An Adversarial Risk Analysis Framework for the Software Release Problem

Fabrizio Ruggeri

Istituto di Matematica Applicata e Tecnologie Informatiche
Consiglio Nazionale delle Ricerche
Via Alfonso Corti 12, I-20133, Milano, Italy, European Union

fabrizio@mi.imati.cnr.it

www.mi.imati.cnr.it/fabrizio

ADVERSARIAL RISK ANALYSIS

- (Bayesian) Decision Analysis supports a Decision Maker (DM) in making decisions under uncertainty:
  - Set of alternatives (actions) $a \in \mathcal{A}$
  - Unknown parameter θ depending on *state of nature*
  - Consequence $c(a, \theta)$ of action a when θ occurs
  - Utility function $u(c(a, \theta))$
  - Posterior distribution $\pi(\theta|x)$ on parameter θ, after observing x
  - Optimal action satisfies the Maximum (Subjective) Expected Utility Principle:

$$a^* = \arg\max_{a \in \mathcal{A}} \int u(c(a, \theta))\pi(\theta|x)d\theta$$

- Just one agent playing against *Nature*

ADVERSARIAL RISK ANALYSIS

- More intelligent agents involved (two in the next, Defender [she] and Attacker [he]), all expected utility maximisers
  - Competitors in auction bidding for projects
  - Terrorists and intelligence agencies
  - Competing interest groups (e.g. jobs vs. health in risky productions)
- Decisions by an agent affecting the others
- Knowledge of others' behaviour, e.g. in many Game Theory approaches
  $\Rightarrow$ unrealistic
- Attacker's probabilities and utilities unknown to Defender
- Our goal: help Defender to assess probabilities and utilities of the Attacker, and his consequent optimal decisions, to mitigate effects of his actions

ADVERSARIAL SOFTWARE TESTING

- Software subject to (possibly expensive and dangerous) failures in programming or system design
- $\Rightarrow$ software must undergo rigorous testing, both during development and operation, to verify its reliability
- Optimal policies for software release $\Rightarrow$ important issue in software engineering
- Challenges due to several, often uncertain, complicating factors
- Endogenous factors
  - number of bugs in the software
  - skill in detecting bugs
- Exogenous factors
  - release decisions made by competitors
  - eventual purchasing decision by software buyers

ADVERSARIAL SOFTWARE TESTING

- Monetary aspects
  - costs related to time on test
  - costs related to bugs discovering and their fixing during testing
  - costs related to bugs discovering and their fixing after the release
  - monetary gain for the software sale
- Reputational aspects
- Early software release $\Rightarrow$ larger commercial advantage over competitors
- Less intensely tested software $\Rightarrow$ possible lower quality $\Rightarrow$ potential advantage to competitors

ADVERSARIAL SOFTWARE TESTING

- Singpurwalla and Wilson (2012): Review of software reliability and testing
- Anand, Singh, Das (2015): evaluation of two types (simple and serious) failures in successive versions of a software, during testing and operational phases
- Wilson and O'Riordain (2018): optimal release policy of new versions of Mozilla Firefox based on bug detection data
- Saraf and Iqbal (2019): software reliability model based on NHPP, performing fault detection, observation and correction in two stages and multiple versions
- Mishra, Kapur, Srivastava (2018): reliability growth of software over multiple versions
- Kenett, Ruggeri, Faltin (2018): thorough review of analytic methods in systems and software testing
- Ay, Landon, Ruggeri, Soyer (2022): software testing with possible introduction of bugs

ADVERSARIAL SOFTWARE TESTING

- Ruggeri, Soyer (2018): overview of games and decision models for software testing
- Forman, Singpurwalla (1977, 1979) and Okumoto, Goel (1979): introduction of stopping time models to support software release decisions
- Dalal, Mallows (1988): pioneer work on decision theoretic models for release
- Morali, Soyer (2003): sequential Bayesian decision theoretic setup for developing optimal stopping policies for software testing
- Zeephongsekul, Chiera (1995): first game theoretic approach looking for optimal release policies through Nash equilibrium
  - Dohi, Teraoka, Osaki (2000): different approach since previous solution restricted to particular case and computationally intractable
  - Saito, Dohi (2022): uncovered faults in the earlier two papers showing the existence of Nash equilibrium under some parametric conditions

ADVERSARIAL SOFTWARE TESTING

- Overview of Zeephongsekul and Chiera (1995)
- First work to consider also actions and costs of a competitor
- Two competitors ($i = 1, 2$) produce software performing the same set of tasks and with life cycle length non exceeding T
- Competitor i, $i = 1, 2$, decides to release the software at any time t in $[0, T]$ and sells the product with probability $A_i(t)$ to the only buyer (who buys from one competitor at most)
- $A_i(t)$, $i = 1, 2$, continuously differentiable, concave and s.t. $A_i(0) = A_i(T) = 0$ with a unique maximum at time η_i
  - Choice of $A_i(t)$ not only for mathematical convenience but also justified by actual behaviour
  - Success probability expected to be close to 0 both at the beginning and the end of the life cycle $[0, T]$, because of initial poor reliability and final obsolescence, respectively

