A Local Search Algorithm for the Witsenhausen's Counterexample

Shih-Hao Tseng and Ao Tang

***Abstract*— We propose a local search algorithm to find an optimal controller of the Witsenhausen's counterexample, which still remains unknown. Via variational analysis, a few necessary conditions are derived, and the algorithm is guided by the conditions. The analysis does not require preliminary knowledge about the property of the target cost function, nor does it make any assumption about the functional form of the controller. As such, our method is applicable for a wide range of problems with similar structures as the Witsenhausen's counterexample. Through numerical simulations, we demonstrate that our algorithm outperforms all previous results on the benchmark case. Our results also manifest some structural properties of the first-stage state variable, including that it is not piecewise affine but nonlinear within each step, and that its shape changes from piecewise continuous to sigmoid-like as the first-stage cost coefficient increases. Furthermore, a variety of parameter settings are fed to the algorithm, and the generated results can serve as the best upper bounds known thus far under those parameter settings.**

I. Introduction

Witsenhausen's counterexample [1] is a 2-stage stochastic control problem with the objective

$$\min \mathbb{E}\left[k^2 u_0(y_0)^2 + x_2^2\right],$$

where $k \in \mathbb{R}$ and

$$\begin{aligned} x_1 &= x_0 + u_0(y_0), \qquad y_0 = x_0, \\ x_2 &= x_1 - u_1(y_1), \qquad y_1 = x_1 + w. \end{aligned} \tag{1}$$

$x_0 \sim \mathcal{N}(0, \sigma^2)$ and $w \sim \mathcal{N}(0, 1)$ are normal distributed random variables with probability distribution functions $f_X(x_0)$ and $f_W(w)$, where $\mathcal{N}(0, \sigma^2)$ is the normal random variable with mean 0 and variance σ^2. The goal is to minimize the cost function $\mathbb{E}\left[k^2 u_0(y_0)^2 + x_2^2\right]$ by Borel control functions (controllers) $u_0(y_0)$ and $u_1(y_1)$, whose existence are proven in [1].

Witsenhausen demonstrates that the optimal affine controllers can perform strictly worse than a non-linear controller in this simple 2-stage linear quadratic Guassian (LQG) system [1]. Two key questions, which are still open, then emerge from the counterexample: what the optimal controller is and how to efficiently obtain it.

In the literature, the two questions are tackled analytically and numerically. The analytical arguments mainly focus on the necessary conditions of an optimal controller [1]–[5] and the lower bound on the objective value [6], [7]. The optimal controller is proven neither affine [1], [2] nor step-like [3]. [4] introduces transport theoretic formulation of the counterexample and infers that the optimal x_1 is a strictly increasing function with a real analytic left inverse. Some other necessary conditions for the quantile functions of the controllers are given in [5]. Since the discrete version of the counterexample is shown NP-complete [8], [6] establishes a lower bound on the optimal cost and shows that there exists either a linear or a state-quantization strategy which achieves within constant factor of the optimal cost, regardless of the problem parameters. Such constant factor is also derived in [7] for the finite-dimensional version of the counterexample.

On the other hand, numerical approximation methods are developed to realize good solutions in practice. Mostly, the numerical methods target a class of functions and tune the parameters to find the best controller in the class. For instance, step functions are targeted by [9], [10]; discrete output functions are assumed in [11], [12]; [13] considers only the functions that can be expressed as a linear combination of given basis functions; and [14] works on piecewise affine functions. The methods to tune the parameters are quite diverse. [9] utilizes ordinal optimization. [10] carefully selects the discontinuous points. [11] discretizes the counterexample and formulates it as a game with several agents. The agents learn with fading memory and form the controller. [12] uses noisy channel relaxation to find the optimal controller, which searches from a large k to the designated k. For each k, [12] discretizes the output of x_1 and updates x_1 and u_1 alternatively based on necessary conditions. [13] gives the suboptimal controller by optimizing the linear combination coefficients and the parameters of the basis functions. [14] takes deterministic annealing approach. Starting with an affine function and a large offset (controlled by "temperature") of the cost, [14] optimizes the coefficients of the affine function over an augmented cost. As the offset decreases, the affine function becomes piecewise affine. We refer the reader to [15]–[17] for a comprehensive survey and the connection to some other dynamic decision problems.

A. Contribution and Organization

The crux of our approach centers on variational analysis. We first derive necessary conditions of the optimal controllers, then a local search algorithm follows by satisfying all necessary conditions. Unlike previous approaches [9]–[14], preliminary knowledge regarding the cost function property, especially the minimum mean square error (MMSE) knowledge, is not mandatory for our approach. In addition, we don't make any assumption about the functional form of the controllers, and hence we won't search only within a class of functions. Also, our approach does not augment the cost function as noisy channel relaxation [12] and deterministic

Supported by NSF grant CNS-1544761.

Shih-Hao Tseng and Ao Tang are with the School of Electrical and Computer Engineering, Cornell University, Ithaca, NY 14853, USA. Emails: {st688, at422}@cornell.edu

 DOI: 10.1109/CDC.2017.8264401

annealing [14], which avoids the precomputation before handling the target cost function. Therefore our approach can be easily applied to general control problems with similar structures as the Witsenhausen's counterexample.

We first define the ideas *minimizer* and *local Nash minimizer* in Section II, where an optimal controller is a minimizer. Based on the definitions, we derive first and second order necessary conditions, which lead to a local search algorithm in Section III. In Section IV, the algorithm is applied to $\sigma = 5$ under various k. We not only show that the algorithm outperforms all prior results on the benchmark case $k = 0.2$ but also provide our results under several different k, which can serve as upper bounds on the optimal costs. Furthermore, we discuss the proper choice of the initial function in Section V. Although $x_1(x_0) = x_0$ is the best initial function based on simulation results, there is a class of initial functions which result in similar costs. In Section VI, we demonstrate that our method can also be applied to other problems of similar structures by dealing with the inventory control problem. Finally, we conclude the paper in Section VII.

II. Analysis

To approach the problem, we first express the cost function as a functional of the control functions. Through variational analysis, we can derive the necessary conditions for optimal control functions, which are named *minimizers* in this work.

A. