Supplemental Material for Scalable Operational Decision Optimization in Adversarial Environments

Bo Li and Yevgeniy Vorobeychik
Electrical Engineering and Computer Science
Vanderbilt University
Nashville, TN
{bo.li.2,yevgeniy.vorobeychik}@vanderbilt.edu

1 Model Analysis

Proposition 1.1. *Suppose that $P_A = 0$ and $c = 1$ (i.e., no budget constraint). Then the optimal policy is*

$$q(\vec{x}) = \begin{cases} 1 & \text{if } p(\vec{x}) \geq \frac{G(\vec{x})}{G(\vec{x})+V(\vec{x})} \\ 0 & o.w. \end{cases}$$

Proof. Since we consider only static adversaries and there is no budget constraint, the objective becomes

$$\max_{\vec{q}} \sum_{\vec{x}\in\mathcal{X}} \left[(1-q(\vec{x}))G(\vec{x})(1-p(\vec{x})) - p(\vec{x})v_S(\vec{x})\right],$$

and the only remaining constraint is that $q(\vec{x}) \in [0,1]$ for all $\vec{x}$. Since now the objective function is entirely decoupled for each $\vec{x}$, we can optimize each $q(\vec{x})$ in isolation for each $\vec{x} \in \mathcal{X}$. Rewriting, maximizing the objective for a given $\vec{x}$ is equivalent to minimizing $q(\vec{x})[G(\vec{x}) - p(\vec{x})(G(\vec{x}) + V(\vec{x}))]$. Whenever the right multiplicand is negative, the quantity is minimized when $q(\vec{x}) = 1$, and when it is positive, the quantity is minimized when $q(\vec{x}) = 0$. Since $p(\vec{x}) \geq \frac{G(\vec{x})}{G(\vec{x})+V(\vec{x})}$ implies that the right multiplicand is negative (more accurately, non-positive), the result follows. □

Proposition 1.2. *Suppose that $P_A = 0$ and $c|\mathcal{X}|$ is an integer. Then the optimal policy is to let $q(\vec{x}) = 0$ for all $\vec{x}$ with*

$$p(\vec{x}) < \frac{G(\vec{x})}{G(\vec{x}) + V(\vec{x})}.$$

Rank the remaining $\vec{x}$ in descending order of $p(\vec{x})$ and set $q(\vec{x}) = 1$ for the top $c|\mathcal{X}|$ inputs, with $q(\vec{x}) = 0$ for the rest.

Proof. The LP can be rewritten so as to minimize

$$\sum_{\vec{x}} q(\vec{x})[G(\vec{x}) - p(\vec{x})(G(\vec{x}) + V(\vec{x}))]$$

subject to the budget constraint. By the same argument as above, whenever $p(\vec{x})$ is below the threshold, the optimal $q(\vec{x}) = 0$. Removing the corresponding $\vec{x}$ from the objective, we obtain a special knapsack problem in which the above greedy solution is optimal, since the coefficient on the budget constraint is 1. □

2 Computing Adversary's Best Response

Theorem 2.1. *EVASION is NP-complete.*

Proof. This adversary evasion problem is in NP, as we can non-deterministically pick $a \leq k$ features and verify if $q(\vec{x'}) \leq \lambda$.

We prove that the problem is NP-hard via a reduction from 3-dimensinal matching (3DM). For an arbitrary instance of 3DM, W, Y, and Z are finite, disjoint sets with the same number of d elements. T is a subset of $W \times Y \times Z$, which means T consists of triples (w, y, z) such that $w \in W, y \in Y$, and $z \in Z$. $M \subseteq T$ ($|M| = d$) is a 3-dimensional matching if for any two distinct triples $(w_1, y_1, z_1) \in M$ and $(w_2, y_2, z_2) \in M$, $w_1 \neq w_2$, $y_1 \neq y_2$, and $z_1 \neq z_2$.