ADVERSARIAL SOFTWARE TESTING

- Introduction of expected cost function $c_i(t)$ incurred by player i in releasing the software at time t

- $c_i(t) = c_{1i}t + c_{2i}m_i(t) + c_{3i}\left(m_i(T) - m_i(t)\right)$
  - c_{1i} cost of testing per unit time
  - c_{2i} cost of removing a fault during testing
  - c_{3i} cost of removing a fault during operation, with $c_{3i} > c_{2i}$ since fixing an error is more expensive after release than before it
  - $m_i(t)$ expected number of faults detected up to time t
  - increasing, concave and differentiable $m_i(t)$, with $m(0) = 0$

- $\Rightarrow c_i(t)$ convex function with minimum at γ_i s.t. $\Rightarrow m_i^{'}(\gamma_i) = \dfrac{c_{i1}}{(c_{3i} - c_{2i})}$

- T is sufficiently large so that $\gamma_i < T$

ADVERSARIAL SOFTWARE TESTING

- $p_i > 0$: selling price of the software produced by player i
- If player 1 releases software at time x and player 2 at time $y \Rightarrow M_i(x, y)$ is the expected unit profit to player i, with

$$M_1(x,y) = \begin{cases} p_1 A_1(x) - c_1(x) & 0 \leq x < y \leq T \\ p_1(1 - A_2(y))A_1(x) - c_1(x) & 0 \leq y < x \leq T \end{cases}$$

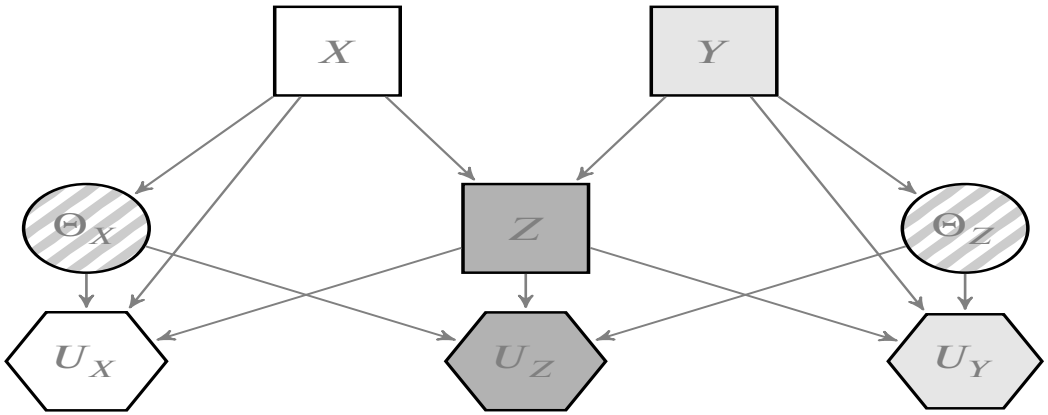
- $M_2(x, y)$ can be described similarly and $M_1(x, y) \neq M_2(x, y)$ in general
- $\Rightarrow$ optimal release policies among Nash equilibrium points in this non-zero sum game (with concerns about the results as mentioned earlier)
- The paper, and all game theoretic work in the field, entails common knowledge assumptions, debatable in competitive business settings as in software development
- $\Rightarrow$ Adversarial Risk Analysis $\Rightarrow$ Adversarial Software Testing

ADVERSARIAL SOFTWARE TESTING

- Guevara, Pierce, Rios Insua, Ruggeri, Soyer (submitted)
- Support for producer X against competitor Y, trying both to sell software to buyer Z (purchasing from one producer at most)
- X can release the software at any time $x \in [0, T]$
- In absence of competitors, X would succeed in selling the product at the price p_X with probability $A_X(x)$, with $A_X(0) = A_X(T) = 0$ (less restrictive than before)
- Y releases at time $y \in [0, T]$ independently, succeeding to sell at fixed price p_Y with probability $A_Y(y)$, with similar properties as A_X
- Consider a stochastic number $N_X(t)$ of faults found until time t, instead of the expected number $m_X(t) = E[N_X(t)]$
- $N_X(t)$ NHPP with intensity $\lambda_X(t)$ and mean value function $m_X(t) = \int_0^t \lambda_X(u)du$
- Similar definitions apply to Y

