, at the optimal solution point, the quadratic constraints can be partitioned into a) those that are tight, and b) those in which the probability variables p_i are zero (Lemma 1). Each quadratic constraint corresponding to p_i can be characterized by the curve $p_n(x + \Delta_n) + \delta_{i,n} = 0$. The quadratic constraints are thus parallel hyperbolic curves on the (p_n, x) plane; see Figure 1 for an illustration. The optimal values p_n^o, x^o partition the constraints (equivalently, the curves): the constraints lying below the optimal value are tight, and in the constraints above the optimal value the probability variable p_i is zero (Lemma 2). The partitioning allows a linear number of iterations in the search for the solution, with each iteration assuming that the optimal solution lies between adjacent curves and then solving the sub-problem with equality quadratic constraints.

Next, we reduce the problem with equality quadratic constraints to a problem with two variables, exploiting the nature of the constraints themselves, along with the fact that the objective has only two variables. The two-variable problem can be further reduced to a single-variable objective using an equality constraint, and elementary calculus then reduces the problem to finding the roots of a polynomial. Finally, we use known results to find approximate values of irrational roots.

3.2 Algorithm and Main Result

The main result of our paper is the following theorem:

Theorem 1. *Problem P_n can be approximated to an additive ϵ in time $O(n^5 K + n^4 \log(\frac{1}{\epsilon}))$ using the splitting circle method [Schönhage, 1982] for approximating roots.*

Remark The technique of Lenstra et al. [1982] can be used to exactly compute rational roots. Employing it in conjunction with the splitting circle method yields a time bound $O(\max\{n^{13} K^3, n^5 K + n^4 \log(1/\epsilon)\})$. Also, this technique finds an exact optimal solution if the solution is rational.

Before presenting our algorithm we state two results about the optimization problem P_n that motivate the algorithm and

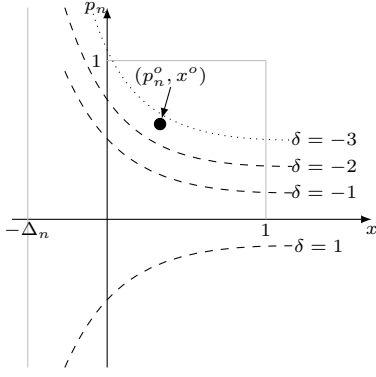


Figure 1: The quadratic constraints are partitioned into those below (p_n^o, x^o) that are tight (dashed curves), and those above (p_n^o, x^o) where $p_i = 0$ (dotted curves).

are also used in the correctness analysis. The proof of the first lemma is omitted due to lack of space.

Lemma 1. *Let p_n^o, x^o be the optimal solution. Assume $x^o > 0$ and $p_n^o < 1$. Then, at p_n^o, x^o , for all $i \neq n$, either $p_i = 0$ or $p_n^o(x^o + \Delta_n) + \delta_{i,n} = p_i(x^o + \Delta_i)$, i.e., the i^{th} quadratic constraint is tight.*

Lemma 2. *Assume $x^o > 0$ and $p_n^o < 1$. Let $p_n^o(x^o + \Delta_n) + \delta = 0$. If for some i , $\delta_{i,n} < \delta$ then $p_i = 0$. If for some i , $\delta_{i,n} > \delta$ then $p_n^o(x^o + \Delta_n) + \delta_{i,n} = p_i(x^o + \Delta_i)$. If for some i , $\delta_{i,n} = \delta$ then $p_i = 0$ and $p_n^o(x^o + \Delta_n) + \delta_{i,n} = p_i(x^o + \Delta_i)$.*

Proof. The quadratic constraint for p_i is $p_n^o(x^o + \Delta_n) + \delta_{i,n} \leq p_i(x^o + \Delta_i)$. By Lemma 1, either $p_i = 0$ or the constraint is tight. If $p_n^o(x^o + \Delta_n) + \delta_{i,n} < 0$, then, since $p_i \geq 0$ and $x^o + \Delta_i \geq 0$, the constraint cannot be tight. Hence, $p_i = 0$. If $p_n^o(x^o + \Delta_n) + \delta_{i,n} > 0$, then, $p_i \neq 0$ or else with $p_i = 0$ the constraint is not satisfied. Hence the constraint is tight. The last case with $p_n^o(x^o + \Delta_n) + \delta_{i,n} = 0$ is trivial. \square

From Lemma 2, if p_n^o, x^o lies in the region between the adjacent hyperbolas given by $p_n^o(x^o + \Delta_n) + \delta_{i,n} = 0$ and $p_n^o(x^o + \Delta_n) + \delta_{j,n} = 0$ (and $0 < x^o \leq 1$ and $0 \leq p_n^o < 1$), then $\delta_{i,n} \leq 0$ and $p_i \geq 0$ and for the k^{th} quadratic constraint with $\delta_{k,n} < \delta_{i,n}$, $p_k = 0$ and for the j^{th} quadratic constraint with $\delta_{j,n} > \delta_{i,n}$, $p_j \neq 0$ and the constraint is tight.

These insights lead to Algorithm 1. After handling the case of $x = 0$ and $p_n = 1$ separately, the algorithm sorts the δ 's to get $\delta_{(1),n}, \dots, \delta_{(n-1),n}$ in ascending order. Then, it iterates over the sorted δ 's until a non-negative δ is reached, assuming the corresponding p_i 's to be zero and the other quadratic constraints to be equalities, and using the subroutine EQ_OPT to solve the induced sub-problem. For ease of exposition we assume δ 's to be distinct, but the extension to repeated δ 's is quite natural and does not require any new results. The sub-

problem for the i^{th} iteration is given by the problem $Q_{n,i}$:

$$\begin{aligned} & \max_{x, p_{(1)}, \dots, p_{(i)}, p_n} p_n \Delta_{D,n} - ax, \\ & \text{subject to} \quad p_n(x + \Delta_n) + \delta_{(i),n} \geq 0, \\ & \quad \text{if } i \geq 2 \text{ then } p_n(x + \Delta_n) + \delta_{(i-1),n} < 0, \\ & \quad \forall j \geq i. p_n(x + \Delta_n) + \delta_{(j),n} = p_{(j)}(x + \Delta_j), \\ & \quad \forall j > i. 0 < p_{(j)} \leq 1, \\ & \quad 0 \leq p_{(i)} \leq 1, \\ & \quad \sum_{k=i}^{n-1} p_{(k)} = 1 - p_n, \\ & \quad 0 \leq p_n < 1, \\ & \quad 0 < x \leq 1. \end{aligned}$$

The best (maximum) solution from all the sub-problems (including $x = 0$ and $p_n = 1$) is chosen as the final answer.

Lemma 3. *Assuming EQ_OPT produces an ϵ -additive approximate objective value, Algorithm 1 finds an ϵ -additive approximate objective of optimization problem P_n .*

The proof is omitted due to space constraints.

Algorithm 1: APX_SOLVE(ϵ, P_n)

$l \leftarrow \text{prec}(\epsilon, n, K)$, where prec is defined after Lemma 7
Sort δ 's in ascending order to get $\delta_{(1),n}, \dots, \delta_{(n-1),n}$, with corresponding variables $p_{(1)}, \dots, p_{(n-1)}$ and quadratic constraints $C_{(1)}, \dots, C_{(n-1)}$
Solve the LP problem for the two cases when $x = 0$ and $p_n = 1$ respectively. Let the solution be $S^0, p_{(1)}^0, \dots, p_{(n-1)}^0, p_n^0, x^0$ and $S^{-1}, p_{(1)}^{-1}, \dots, p_{(n-1)}^{-1}, p_n^{-1}, x^{-1}$ respectively.
for $i \leftarrow 1$ **to** $n - 1$ **do**
 if $\delta_{(i),n} \leq 0 \vee (\delta_{(i),n} > 0 \wedge \delta_{(i-1),n} < 0)$ **then**
 $p_{(j)} \leftarrow 0$ for $j < i$.
 Set constraints $C_{(i)}, \dots, C_{(n-1)}$ to be equalities.
 $S^i, p_{(1)}^i, \dots, p_{(n-1)}^i, p_n^i, x^i \leftarrow \text{EQ_OPT}(i, l)$
 else
 $S^i \leftarrow -\infty$
 end if
 $f \leftarrow \arg \max_i \{S^{-1}, S^0, S^1, \dots, S^i, \dots, S^{n-1}\}$
 $p_1^f, \dots, p_{n-1}^f \leftarrow \text{Unsort } p_{(1)}^f, \dots, p_{(n-1)}^f$
return p_1^f, \dots, p_n^f, x^f

EQ_OPT solves a two-variable problem $R_{n,i}$ instead of $Q_{n,i}$. The problem $R_{n,i}$ is defined as follows:

$$\begin{aligned} & \max_{x, p_n} p_n \Delta_{D,n} - ax, \\ & \text{subject to} \quad p_n(x + \Delta_n) + \delta_{(i),n} \geq 0, \\ & \quad \text{if } i \geq 2 \text{ then } p_n(x + \Delta_n) + \delta_{(i-1),n} < 0, \\ & \quad p_n \left(1 + \sum_{j:i \leq j \leq n-1} \frac{x + \Delta_n}{x + \Delta_{(j)}} \right) = 1 - \sum_{j:i \leq j \leq n-1} \frac{\delta_{(j),n}}{x + \Delta_{(j)}}, \\ & \quad 0 \leq p_n < 1, \\ & \quad 0 < x \leq 1. \end{aligned}$$

The following result justifies solving $R_{n,i}$ instead of $Q_{n,i}$.

Lemma 4. *$Q_{n,i}$ and $R_{n,i}$ are equivalent for all i .*

Proof. Since the objectives of both problems are identical, we prove that the feasible regions for the variables in the objective (p_n, x) are identical. Assume $p_n, x, p_{(i)}, \dots, p_{(n-1)}$ is feasible in $Q_{n,i}$. The first two constraints are the same in $Q_{n,i}$ and $R_{n,i}$. Divide each equality quadratic constraint corresponding to non-zero $p_{(j)}$ by $x + \Delta_{(j)}$. Add all such constraints to get:

$$- \sum_{j:1 \leq j \leq i} p_{(j)} + p_n \left(\sum_{j:1 \leq j \leq i} \frac{x + \Delta_n}{x + \Delta_{(j)}} \right) + \sum_{j:1 \leq j \leq i} \frac{\delta_{(j),n}}{x + \Delta_{(j)}} = 0$$

Then, since $\sum_{k:1 \leq k \leq i} p_{(k)} = 1 - p_n$ we get

$$p_n \left(1 + \sum_{j:i \leq j \leq n-1} \frac{x + \Delta_n}{x + \Delta_{(j)}} \right) = 1 - \sum_{j:i \leq j \leq n-1} \frac{\delta_{(j),n}}{x + \Delta_{(j)}}.$$

The last two constraints are the same in $Q_{n,i}$ and $R_{n,i}$.