Each triple $(w_i, y_i, z_i) \in T$ corresponds to one feature, which controls a set of basis $(s_{w_i}, s_{d+y_i}, s_{2d+z_i})$. There are $n = |T|$ features and $m = |W| + |Y| + |Z| = 3d$ basises, which forms the basis matrix as the figure 1 below. Each elements within the matrix $b_{ji} = 1$ denotes that the jth basis is controlled by the ith feature; otherwise 0. As each feature controls exactly one basis from each part, we have for any feature $i(1 \leq i \leq n)$ and basis $j(1 \leq j \leq m)$, $\sum_{j=1}^{d} b_{ji} = 1$, $\sum_{d+1}^{2d} b_{ji} = 1$, $\sum_{2d+1}^{3d} b_{ji} = 1$, $(d = \frac{1}{3}m)$. Let $k = d$, $\lambda = q(x) - 3d/D$, $(D \geq 3d)$, $\Delta = q(\vec{x}) - q(\vec{x'})$. If

$q(x') \leq \lambda$, we have $\Delta = q(\vec{x}) - q(\vec{xx'}) \geq 3d/D$. Let $\alpha_1 = \alpha_2 = ... = \alpha_m = \frac{1}{2D}$, and x is a vector with all 0. Therefore $\phi_j(x) = \alpha_j(-1)^{s_j x} = \frac{1}{2D}$ for $1 \leq j \leq m$. Consequentially, let $x^{l'}$ denotes the modified instance x', which only differs in feature l with x. If $b_{hl} = 1$, the corresponding basis function would flip the sigh, thus $\phi_h(x^{l'}) = \alpha_l(-1)^{s_h x^{l'}} = -\frac{1}{2D}$. Suppose there are J bases that have been flipped the sign,

$$\begin{aligned}\Delta =& q(\vec{x}) - q(\vec{x'}) = \sum_{j=1}^{m} \alpha_j(-1)^{s_j x} - \sum_{j=1}^{m} \alpha_j(-1)^{s_j x'} \\ =& \left(\sum_{j\in J} \alpha_j(-1)^{s_j x} + \sum_{j\in S\setminus J} \alpha_j(-1)^{s_j x}\right) - \\ & \left(\sum_{j\in J} \alpha_j(-1)^{s_j x'} + \sum_{j\in S\setminus J} \alpha_j(-1)^{s_j x'}\right).\end{aligned}$$

As $\sum_{j\in S\setminus J} \alpha_j(-1)^{s_j x} = \sum_{j\in S\setminus J} \alpha_j(-1)^{s_j x'}$, $\Delta = \frac{1}{2D}|J| - (-\frac{1}{2D})|J| = \frac{|J|}{D}$, which means the decrement of $q(x)$ equals to the number of basises that would flip the sign divided by D. It is easy to see how this construction can be accomplished in polynomial time. Therefore,

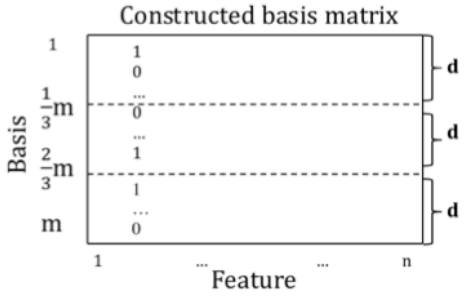


Figure 1: Illustration for the problem construction

suppose there are $a \leq k$ features that can be modified in x to satisfy that $q(\vec{x'}) \leq \lambda$. It follows that $\Delta = q(\vec{x}) - q(\vec{x'}) \geq 3d/D$. Additionally, as each feature only control 3 basises, the total number of basis that would flip the sign is $\Delta = q(\vec{x}) - q(\vec{x'}) \leq 3a/D \leq 3k/D = 3d/D$. It derives that $\Delta = 3d/D$, which means there is no overlap between selected basis. Accordingly, subset M ($|M| = d$) is chosen and each triple $(w_i, y_i, z_i) \in M$ corresponds to the set of controlled basises by feature i. Therefore the total number of elements within the selected subsets in M satisfies $|W| + |Y| + |Z| = \Delta \cdot D = 3d$. So any two selected distinct triples $(w_1, y_1, z_1) \in M$ and $(w_2, y_2, z_2) \in M$, $w_1 \neq w_2$, $y_1 \neq y_2$, and $z_1 \neq z_2$. This means if there is a solution for the adversary evasion problem, there exists a 3-dimensional matching.