ADVERSARIAL SOFTWARE TESTING

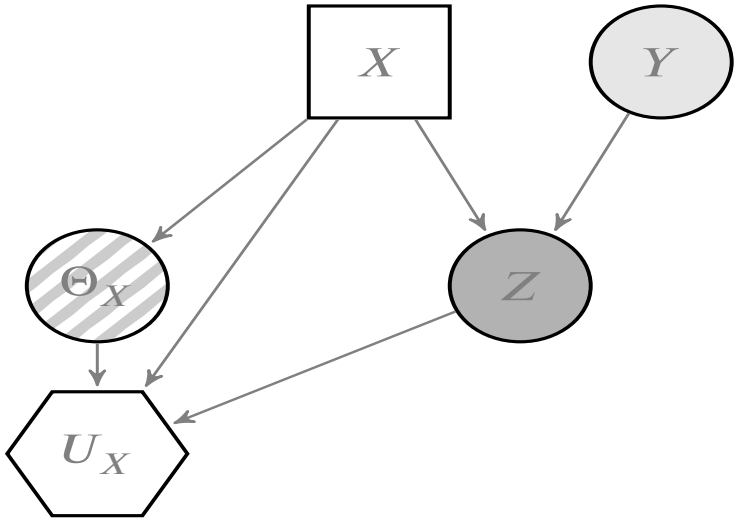
Tri-agent influence diagram representing the basic problem



- Global perspective
- Different colours for different agents
- Square nodes: Decisions by producers (X and Y) and buyer (Z)
- Circle nodes: Uncertain features of X (Θ_X) and Y (Θ_Y), like number of bugs
- Hexagonal nodes: Utilities U_X, U_Y, U_Z for X, Y, Z

ADVERSARIAL SOFTWARE TESTING

Tri-agent influence diagram representing the basic problem



- Perspective from producer X, the one we are taking in the work
- Y's decision now as a circle since it is uncertain for X

ADVERSARIAL SOFTWARE TESTING

- (X, Y, Z) maximise their expected utilities
- X is a level-2 agent who thinks Y is a strategic adversary, while Y is a level-1 thinker who assumes that X is a level-0, that is a nonstrategic adversary
- Subscripts indicate agent and capital letters random utilities and probabilities
- Goal: find $x^* = \arg\max_{x \in \mathcal{X}} \int\int\int u_X(x, \theta_X, z) p_X(\theta_X|x) p_X(z|x, y) p_X(y) d\theta_X dz dy$, where $\mathcal{X}$ represents the set of feasible decisions available to X
- Assume $p_X(\theta_X|x), p_X(z|x, y), p_X(y)$ are available, the continuity of u_X in x for almost all θ_X and z leads to the continuity of the expected utility which, with the compactness of the feasible space $\mathcal{X}$, guarantees the existence of the optimal x^*
- **Proposition:** If u_X is continuous in x for almost all θ_X and z and $\mathcal{X}$ is compact, then x^* is well-defined
- x^* found through Monte Carlo simulation, based on standard elements in Decision Analysis, except for $p_X(y)$ and $p_X(z|x, y)$ which entail strategic thinking

ADVERSARIAL SOFTWARE TESTING

- Regarding $p_X(y)$, observe that Y faces a symmetric problem w.r.t. X:
  $y^* = \arg\max_{y \in \mathcal{Y}} \iiint u_Y(y, \theta_Y, z) p_Y(\theta_Y|y) p_Y(z|x, y) p_Y(x) d\theta_Y dz dx$

- Unavailable utilities and probabilities for agent $Y \Rightarrow$ model uncertainty through random probabilities P_Y and utilities U_Y

- Random optimal $Y^* = \arg\max_{y \in \mathcal{Y}} \iiint U_Y(y, \theta_Y, z) P_Y(\theta_Y|y) P_Y(z|x, y) P_Y(x) d\theta_Y dz dx$,
  with $\mathcal{Y}$ as Y's feasible decision space

- We then make $Pr_X(Y \leq y) = Pr(Y^* \leq y)$

- **Proposition:** If the utilities in the support of U_Y are a.s. continuous in y, θ_Y and z, and $\mathcal{Y}$ is compact, then $Pr_X(Y \leq y)$ is well-defined

- Implementation through Monte Carlo, by sampling from the random utilities and probabilities and solving the corresponding optimization problem for each sample of utilities and probabilities, which in turn provides a sample from Y^*

ADVERSARIAL SOFTWARE TESTING

- Regarding $p_X(z|x,y)$, the consumer would solve a pairwise comparison problem $z^*(x,y) = \arg\max_{x,y} \left(\int u_Z(x,\theta_X) p_Z(\theta_X|x) d\theta_X, \int u_Z(y,\theta_Y) p_Z(\theta_Y|y) d\theta_Y \right)$
- Unavailable utilities and probabilities for agent $Z \Rightarrow$ model uncertainty through random probabilities P_Z and utilities U_Z
- Random optimal $Z^*(x,y) = \arg\max_{x,y} \left(\int U_Z(x,\theta_X) P_Z(\theta_X|x) d\theta_X, \int U_Z(y,\theta_Y) P_Z(\theta_Y|y) \right)$
- **Proposition:** $p_X(z|x,y)$ exists if the utility functions in the support of U_Z are integrable
- In general, the above process has to be implemented through Monte Carlo, by sampling from the random utilities and probabilities and solving the corresponding pairwise comparison problem
- From samples from $Z^*(x,y)$ we build the required $p_X(z|x,y) = Pr(Z^*(x,y) = z)$