Next, assume p_n, x is feasible in $R_{n,i}$. Choose

$$p_{(j)} = p_n \left(\frac{x + \Delta_n}{x + \Delta_{(j)}} \right) + \frac{\delta_{(j),n}}{x + \Delta_{(j)}}.$$

Since $p_n(x + \Delta_n) + \delta_{(i),n} \geq 0$, we have $p_{(i)} \geq 0$, and since $p_n(x + \Delta_n) + \delta_{(j),n} > 0$ for $j > i$ (δ 's are distinct) we have $p_{(j)} > 0$. Also,

$$\sum_{j=i}^{n-1} p_{(j)} = p_n \left(\sum_{j:i \leq j \leq n-1} \frac{x + \Delta_n}{x + \Delta_{(j)}} \right) + \sum_{j:i \leq j \leq n-1} \frac{\delta_{(j),n}}{x + \Delta_{(j)}},$$

which by the third constraint of $R_{n,i}$ is $1 - p_n$. This implies $p_{(j)} \leq 1$. Thus, $p_n, x, p_{(i)}, \dots, p_{(n-1)}$ is feasible in $Q_{n,i}$. \square

The equality constraint in $R_{n,i}$, which forms a curve K_i , allows substituting p_n with a function $F_i(x)$ of the form $f(x)/g(x)$. Then, the steps in EQ_OPT involve taking the derivative of the objective $f(x)/g(x)$ and finding those roots of the derivative that ensure that x and p_n satisfy all the constraints. The points with zero derivative are however local maxima only. To find the global maxima, other values of x of interest are where the curve K_i intersects the *closed* boundary of the region defined by the constraints. Only the closed boundaries are of interest, as maxima (rather suprema) attained on open boundaries are limit points that are not contained in the constraint region. However, such points are covered in the other optimization problems, as shown below.

The limit point on the open boundary $p_n(x + \Delta_n) + \delta_{(i-1),n} < 0$ is given by the roots of $F_i(x) + \frac{\delta_{(i-1),n}}{x + \Delta_n}$. This point is the same as the point considered on the closed boundary $p_n(x + \Delta_n) + \delta_{(i-1),n} \geq 0$ in problem $R_{n,i-1}$ given by roots of $F_{i-1}(x) + \frac{\delta_{(i-1),n}}{x + \Delta_n}$, since $F_{i-1}(x) = F_i(x)$ when $p_n(x + \Delta_n) + \delta_{(i-1),n} = 0$. Also, the other cases when $x = 0$ and $p_n = 1$ are covered by the LP solved at the beginning of Algorithm 1.

The closed boundary in $R_{n,i}$ are obtained from the constraint $p_n(x + \Delta_n) + \delta_{(i),n} \geq 0$, $0 \leq p_n$ and $x \leq 1$. The value x of the intersection of $p_n(x + \Delta_n) + \delta_{(i),n} = 0$ and

Algorithm 2: EQ_OPT(i, l)

Define $F_i(x) = \frac{1 - \sum_{j:1 \leq j \leq i-1} \frac{\delta_{(j),n}}{x + \Delta_j}}{1 + \sum_{j:1 \leq j \leq i-1} \frac{\delta_{(j),n}}{x + \Delta_j}}$

Define

$feas(x) = \begin{cases} true & (x, F_i(x)) \text{ is feasible for } R_{n,i} \\ false & \text{otherwise} \end{cases}$

Find polynomials f, g such that $\frac{f(x)}{g(x)} = F_i(x)\Delta_{D,n} - ax$

$h(x) \leftarrow g(x)f'(x) - f(x)g'(x)$
 $\{r_1, \dots, r_s\} \leftarrow \text{ROOTS}(h(x), l)$

$\{r_{s+1}, \dots, r_t\} \leftarrow \text{ROOTS}(F_i(x) + \frac{\delta_{(i),n}}{x + \Delta_n}, l)$

$\{r_{t+1}, \dots, r_u\} \leftarrow \text{ROOTS}(F_i(x), l)$

$r_{u+1} \leftarrow 1$

for $k \leftarrow 1$ **to** $u + 1$ **do**

if $feas(r_k)$ **then**

$O_k \leftarrow \frac{f(r_k)}{g(r_k)}$

else

if $feas(r_k - 2^{-l})$ **then**

$O_k \leftarrow \frac{f(r_k - 2^{-l})}{g(r_k - 2^{-l})}$; $r_k \leftarrow r_k - 2^{-l}$

else

if $feas(r_k + 2^{-l})$ **then**

$O_k \leftarrow \frac{f(r_k + 2^{-l})}{g(r_k + 2^{-l})}$; $r_k \leftarrow r_k + 2^{-l}$

else

$O_k \leftarrow -\infty$

$b \leftarrow \arg \max_k \{O_1, \dots, O_k, \dots, O_{u+1}\}$

$p_{(j)} \leftarrow 0$ for $j < i$

$p_{(j)} \leftarrow \frac{p_n(r_b + \Delta_n) + \delta_{(j),n}}{r_b + \Delta_{(j)}}$ for $j \in \{i, \dots, n-1\}$

return $O_b, p_{(1)}, \dots, p_{(n-1)}, p_n, r_b$

K_i is given by the roots of $F_i(x) + \frac{\delta_{(i),n}}{x + \Delta_n} = 0$. The value x of the intersection of $p_n = 0$ and K_i is given by roots of $F_i(x) = 0$. The value x of the intersection of $x = 1$ and K_i is simply $x = 1$. Additionally, as checked in EQ_OPT, all these intersection points must lie with the constraint regions defined in $Q_{n,i}$.