Conversely, suppose M is a 3DM. The d selected exclusive triples correspond to $k = d$ specific feature, each of which controls 3 basis. As all the triples are non-overlapped, there are $3d$ different responding basises that would flip the sign, which means $q(\vec{x'}) = q(\vec{x}) - \Delta = q(\vec{x}) - 3d/D = \lambda$. Therefore, the adversary evasion problem can be solved if and only if a 3DM exists. □

Theorem 2.2. *Suppose that the number of inputs in any basis is bounded by a constant c. Then* Approx-Evasion *(Algorithm 1) computes a solution x' to problem 6 which achieves $\hat{\Delta} \geq \frac{\Delta^*}{1+\epsilon}$, where $\hat{\Delta} = \Delta(x')$ in time poly$(n, \frac{1}{\epsilon}, 2^c)$.*

Proof. The operations of $Trim$ and removing from D_l every member that is greater than $q(\vec{x})$ maintain the property that every element of D_l meets our decreasing requirement. For every element d_i in D_i that the corresponding retrieved value is at most $q(x)$, there exists an element $d_k \in D_i$ such that $\frac{Retrieve(d_i)}{(1+\epsilon/2n)^i} \leq Retrieve(d_k) \leq Retrieve(d_i)$. This must hold for the optimal Δ^*, therefore there exists an element $d \in D_n$ that $\frac{\Delta^*}{(1+\epsilon/2n)^n} \leq Retrieve(d) \leq \Delta^*$. Thus $\frac{\Delta^*}{Retrieve(d)} \leq (1 + \frac{\epsilon}{2n})^n$. As this inequality must also hold for $\hat{\Delta}$, $\frac{\Delta^*}{\hat{\Delta}} \leq (1 + \frac{\epsilon}{2n})^n$. Since $\lim_{n\to\infty}(1 + \epsilon/2n)^n = e^{\epsilon/2}$ and $\frac{d}{dn}(1 + \epsilon/2n)^n > 0$, the function $(1 + \epsilon/2n)^n$ increases with n and we have $(1 + \epsilon/2n)^n \leq e^{\epsilon/2} \leq 1 + \epsilon/2 + (\epsilon/2)^2 \leq 1 + \epsilon$.
Therefore, $\hat{\Delta} \geq \frac{\Delta^*}{1+\epsilon}$.

Next we show that it is a polynomial-time approximation scheme based on a restrictive feature group size c, which is the maximum size of each feature group obtained from Algorithm 2. To analyze the run time, we need to derive the bound on the length of D_i. After trimming between groups of features, successive elements d and d' of D_i must have the relationship $d'/d > 1 + \epsilon/2n$. That is, they must differ by a factor of at least $1 + \epsilon/2n$. Each list, therefore, constraints the value 0, possibly value $\delta > 0$, which is a small number less than the minimal α value, and up to $\lfloor \log_{1+\epsilon/2n} \frac{q(x)}{\delta} \rfloor$. Therefore we can derive that the number of elements in each list D_i is at most

$$2^c \left(\log_{1+\epsilon/2n} \frac{q(x)}{\delta} + 2\right) = 2^c \left(\frac{\ln \frac{q(x)}{\delta}}{\ln(1 + \epsilon/2n)} + 2\right) \quad (1)$$

$$\leq 2^c \left(\frac{2n(1 + \epsilon/2n) \ln \frac{q(x)}{\delta}}{\epsilon} + 2\right) \quad (2)$$

$$< 2^c \left(\frac{3n \ln \frac{q(x)}{\delta}}{\epsilon} + 2\right). \quad (3)$$

Therefore, this bound of the list length is polynomial in the size of the input n when $c \leq \log_2 n$. Since the running time of *ApproxAdversaryEvasion* is polynomial in the lengths of the D_i, we conclude that there is a polynomial-time approximation scheme ($O(n \cdot 2^c)$) with respect to the restricted feature group size as c ($c \leq \log_2 n$). □

Algorithm 1 returns the approximate solution of maximum Δ to obtain a minimal $q(\vec{x'})$ by modifying less or equal to k features.