ADVERSARIAL SOFTWARE TESTING

- $c_X(t) = c_{1X}t + c_{2X}N_X(t) + c_{3X}\,[N_X(T) - N_X(t)]$
  - c_{1i} cost of testing per unit time
  - c_{2i} cost of removing a fault during testing
  - $c_{3i} > c_{2i}$ cost of removing a fault during operation
- We assume that no new bugs are introduced during the debugging phase
- We assume that fault arrivals can be described by the same process during debugging and operational phase after the software has been released
- There are other assumptions leading to further developments, e.g., price fixed in advance, only two producers, only one buyer, fixed purchase probability

ADVERSARIAL SOFTWARE TESTING

- X and Y release their software at times x and y, respectively ($x \neq y$ a.s.)
- X stops testing if the buyer does not purchase its software, either because it rejects the product or because it has already bought it from Y
- $g_X(x, y)$ (random) gain of producer X given such release times
- Start with $x < y$ and rename g_X as g_{X1}
- $\Rightarrow g_{X1}(x, y) = A_X(x)\,[p_X - c_X(x)] - [1 - A_X(x)]\,[c_{1X}\,x + c_{2X} N_X(x)]$
- First term: expected gain if Z buys X's software given by purchase probability at time x times the difference between selling price and costs due to debugging until x and fault removals after the release up to time T
- Second term: expected loss due to refusal by Z and costs incurred until release time
- Note that $g_{X1}(x, y)$ does not depend on y

ADVERSARIAL SOFTWARE TESTING

- Similarly, Y's gain, for $y < x$, not dependent on x:
- $g_{Y1}(x, y) = A_Y(y)\,[p_Y - c_Y(y)] - [1 - A_Y(y)]\,[c_{1Y}\,y + c_{2Y} N_Y(y)]$
- When $x > y$, the X's gain is renamed as g_{X2}

$$\begin{aligned} g_{X2}(x, y) \quad = \quad & -A_Y(y)\,[c_{1X} y + c_{2X} N_X(y)] + [1 - A_Y(y)]\,\{A_X(x)\,[p_X - c_X(x)] - \\ & - [1 - A_X(x)]\,[c_{1X} x + c_{2X} N_X(x)]\} \end{aligned}$$

- First term: Z buys Y's software and X stops debugging its own
- Second and third term: like earlier, but after Z's refusal of buying Y's software
- Similar result for Y when $y > x$

ADVERSARIAL SOFTWARE TESTING

- Assuming risk neutrality $\Rightarrow$ expected gain $h_X(x,y)$ replacing $N_X(t)$ with its expectation, like for $x < y$

$$\begin{aligned} h_{X1}(x,y) &= A_X(x)\,[p_X - (c_{1X}x + c_{2X}m_X(x) + c_{3X}\,[m_X(T) - m_X(x)])] \\ &\quad - [1 - A_X(x)]\,[c_{1X}x + c_{2X}m_X(x)] \end{aligned}$$

- As an anticipation of what is next, X can also consider $A_Y(y)$ as random and compute its expectation when $x > y$

$$h_{X2}(x,y) = -E(A_Y(y))[c_{1X}y + c_{2X}m_X(y)] + (1 - E(A_Y(y)))\times$$

$$\times[[A_X(x)[p_X - (c_{1X}x + c_{2X}m_X(x) + c_{3X}[m_X(T) - m_X(x)]] - [1 - A_X(x)]\times$$

$$\times[c_{1X}x + c_{2X}m_X(x)]]]$$

- Similar results apply to Y

ADVERSARIAL SOFTWARE TESTING

- $\pi_Y^X(y)$: density modelling X's beliefs about Y's release decision being time y
- Expected gain associated with release decision x
  $M_X(x) = \int h_X(x,y)\pi_Y^X(y)dy = \int_0^x h_{X2}(x,y)\pi_Y^X(y)dy + \int_x^T h_{X1}(x,y)\pi_Y^X(y)dy$
- Optimal release time for X: $x^* = \arg\max_{0\leq x\leq T} M_X(x)$
- Above arguments slightly modified in absence of risk neutrality, i.e., when considering a utility function u_X

  $g_{X1}(x,y) = A_X(x)\times u_X(p_X - c_X(x)) + [1 - A_X(x)]\times u_X(-(c_{1X}(x) + c_{2X}N_X(x)))$

  $$\begin{aligned} g_{X2}(x,y) \quad = \quad & A_Y(y) \times u_X(-[c_{1X}y + c_{2X}N_X(y)]) + [1 - A_Y(y)] \times \\ & \times \{A_X(x)u_X([p_X - c_X(x)]) + [1 - A_X(x)]\, u_X(-[c_{1X}x + c_{2X}N_X(x)])\} \end{aligned}$$