The optimal x is then the value among all the points of interest stated above that yields the maximum value for $\frac{f(x)}{g(x)}$. Algorithm 2 describes EQ_OPT, which employs a root finding subroutine ROOTS. Algorithm 2 also takes care of approximate results returned by the ROOTS. As a result of the 2^{-l} approximation in the value of x , the computed x and p_n can lie outside the constraint region when the actual x and p_n are very near the boundary of the region. Thus, we check for containment in the constraint region for points $x \pm 2^{-l}$ and accept the point if the check passes.

Remark (dummy target): As discussed in Section 2, we allow for a dummy target with all costs zero. Let this target be t_0 . For n not representing 0, there is an extra quadratic constraint given by $p_0(-x_0 - \Delta_0) + p_n(x + \Delta_n) + \delta_{0,n} \leq 0$, but, as x_0 and Δ_0 are 0 the constraint is just $p_n(x + \Delta_n) + \delta_{0,n} \leq$

0. When n represents 0, then the i^{th} quadratic constraint is $p_i(-x - \Delta_i) + \delta_{i,0} \leq 0$, and the objective is independent of p_n as $\Delta_{D,n} = 0$. We first claim that $p_0 = 0$ at any optimal solution. The proof is provided in Lemma 9 in Appendix. Thus, Lemma 1 and 2 continue to hold for $i = 1$ to $n - 1$ with the additional restriction that $p_n^o(x^o + \Delta_n) + \delta_{0,n} \leq 0$.

Thus, when n does not represent 0, Algorithm 1 runs with the the additional check $\delta_{(i),n} < \delta_{0,n}$ in the if condition inside the loop. Algorithm 2 stays the same, except the additional constraint that $p_0 = 0$. The other lemmas and the final results stay the same. When n represents 0, then x needs to be the smallest possible, and the problem can be solved analytically.

3.3 Analysis

Before analyzing the algorithm's approximation guarantee we need a few results that we state below.

Lemma 5. *The maximum bit precision of any coefficient of the polynomials given as input to ROOTS is $2n(K + 1.5) + \log(n)$.*

Proof. The maximum bit precision will be obtained in $g(x)f'(x) - f(x)g'(x)$. Consider the worst case when $i = 1$. Then, $f(x)$ is of degree n and $g(x)$ of degree $n - 1$. Therefore, the bit precision of $f(x)$ and $g(x)$ is upper bounded by $nK + \log(\binom{n}{n/2})$, where nK comes from multiplying n K -bit numbers and $\log(\binom{n}{n/2})$ arises from the maximum number of terms summed in forming any coefficient. Thus, using the fact that $\binom{n}{n/2} \leq (2e)^{n/2}$ the upper bound is approximately $n(K + 1.5)$. We conclude that the bit precision of $g(x)f'(x) - f(x)g'(x)$ is upper bounded by $2n(K + 1.5) + \log(n)$. \square

We can now use Cauchy's result on bounds on root of polynomials to obtain a lower bound for x . Cauchy's bound states that given a polynomial $a_n x^n + \dots + a_0$, any root x satisfies

$$|x| > 1 / (1 + \max\{|a_n|/|a_0|, \dots, |a_1|/|a_0|\}) .$$

Using Lemma 5 it can be concluded that any root returned by ROOTS satisfies $x > 2^{-4n(K+1.5)-2\log(n)-1}$.

Let $B = 2^{-4n(K+1.5)-2\log(n)-1}$. The following lemma (whose proof is omitted due to lack of space) bounds the additive approximation error.

Lemma 6. *Assume x is known with an additive accuracy of ϵ , and $\epsilon < B/2$. Then the error in the computed $F(x)$ is at most $\epsilon\Psi$, where $\Psi = \frac{Y + \sqrt{Y^2 + 4X}}{2}$ and*

$$X = \min \left\{ \sum_{\substack{j:i \leq j \leq n-1, \\ \delta_{j,n} < 0}} \frac{|\delta_{j,n}|}{(B + \Delta_j)^2}, \sum_{\substack{j:i \leq j \leq n-1, \\ \delta_{j,n} > 0}} \frac{2\delta_{j,n}}{(B + \Delta_j)^2} \right\}$$

$$Y = \min \left\{ \sum_{\substack{j:i \leq j \leq n-1, \\ \Delta_n - \Delta_j < 0}} \frac{|\Delta_n - \Delta_j|}{(B + \Delta_j)^2}, \sum_{\substack{j:i \leq j \leq n-1, \\ \Delta_n - \Delta_j > 0}} \frac{2(\Delta_n - \Delta_j)}{(B + \Delta_j)^2} \right\}$$

Moreover, Ψ is of order $O(n^{2(8n(K+1.5)+4\log(n)+K)})$.

We are finally ready to establish the approximation guarantee of our algorithm.

Lemma 7. *Algorithm 1 solves problem P_n with additive approximation term ϵ if*

$$l > \max\{1 + \log\left(\frac{\Delta_{D,n}\Psi + a}{\epsilon}\right), 4n(K + 1.5) + 2\log(n) + 3\}.$$

Also, as $\log\left(\frac{\Delta_{D,n}\Psi + a}{\epsilon}\right) = O(nK + \log(\frac{1}{\epsilon}))$, l is of order $O(nK + \log(\frac{1}{\epsilon}))$.

Proof. The computed value of x can be at most $2 \cdot 2^{-l}$ far from the actual value. The additional factor of 2 arises due to the boundary check in EQ_OPT. Then using Lemma 6, the maximum total additive approximation is $2 \cdot 2^{-l} \Delta_{D,n} \Psi + 2 \cdot 2^{-l} a$. For this to be less than ϵ , $l > 1 + \log\left(\frac{\Delta_{D,n}\Psi + a}{\epsilon}\right)$. The other term in the max above arises from the condition $\epsilon < B/2$ (this ϵ represents $2 \cdot 2^{-l}$) in Lemma 6. \square

Observe that the upper bound on ψ is only in terms of n and K . Thus, we can express l as a function of ϵ , n and K — $l = \text{prec}(\epsilon, n, K)$.