Algorithm 1 $ApproxAdversaryEvasion(F, q(\vec{x}), \epsilon, k)$

$n \leftarrow |F|$
$D_0 \leftarrow \{(\emptyset, 0)\}$ // tuple $d_i = (feaSet, value) \in D$
$G = GenFeaGroup(F, S)$
$l \leftarrow 0$
for $i \leftarrow 1$ to $|G|$ **do**
  for $j \leftarrow 1$ to $|g_i|$ **do**
    $l \leftarrow l + 1$ // merge two tuple-lists by $d_i.value$
    $D_l \leftarrow MergeTuple(D_{l-1}, AddFea(D_{l-1}, f_{ij}, k))$
  end for
  $D_l \leftarrow Trim(D_l, \epsilon/2n)$
  remove elements from D_l that $d_l.value > q(\vec{x})$
end for
let d^* correspond to the maximum $d.value$ in D_n
return d^*

As the length of D_i can be 2^i, which makes the merge algorithm take exponential time, here we employ the Algorithm 4 to trim the list length. The idea is that if some combination of features make the decrease of $q(\vec{x})$ similar, then only one combination should be kept. This means that with a trimming parameter δ, for any element d_i removed from D_i, there is an element d_j that approximates d_i, that is,
$\frac{Retrieve(d_i)}{1+\delta} \leq Retrieve(d_j) \leq Retrieve(d_i)$.

Algorithm 2 $GenFeaGroup(F, S)$

$G \leftarrow \emptyset$
$n \leftarrow |F|$
$m \leftarrow |S|$
for $j \leftarrow 1$ to m **do**
  for $i \leftarrow 1$ to n **do**
    $g_j \leftarrow \emptyset$
    if $s_{ji} = 1$ **then**
      $g_j \leftarrow g_j \cup f_i$
    end if
  end for
  $G \leftarrow G \cup g_j$
end for
$G \leftarrow DisjointSet(G)$ // convert G to disjoint-sets
return G

However, this $Trim$ action can only be done for features that have no common basises to avoid missing qualified feature combination. Therefore, Algorithm 2 is applied to group the features that need to be added as a whole before $Trim$; and algorithm 3 helps to form different feature combinations and guarantee only less or equal to k features are considered.

Algorithm 3 $AddFea(D, f, k)$

$m \leftarrow |D|$
$D' \leftarrow \emptyset$
for $i \leftarrow 1$ to m **do**
  if $size(i.set \cup f) \leq k$ **then**
    $t'_i.set \leftarrow t_i.set \cup f$
    $t'_i.value \leftarrow Retrieve(t'_i.set)$
    insert t'_i into ordered D' by $t'_i.value$
  end if
end for
return D'

Algorithm 4 $Trim(D, \epsilon)$

$m \leftarrow |D|$
$D' \leftarrow d_1$
$last \leftarrow d_1.value$
for i $\leftarrow$ 2 to m **do**
  if $d_i.value > last \cdot (1 + \epsilon)$ **then**
    append d_i onto the end of D'
    $last \leftarrow d_i.value$
  end if
end for
return D'

Finally, for each feature combination we would use the algorithm 5 to obtain the corresponding value based on the chosen bases. Our goal is to find the feature combination that reduce the most from $q(\vec{x})$ by flipping fewer features, which means the strategy x' can have a higher chance to pass the classifier after less modification on the original "ideal" instance x.