ADVERSARIAL SOFTWARE TESTING

- All the elements introduced above are standard in the decision analysis and software reliability literature and practice, except for those entailing strategic thinking:
  - $A_Y(y)$ (purchase probability of Y's software)
  - $\pi_Y^X(y)$ (X's beliefs about Y releasing its product at time y)
- Need for procedures to facilitate their assessment, starting with $\pi_Y^X(y)$
- Look at Y's perspective on product release
- Remember that Y has a cost function $c_Y(t)$ and a purchase probability function $A_Y(t)$ for a fixed price p_Y, with similar properties and definitions than those of X
- Presenting now an approach to obtain an estimate $\widehat{\pi}_Y^X(t)$ of $\pi_Y^X(t)$ reflecting upon the optimisation problem faced by Y

ADVERSARIAL SOFTWARE TESTING

- Suppose X has complete knowledge about Y's behaviour, i.e., c_{1Y}, c_{2Y}, c_{3Y}, p_Y, $\lambda_Y(t)$, $A_Y(t)$ and $\pi_X^Y(t)$ (which models Y's beliefs about X's release time)

- $\Rightarrow X$ could guess Y's actual optimal release time y^*, using the previous computations by interchanging X and Y

- But we have uncertainty about Y's elements so that we
  - model such uncertainty through probability measures $\Pi_X^Y(t)$, C_{1Y}, C_{2Y}, C_{3Y}, P_Y, $\mathcal{A}_Y$ and $\mathcal{N}_Y(t)$ over the space of suitable densities $\pi_X^Y(t)$, constants c_{1Y}, c_{2Y}, c_{3Y}, p_Y, functions A_Y and processes $N_Y(t)$, respectively
  - make a sufficiently large number of draws from these components, compute the corresponding optimal release time y^* for each draw, and estimate an empirical distribution over y^*, which will be considered as the estimate $\widehat{\pi}_Y^X(y)$
  - $\Rightarrow X$ will be able to compute its optimal release time x^*

ADVERSARIAL SOFTWARE TESTING

- The random ingredients could be specified gathering all information available and modelling with standard expert judgement

- Here we consider several heuristics based on adding some uncertainty to the judgements concerning X

- Y's random beliefs about X's decision $\Pi_X^Y(t)$
  - Transform the time interval $[0, T]$ into the unit interval via the transformation $t \to t/T$, $0 \leq t \leq T$
  - Consider suitable densities $\pi_X^Y(t)$ in the space of all beta densities over $[0, 1]$ or a proper subset, if X feels capable of adding some constraints about their parameters, e.g. by fixing lower and/or upper bounds over mean and/or variance of the beta distributions
  - Randomly generate densities from such class, e.g., drawing a uniform distribution over both parameters of the beta distribution or its mean-variance pair

ADVERSARIAL SOFTWARE TESTING

- Y's random beliefs about X's decision $\Pi_X^Y(t)$
  - Use distortion function as in Arias-Nicolas, Ruggeri and Suárez-Llorens (2016)
  - Start from an absolutely continuous (for simplicity) pdf $\pi_X(t)$ and its cdf $\Pi_X(t)$, expressing X's opinion on Y's release time and build a random space of cdf's $\pi_X^Y(t)$ around it
  - Consider distortion functions $h(t)$, i.e. non-decreasing functions such that $h : [0, 1] \to [0, 1]$, $h(0) = 0, h(1) = 1$
  - Apply $h(\cdot)$ to $\Pi_X(t)$ and obtain random pdf's $\Pi_{hX}^Y(t) = h(\Pi_X(t))$ and cdf's $\pi_{hX}^Y(t) = h'(\Pi_X(t))\pi_X(t)$
  - Consider a band around $\Pi_X(t)$ taking one convex and one concave distortion function to get, respectively, its lower and upper bounds
  - A useful choice for a distortion function is $h(t) = t^\alpha$, which is convex for $0 < \alpha < 1$ and concave for $\alpha > 1$
  - Randomness is induced by, say, considering that α follows a uniform distribution on a certain interval

ADVERSARIAL SOFTWARE TESTING

- Uncertainty about Y's costs
  - Model X's uncertainty about c_{1Y}, c_{2Y} and c_{3Y} considering independent (Gaussian) distributions centered around the corresponding values c_{1X}, c_{2X}, c_{3X}
  - Alternatively, if X can provide upper and lower bounds for c_{1Y}, c_{2Y} and $d_Y = c_{3Y} - c_{2Y}$, then independent shifted beta distributions could be considered
  - The variances of those distributions will be determined by X depending on the confidence about the chosen means
- Uncertainty about Y's price P_Y
  - In absence of further information consider a (Gaussian) distribution with mean p_X and variance σ^2 denoting the degree of uncertainty around p_X
- Uncertainty about Y's purchase probability $A_Y(y)$
  - Transform $A_X(x) \rightarrow a\,[A_X(x)]^b$, with $a \in [0, 1]$ (decreasing effect) and $b \in [0, 1]$ (increasing effect)
  - a and b randomly generated to obtain values of $A_Y(y)$