We still need to analyze the running time of the algorithm. First, we briefly discuss the known algorithms that we use and their corresponding running-time guarantees. Linear programming can be done in polynomial time using Karmarkar's algorithm [Karmarkar, 1984] with a time bound of $O(n^{3.5}L)$, where L is the length of all inputs.

The splitting circle scheme to find roots of a polynomial combines many varied techniques. The core of the algorithm yields linear polynomials $L_i = a_i x + b_i$ (a, b can be complex) such that the norm of the difference of the actual polynomial P and the product $\prod_i L_i$ is less than 2^{-s} , i.e., $|P - \prod_i L_i| < 2^{-s}$. The norm considered is the sum of absolute values of the coefficient. The running time of the algorithm is $O(n^3 \log n + n^2 s)$ in a pointer based Turing machine. By choosing $s = \theta(nl)$ and choosing the real part of those complex roots that have imaginary value less than 2^{-l} , it is possible to obtain approximations to the real roots of the polynomial with l bit precision in time $O(n^3 \log n + n^3 l)$. The above method may yield real values that lie near complex roots. However, such values will be eliminated in taking the maximum of the objective over all real roots, if they do not lie near a real root.

With these properties, we can state the following lemma.

Lemma 8. *The running time of Algorithm 1 with input approximation parameter ϵ and inputs of K bit precision is bounded by $O(n^5 K + n^4 \log(\frac{1}{\epsilon}))$. Using LLL yields the running time $O(\max\{n^{13} K^3, n^5 K + n^4 \log(\frac{1}{\epsilon})\})$*

Proof. The length of all inputs is $O(nK)$, where K is the bit precision of each constant. The linear programs can be computed in time $O(n^{4.5}K)$. The loop in Algorithm 1 runs less than n times and calls EQ_OPT. In EQ_OPT, the computation happens in calls to ROOTS and evaluation of the polynomial for each root found. ROOTS is called three times with a polynomial of degree less than $2n$ and coefficient bit precision less than $2n(K + 1.5) + \log(n)$. Thus, the total number of roots found is less than $6n$ and the precision of roots is l bits. By Horner's method [Horner, 1819], polynomial evaluation can be done in the following simple manner: given a polynomial $a_n x^n + \dots + a_0$ to be evaluated

at x_0 computing the following values yields the answer as $b_0, b_n = a_n, b_{n-1} = a_{n-1} + b_n x_0, \dots, b_0 = a_0 + b_1 x_0$. From Lemma 7 we get $l \geq 2n(K + 1.5) + \log(n)$, thus, b_i is approximately $(n + 1 - i)l$ bits, and each computation involves multiplying two numbers with less than $(n + 1 - i)l$ bits each. We assume a pointer-based machine, thus multiplication is linear in number of bits. Hence the total time required for polynomial evaluation is $O(n^2 l)$. The total time spent in all polynomial evaluation is $O(n^3 l)$. The splitting circle method takes time $O(n^3 \log n + n^3 l)$. Using Lemma 7 we get $O(n^4 K + n^3 \log(\frac{1}{\epsilon}))$ as the running time of EQ-OPT. Thus, the total time is $O(n^5 K + n^4 \log(\frac{1}{\epsilon}))$. \square

When using LLL, the time in ROOTS is dominated by LLL. The time for LLL is given by $O(n^{12} + n^9(\log n + nK)^3)$, which is $O(n^{12} K^3)$. Thus, the overall the time is bounded by $O(\max\{n^{13} K^3, n^4 l\})$, which using Lemma 7 is $O(\max\{n^{13} K^3, n^5 K + n^4 \log(\frac{1}{\epsilon})\})$.

4 Discussion

We have introduced a novel model of audit games that we believe to be compelling and realistic. Modulo the punishment parameter our setting reduces to the simplest model of security games. However, the security game framework is in general much more expressive. The model [Kiekintveld *et al.*, 2009] includes a defender that controls multiple security resources, where each resource can be assigned to one of several *schedules*, which are subsets of targets. For example, a security camera pointed in a specific direction monitors all targets in its field of view. As audit games are also applicable in the context of prevention, the notion of schedules is also relevant for audit games. Other extensions of security games are relevant to both prevention and detection scenarios, including an adversary that attacks multiple targets [Korzhyk *et al.*, 2011], and a defender with a budget [Bhattacharya *et al.*, 2011]. Each such extension raises difficult algorithmic questions.

Ultimately, we view our work as a first step toward a computationally feasible model of audit games. We envision a vigorous interaction between AI researchers and security and privacy researchers, which can quickly lead to deployed applications, especially given the encouraging precedent set by the deployment of security games algorithms [Tambe, 2011].

References

- [Becker, 1968] Gary S. Becker. Crime and punishment: An economic approach. *Journal of Political Economy*, 76:169, 1968.
- [Bhattacharya *et al.*, 2011] Sayan Bhattacharya, Vincent Conitzer, and Kamesh Munagala. Approximation algorithm for security games with costly resources. In *Proceedings of the 7th international conference on Internet and Network Economics*, WINE'11, pages 13–24, Berlin, Heidelberg, 2011. Springer-Verlag.
- [Blocki *et al.*, 2012] Jeremiah Blocki, Nicolas Christin, Anupam Datta, and Arunesh Sinha. Audit mechanisms for provable risk management and accountable data governance. In *GameSec*, 2012.
- [Bodik *et al.*, 2010] Peter Bodik, Moises Goldszmidt, Armando Fox, Dawn B. Woodard, and Hans Andersen. Fingerprinting the datacenter: automated classification of performance crises. In *Proceedings of the 5th European conference on Computer systems*, EuroSys '10, 2010.
- [Conitzer and Sandholm, 2006] Vincent Conitzer and Tuomas Sandholm. Computing the optimal strategy to commit to. In *ACM Conference on Electronic Commerce*, 2006.
- [Garg *et al.*, 2011] Deepak Garg, Limin Jia, and Anupam Datta. Policy auditing over incomplete logs: theory, implementation and applications. In *ACM Conference on Computer and Communications Security*, 2011.
- [Horner, 1819] W. G. Horner. A new method of solving numerical equations of all orders, by continuous approximation. *Philosophical Transactions of the Royal Society of London*, 109:308–335, 1819.
- [Karmarkar, 1984] Narendra Karmarkar. A new polynomial-time algorithm for linear programming. In *STOC*, 1984.
- [Kiekintveld *et al.*, 2009] Christopher Kiekintveld, Manish Jain, Jason Tsai, James Pita, Fernando Ordóñez, and Milind Tambe. Computing optimal randomized resource allocations for massive security games. In *Proceedings of The 8th International Conference on Autonomous Agents and Multiagent Systems - Volume 1*, AAMAS '09, pages 689–696. International Foundation for Autonomous Agents and Multiagent Systems, 2009.
- [Korzhyk *et al.*, 2010] Dmytro Korzhyk, Vincent Conitzer, and Ronald Parr. Complexity of computing optimal stackelberg strategies in security resource allocation games. In *AAAI*, 2010.
- [Korzhyk *et al.*, 2011] Dmytro Korzhyk, Vincent Conitzer, and Ronald Parr. Security games with multiple attacker resources. In *IJCAI*, pages 273–279, 2011.
- [Lenstra *et al.*, 1982] A.K. Lenstra, H.W. jun. Lenstra, and László Lovász. Factoring polynomials with rational coefficients. *Math. Ann.*, 261:515–534, 1982.
- [Neumaier, 2004] Arnold Neumaier. Complete search in continuous global optimization and constraint satisfaction. *Acta Numerica*, 13:271–369, 2004.
- [Pita *et al.*, 2008] James Pita, Manish Jain, Fernando Ordóñez, Christopher Portway, Milind Tambe, Craig Western, Praveen Paruchuri, and Sarit Kraus. Armor security for los angeles international airport. In *AAAI*, pages 1884–1885, 2008.
- [Pita *et al.*, 2011] James Pita, Milind Tambe, Christopher Kiekintveld, Shane Cullen, and Erin Steigerwald. Guards - innovative application of game theory for national airport security. In *IJCAI*, pages 2710–2715, 2011.
- [Schönhage, 1982] A. Schönhage. The fundamental theorem of algebra in terms of computational complexity. Technical report, University of Tübingen, 1982.
- [Tambe, 2011] Milind Tambe. *Security and Game Theory: Algorithms, Deployed Systems, Lessons Learned*. Cambridge University Press, 2011.

[Vaughan *et al.*, 2008] Jeffrey A. Vaughan, Limin Jia, Karl Mazurak, and Steve Zdancewic. Evidence-based audit. In *CSF*, pages 177–191, 2008.

[von Stackelberg, 1934] H. von Stackelberg. *Marktform und Gleichgewicht*. - Wien & Berlin: Springer 1934. VI, 138 S. 8. J. Springer, 1934.

[Zhao and Johnson, 2008] Xia Zhao and Eric Johnson. Information governance: Flexibility and control through escalation and incentives. In *WEIS*, 2008.

[Zheng *et al.*, 2006] Alice X. Zheng, Ben Liblit, and Mayur Naik. Statistical debugging: simultaneous identification of multiple bugs. In *In ICML 06: Proceedings of the 23rd international conference on Machine learning*, pages 1105–1112, 2006.

A Missing proofs

Proof of Lemma 1. We prove the contrapositive. Assume there exists a i such that $p_i \neq 0$ and the i^{th} quadratic constraint is not tight. Thus, there exists an $\epsilon > 0$ such that

$$p_i(-x^o - \Delta_i) + p_n^o(x^o + \Delta_n) + \delta_{i,n} + \epsilon = 0.$$

We show that it is possible to increase to p_n^o by a small amount such that all constraints are satisfied, which leads to a higher objective value, proving that p_n^o, x^o is not optimal. Remember that all Δ 's are ≥ 0 , and $x > 0$.

We do two cases: (1) assume $\forall l \neq n. p_n^o(x^o + \Delta_n) + \delta_{l,n} \neq 0$. Then, first, note that p_n^o can be increased by ϵ_i or less and p_i can be decreased by ϵ'_i to still satisfy the constraint, as long as

$$\epsilon'_i(x^o + \Delta_i) + \epsilon_i(x^o + \Delta_n) \leq \epsilon.$$

It is always possible to choose such $\epsilon_i > 0, \epsilon'_i > 0$. Second, note that for those j 's for which $p_j = 0$ we get $p_n^o(x^o + \Delta_n) + \delta_{j,n} \leq 0$, and by assumption $p_n^o(x^o + \Delta_n) + \delta_{j,n} \neq 0$, thus, $p_n^o(x^o + \Delta_n) + \delta_{j,n} < 0$. Let ϵ_j be such that $(p_n^o + \epsilon_j)(x^o + \Delta_n) + \delta_{j,n} = 0$, i.e., p_n^o can be increased by ϵ_j or less and the j^{th} constraint will still be satisfied. Third, for those k 's for which $p_k \neq 0$, p_n^o can be increased by ϵ_k or less, which must be accompanied with $\epsilon'_k = \frac{x^o + \Delta_n}{x^o + \Delta_i} \epsilon_k$ increase in p_k in order to satisfy the k^{th} quadratic constraint.