Algorithm 5 $Retrieve(d)$

$w \leftarrow \emptyset$
for $f_i \in d$ **do**
  $w \leftarrow w \oplus w_{f_i}$ // w_{f_i} is basis set controled by f_i
end for
$v \leftarrow \sum_{s_j \in w} -2\alpha_{xj}$ // α_{xj} is the actual value in x
return v

3 Experiments

Here we test the AAS scheme with the same set up of simulations on the feature space of 500, and similar results shown as below have demonstrated the consistency and robustness of our proposed approach.

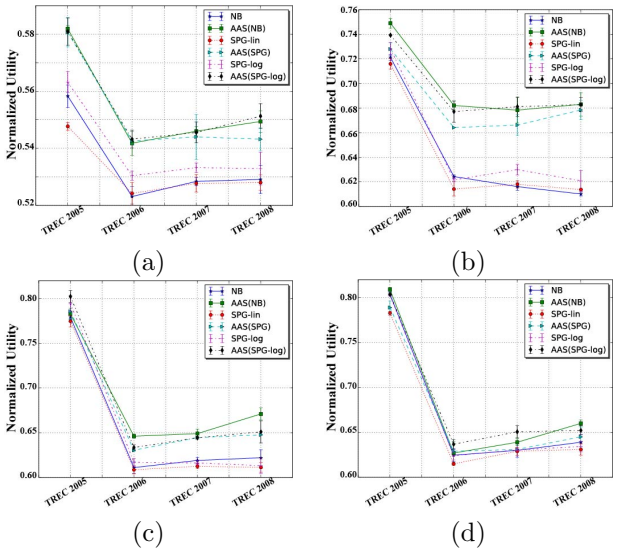


Figure 2: Comparison of normalized utility on TREC data, trained on year 2005, and tested on years 2005-2008. Our method is labeled as $AAS(\cdot)$, where the parameter is the classifier that serves to provide $p(x)$. The following parameters are used: $\delta = 0.2$,$V(x) = G(x) = 1$, $P_A = 1$ (a) c=0.1; (b) c=0.3; (c) c=0.5; (d) c=0.9.

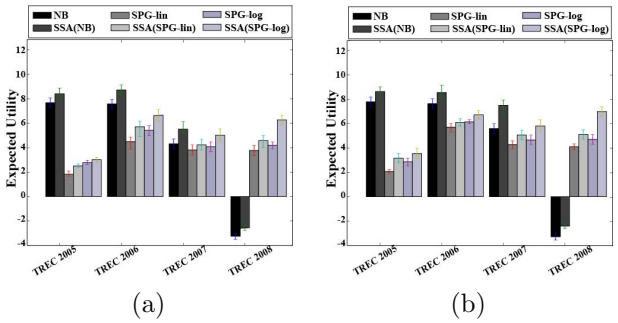


Figure 3: Comparison of the expected utility assuming $P_A = 1$, $V(x) = G(x) = 1$; (a) $c = 0.1$; (b) $c = 0.3$.