ADVERSARIAL SOFTWARE TESTING

- Uncertainty about Y's fault discovery process $\mathcal{N}_Y(t)$
  - Suppose X has chosen a functional form for $N_X(t)$ and estimated its parameters and obtained an estimate $\tilde{m}_X(t)$ for its mean value function
  - First alternative: generate values of the parameters of $\mathcal{N}_Y(t)$ from distributions centered around X's estimated parameters (e.g. posterior distributions)
  - Second alternative: Bayesian non-parametric approach with mean value function as a random measure M, generated by a Gamma process, conjugate w.r.t. the Poisson process (Lo, 1982)
  - Gamma process centered around $\tilde{m}_X(t)$ so that at each interval $[t_0, t_1]$ the mean value function is generated by a Gamma distribution with mean $\tilde{m}_X(t_1) - \tilde{m}_X(t_0)$
  - The variance of the Gamma distribution could determine how close the fault discovery process $N_Y(t)$ is to $N_X(t)$
  - Further details can be found in Cavallo and Ruggeri (2001)

ADVERSARIAL SOFTWARE TESTING: EXAMPLE

- Example based on Zeephongsekul and Chiera (1995)
- Life cycle length $T = 2000$ days
- Cost parameters: $c_{1X} = 0.5$, $c_{2X} = 1$, $c_{3X} = 5$
- Selling price $p_X = 5000$
- Purchase probability $A_X(t) = 0.0002t(10 - 0.005t)$
- Fault discovery process $N_X(t)$: NHPP with mean value function $m_X(t) = at^c$ (power law process) and MLEs of parameters given by $\widehat{a} = 0.256$ and $\widehat{c} = 0.837$, from Zeephongsekul and Chiera (1995) and based on data from Okumoto (1979)
- Cost function with utility function u_X assumed to be the identity ($\Rightarrow$ Risk neutrality)

ADVERSARIAL SOFTWARE TESTING: EXAMPLE

- Cost parameters follow distributions centered around the c_X values:
  - $c_{1Y} \sim N(0.5, 0.02) = N(c_{1X}, 0.02)$
  - $c_{2Y} \sim N(1, 0.05) = N(c_{2X}, 0.05)$
  - $c_{3Y} \sim N(5, 0.5) = N(c_{3X}, 0.5)$
- Selling price $p_Y \sim N(5000, 250) = N(p_X, 250)$
- Random purchase probability $A_Y(t) \sim \tilde{d} A_X(t)^{\tilde{b}}$, with $\tilde{d} \sim U(0, 1)$ and $\tilde{b} \sim U(0, 1)$
- The random fault discovery process $N_Y(t)$ is a NHPP with random mean value function $m_Y(t) = \tilde{a} t^{\tilde{c}}$ with $\tilde{a} \sim N(0.256, 0.05)$ and $\tilde{c} \sim N(0.837, 0.05)$
- Beliefs of Y over X's release time t given by $t/T \sim \beta e(\alpha, \alpha)$, with $\alpha \sim U(1, 3)$
- Y's random cost function $c_Y(t) = c_{1Y} t + c_{2Y} N_Y(t) + c_{3Y} [N_Y(T) - N_Y(t)]$
- Deterministic utility function U_Y: identity $\Rightarrow$ risk neutrality