Choose feasible ϵ'_k 's (which fixes the choice of ϵ_k also) such that $\epsilon'_i - \sum_k \epsilon'_k > 0$. Then choose an increase in p_i : $\epsilon''_i < \epsilon'_i$ such that

$$\epsilon_n = \epsilon''_i - \sum_k \epsilon'_k > 0 \text{ and } \epsilon_n < \min\{\epsilon_i, \min_{p_j=0} \epsilon_j, \min_{p_k \neq 0} \epsilon_k\}$$

Increase p_n^o by ϵ_n , p_k 's by ϵ'_k and decrease p_i by ϵ''_i so that the constraint $\sum_i p_i = 1$ is still satisfied. Also, observe that choosing an increase in p_n^o that is less than any ϵ_k , any ϵ_j , ϵ_i satisfies the quadratic constraints corresponding to p_k 's, p_j 's and p_i respectively. Then, as $\epsilon_n > 0$ we have shown that p_n^o cannot be optimal.

Next, for the case (2) if $p_n^o(x^o + \Delta_n) + \delta_{l,n} = 0$ for some l then $p_l = 0$, $\delta_{l,n} < 0$ and the objective becomes

$$p_n \Delta_n - \frac{\delta_{l,n}}{p_n} - \Delta_n.$$

Thus, increasing p_n increases the objective. Note that choosing a lower than x^o feasible value for x , results in an higher than p_n^o value for p_n . Also, the k^{th} constraint can be written as $p_k(-x - \Delta_k) + \delta_{k,n} - \delta_{l,n} \leq 0$. We show that it is possible to choose a feasible x lower than x^o . If for some j , $p_j = 0$, then x can be decreased without violating the corresponding constraint. Let p_t 's be the probabilities that are non-zero and let the number of such p_t 's be T . By assumption there is an $i \neq l$ such that $p_i > 0$ and

$$p_i(-x^o - \Delta_i) + \delta_{i,n} - \delta_{l,n} + \epsilon = 0.$$

For i , it is possible to decrease p_i by ϵ_i such that $\epsilon_i(x^o + \Delta_i) \leq \epsilon/2$, hence the constraint remains satisfied and is still non-tight.

Increase each p_t by ϵ_i/T so that the constraint $\sum_i p_i = 1$ is satisfied. Increasing p_t makes the t^{th} constraint becomes non-tight for sure. Then, all constraints with probabilities greater than 0 are non-tight. For each such constraint it is possible to decrease x (note $x^o > 0$) without violating the constraint. Thus, we obtain a lower feasible x than x^o , hence a higher p_n than p_n^o . Thus, p_n^o, x^o is not optimal. \square

Proof of Lemma 3. If $p_n^o(x^o + \Delta_n) + \delta_{i,n} \geq 0$ and $p_n^o(x^o + \Delta_n) + \delta_{j,n} < 0$, where $\delta_{j,n} < \delta_{i,n}$ and $\nexists k. \delta_{j,n} < \delta_{k,n} < \delta_{i,n}$, then the exact solution of the i^{th} subproblem will be p_n^o, x^o . Now, since $0 < x \leq 1$ and $0 \leq p_n < 1$, there is one i for which $p_n^o(x^o + \Delta_n) + \delta_{i,n} \geq 0$ and $p_n^o(x^o + \Delta_n) + \delta_{j,n} < 0$, and thus the solution of this sub-problem will return the maximum value. The solution of other sub-problems will return a lower value as the objective is same in all sub-problems. Hence, maximum of the maximum in each iteration is the global maximum. The approximation case is then an easy extension. \square

Proof of Lemma 6.

$$\frac{\Delta_n - \Delta_j}{x - \epsilon + \Delta_j} < \frac{\Delta_n - \Delta_j}{x + \Delta_j} \text{ if } \Delta_n - \Delta_j < 0$$

and using the fact that $\frac{1}{1-\epsilon} < 1 + 2\epsilon$ for $\epsilon < 1/2$,

$$\frac{\Delta_n - \Delta_j}{x - \epsilon + \Delta_j} < \frac{\Delta_n - \Delta_j}{x + \Delta_j} + 2\epsilon \frac{\Delta_n - \Delta_j}{(x + \Delta_j)^2}$$

if $\Delta_n - \Delta_j > 0$ and $\frac{\epsilon}{x + \Delta_j} < 1/2$. The latter condition is true as $\epsilon < B/2$. Thus,

$$\sum_{j:i \leq j \leq n-1} \frac{\Delta_n - \Delta_j}{x - \epsilon + \Delta_j} < \sum_{j:i \leq j \leq n-1} \frac{\Delta_n - \Delta_j}{x + \Delta_j} + \sum_{j:i \leq j \leq n-1, \Delta_n - \Delta_j > 0} \frac{2\epsilon(\Delta_n - \Delta_j)}{(x + \Delta_j)^2}$$

$$\frac{\Delta_n - \Delta_j}{x + \epsilon + \Delta_j} < \frac{\Delta_n - \Delta_j}{x + \Delta_j} \text{ if } \Delta_n - \Delta_j > 0$$

and using the fact that $\frac{1}{1+\epsilon} > 1 - \epsilon$,

$$\frac{\Delta_n - \Delta_j}{x + \epsilon + \Delta_j} < \frac{\Delta_n - \Delta_j}{x + \Delta_j} - \epsilon \frac{\Delta_n - \Delta_j}{(x + \Delta_j)^2}$$

if $\Delta_n - \Delta_j < 0$. Thus,

$$\sum_{j:i \leq j \leq n-1} \frac{\Delta_n - \Delta_j}{x + \epsilon + \Delta_j} < \sum_{j:i \leq j \leq n-1} \frac{\Delta_n - \Delta_j}{x + \Delta_j} + \sum_{j:i \leq j \leq n-1, \Delta_n - \Delta_j < 0} \frac{\epsilon |\Delta_n - \Delta_j|}{(x + \Delta_j)^2}$$