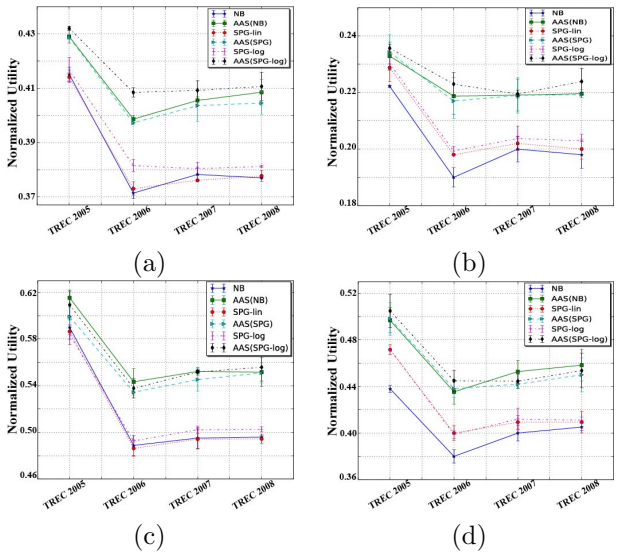


Figure 4: Comparison of normalized utility on TREC data, trained on year 2005, and tested on years 2005-2008. Our method is labeled as $AAS(\cdot)$, where the parameter is the classifier that serves to provide $p(x)$. The following parameters are used: $\delta = 0.2$,$G(x) = 1$, $P_A = 1$ (a) $V(x) = 2$, c=0.1; (b) $V(x) = 10$, c=0.1; (c) $V(x) = 2$, c=0.3; (d) $V(x) = 10$, c=0.3.

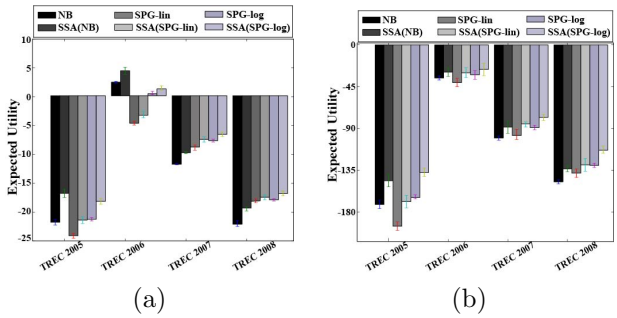


Figure 5: Comparison of the expected utility assuming $P_A = 1$; (a) $V(x) = 2$; (b) $V(x) = 10$. $c = 0.3$.

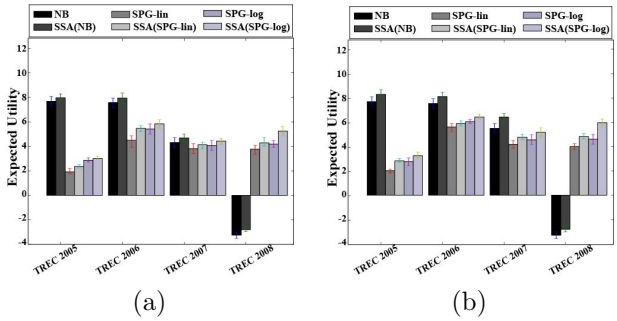
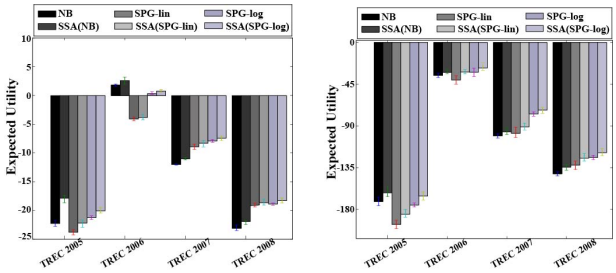
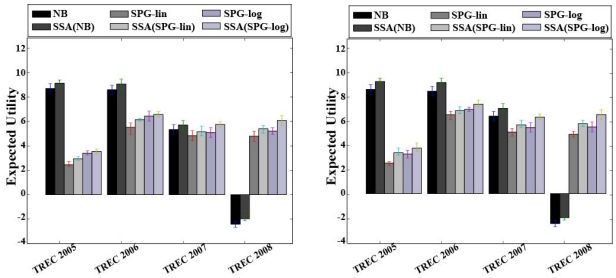


Figure 6: Comparision of the expected utility assuming $P_A = 1$, introducing parameter error with 0.2 for δ; (a) $c = 0.1$; (b) $c = 0.3$.



(a) (b)

Figure 7: Comparison of the expected utility assuming $P_A = 1$, introducing parameter error with 0.2 for δ; (a) $V(x) = 2$; (b) $V(x) = 10$. $c = 0.3$.



(a) (b)

Figure 8: Comparison of the expected utility assuming $P_A = 1$, introducing adversarial model error; (a) $c = 0.1$; (b) $c = 0.3$.