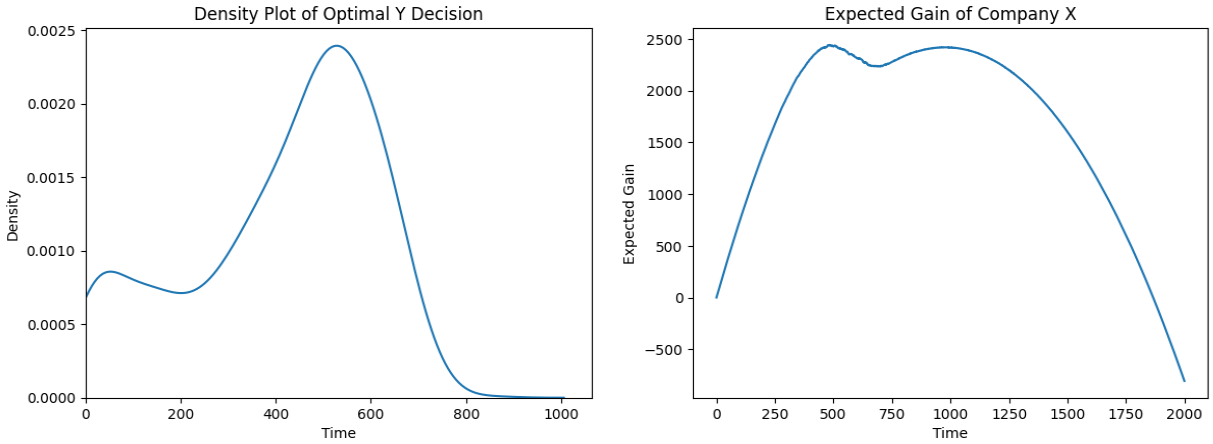
ADVERSARIAL SOFTWARE TESTING: EXAMPLE

- Forecasting Y's release decision
  - Maximise the objective function $M_Y(y) = \int h_Y(x,y)\pi_X^Y(x)dx$
  - For $i = 1, ..., K$
    - Sample c_{1Y}, c_{2Y}, c_{3Y}, p_Y, A_Y, N_Y, α (for π_X^Y, i.e. Y's beliefs on X's release)
    - Given the sampled α_i
      - generate a sample $z_j \sim \beta e(\alpha_i, \alpha_i)$, $j = 1, ..., N$
      - get $x_j = z_j \times T$, $j = 1, ..., N$
    - Monte Carlo approximation $M_Y^i(y)$ through $\frac{1}{N}\sum_{j=1}^{N} h_Y(x_j, y) = \frac{1}{N}[\sum_{x_j<y} h_{Y2}(x_j, y) + \sum_{y<x_j} h_{Y1}(x_j, y)] =$ (omitted)
    - $\Rightarrow$ find $y_i^* = \arg\max_{0\le x\le T} M^i{}_Y(y)$
  - $\Rightarrow$ Get approximate df $\widehat{\Pi}_Y^X(y) = card\{y_i^* : y_i^* \le y\}/K$

ADVERSARIAL SOFTWARE TESTING: EXAMPLE

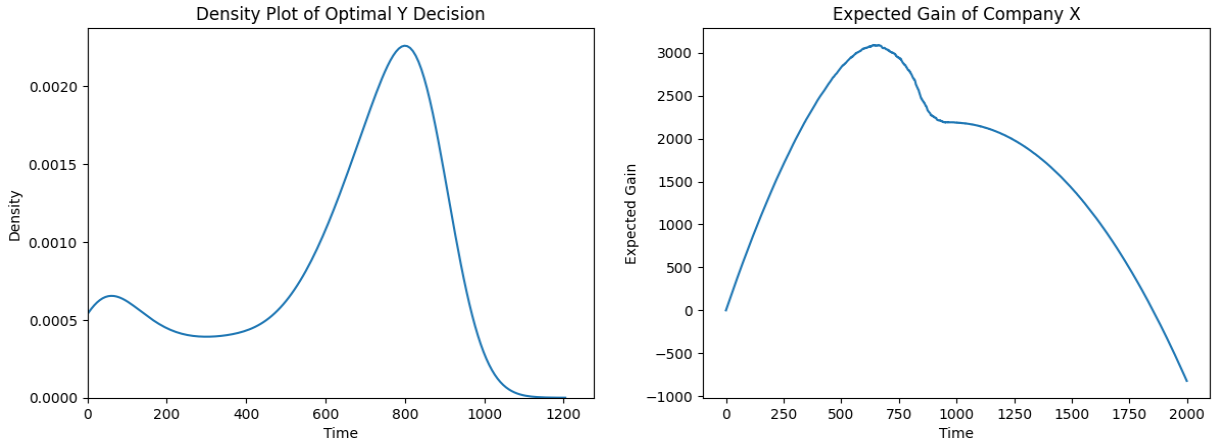
- Deciding X's optimal release
  - Find $x^* = \arg\max_{0 \leq x \leq T} M_X(x)$
  - Maximise the objective function $M_X(x) = \int h_X(x, y) \pi_Y^X(y) dy$
  - Approximate df $\widehat{\Pi}_Y^X(y) = card\{y_i^* : y_i^* \leq y\}/K$
  - Monte Carlo approximation through
    $\frac{1}{K} \sum_{i=1}^{K} h_X(x, y_i^*) = \frac{1}{K} [\sum_{y_i^* \leq x} h_{X2}(x, y_i^*) + \sum_{y_i^* \geq x} h_{X1}(x, y_i^*)] =$ (omitted)

ADVERSARIAL SOFTWARE TESTING: EXAMPLE



- $\beta e(\alpha, \alpha)$ distribution (mean 0.5) on X's release $\Rightarrow$ guess $1000 = 0.5 * 2000$
- LEFT: Y's optimal release time up to 800 days (out of 2000) with some incentive to very early release but the optimal ones are between 300 and 700
- RIGHT: bimodality in X's optimal release, with two possible strategies, one before Y's release and one after it
- X's optimal release occurs on day 483 for an expected gain of 2,442