Thus, using the fact that $x > B$ we have

$$\begin{aligned} \sum_{j:i \leq j \leq n-1} \frac{\Delta_n - \Delta_j}{x_\epsilon + \Delta_j} &< \sum_{j:i \leq j \leq n-1} \frac{\Delta_n - \Delta_j}{x + \Delta_j} + \epsilon \min \left\{ \sum_{j:i \leq j \leq n-1, \Delta_n - \Delta_j < 0} \frac{|\Delta_n - \Delta_j|}{(B + \Delta_j)^2}, \sum_{j:i \leq j \leq n-1, \Delta_n - \Delta_j > 0} \frac{2(\Delta_n - \Delta_j)}{(B + \Delta_j)^2} \right\} \\ &= \sum_{j:i \leq j \leq n-1} \frac{\Delta_n - \Delta_j}{x + \Delta_j} + \epsilon Y \end{aligned}$$

Very similar to the above proof we also get

$$\begin{aligned} - \sum_{j:i \leq j \leq n-1} \frac{\delta_{j,n}}{x_\epsilon + \Delta_j} &> - \sum_{j:i \leq j \leq n-1} \frac{\delta_{j,n}}{x + \Delta_j} - \epsilon \min \left\{ \sum_{j:i \leq j \leq n-1, \delta_{j,n} < 0} \frac{|\delta_{j,n}|}{(B + \Delta_j)^2}, \sum_{j:i \leq j \leq n-1, \delta_{j,n} > 0} \frac{2\delta_{j,n}}{(B + \Delta_j)^2} \right\} \\ &= - \sum_{j:i \leq j \leq n-1} \frac{\delta_{j,n}}{x + \Delta_j} - \epsilon X \end{aligned}$$

Then given

$$p_n = \frac{1 - \sum_{j:i \leq j \leq n-1} \frac{\delta_{(j),n}}{x + \Delta_{(j)}}}{\left(1 + \sum_{j:i \leq j \leq n-1} \frac{x + \Delta_n}{x + \Delta_{(j)}}\right)} = \frac{A}{B}$$

Using the inequalities above

$$F(x_\epsilon) = \frac{1 - \sum_{j:i \leq j \leq n-1} \frac{\delta_{(j),n}}{x_\epsilon + \Delta_{(j)}}}{\left(1 + \sum_{j:i \leq j \leq n-1} \frac{x_\epsilon + \Delta_n}{x_\epsilon + \Delta_{(j)}}\right)} > \frac{A - \epsilon C}{B + \epsilon D}$$

If $p_n \leq \psi$, then since $F(x_\epsilon) \geq 0$, we have $F(x) \geq p_n - \psi$.

If $p_n \geq \psi$, then since $B > 1$, $A > \psi$ we have

$$\frac{A - \epsilon X}{B + \epsilon Y} > \frac{A}{B} \left(1 - \epsilon \left(\frac{X}{A} + \frac{Y}{B}\right)\right) > p_n \left(1 - \epsilon \left(\frac{X}{\psi} + Y\right)\right)$$

And, as $p_n < 1$ we have $F(x_\epsilon) > p_n - \epsilon \left(\frac{X}{\psi} + Y\right)$. Thus,

$F(x_\epsilon) > p_n - \epsilon \min\{\psi, \frac{X}{\psi} + Y\}$. The minimum is less than the positive root of $\psi^2 - Y\psi - X = 0$, that is $\frac{Y + \sqrt{Y^2 + 4X}}{2}$. \square

B Dummy Target

Lemma 9. *If a dummy target t_0 is present for the optimization problem described in Section 2, then $p_0 = 0$ at optimal point p_n^o, x^o , where $x^o > 0$.*

Proof. We prove a contradiction. Let $p_0 > 0$ at optimal point. If the problem under consideration is when n represents 0, the objective is not dependent on p_n and thus, then we want to choose x as small as possible to get the maximum objective value. The quadratic inequalities are $p_i(-x^o - \Delta_i) + \delta_{i,0} \leq 0$. Subtracting ϵ from p_0 and adding ϵ/n to all the other p_i , satisfies the $\sum_i p_i = 1$ constraint. But, adding ϵ/n to all the other p_i , allows reducing x^o by a small amount and yet, satisfying each quadratic constraint. But, then we obtain a higher objective value, hence x^o is not optimal.

Next, if the problem under consideration is when n does not represent 0, then an extra constraint is $p_n^o(x^o + \Delta_n) + \delta_{n,0} \leq 0$. Subtracting ϵ from p_0 and adding $\epsilon/(n-1)$ to all the other p_i (except p_n^o), satisfies the $\sum_i p_i = 1$ constraint. Also, each constraint $p_i(-x^o - \Delta_i) + p_n^o(x^o + \Delta_n) + \delta_{i,n} \leq 0$ becomes non-tight (may have been already non-tight) as a result of increasing p_i . Thus, now x^o can be decreased (note $x^o > 0$). Hence, the objective increases, thus p_n^o, x^o is not optimal. \square