ADVERSARIAL SOFTWARE TESTING: EXAMPLE



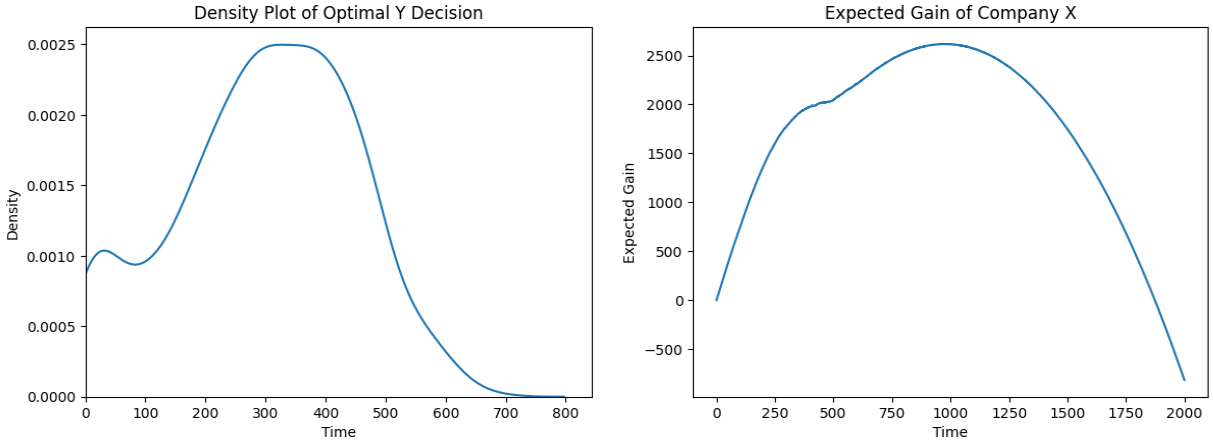
- X thinks that Y thinks that X will release later
  $\Rightarrow \beta e(\alpha, \alpha)$ on X's release replaced with $\beta e(3\alpha, \alpha) \Rightarrow$ guess $1,500 = 0.75{*}2000$

- LEFT: Y's optimal release up to 1200 days with some incentive to very early release and optimal ones between 700 and 900 (compare with 300 and 700)

- RIGHT: X's optimal release is before Y's one

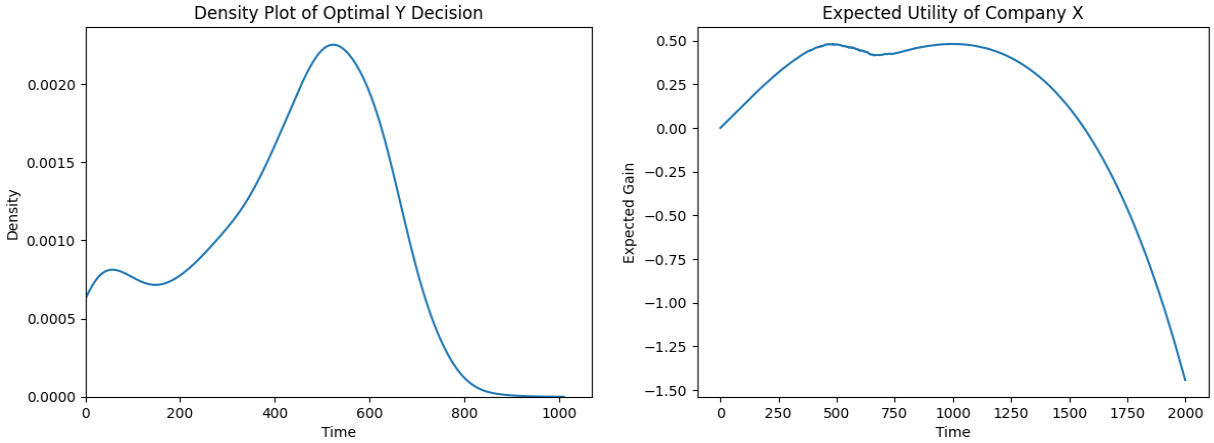
- X's optimal release on day 663 for an expected gain of 3,091 (earlier 483 and 2,442)

ADVERSARIAL SOFTWARE TESTING: EXAMPLE



- X thinks that Y thinks that X will release earlier
  $\Rightarrow \beta e(\alpha, \alpha)$ on X's release replaced with $\beta e(\alpha, 3\alpha) \Rightarrow$ guess $500 = 0.25 * 2000$
- LEFT: Y's optimal release up to 800 days with some incentive to very early release and high-risk early release between 200 and 500 (earlier 300 & 700 and 700 & 900)
- RIGHT: X's optimal release is well after the Y's high-risk one
- X's optimal release on day 978 with expected gain of 2,619 (earlier 483 & 2,442 and 663 & 3,091)

ADVERSARIAL SOFTWARE TESTING: EXAMPLE



- Risk averse $X \Rightarrow$ identity utility replaced with constant absolute risk averse (CARA) model given by $u(x) = 1 - \exp(-\rho x)$, with risk aversion parameter $\rho = 0.001$
- LEFT: Y's optimal release between 300 and 700 unchanged w.r.t. the first plot
- RIGHT: Still bimodal distribution for X's optimal release, but tendency to be more conservative and wait more
- X's optimal release on day 1003 (483 under identity) with expected utility (no more gain!) of 0.48

AST: CURRENT WORK

- Multiple producers
  - Instead of $x < y$ and $x > y$, consider order statistics and position X's release time between $x_{(i-1)}$ and $x_{(i+1)}$ for all i's
  - Similar formulas w.r.t. previous ones
- Multiple decision variables
  - So far the A_X purchase probability has been considered only as a function of the release time but it should depend also on other variables, like price and quality of the software
- Multiple buyers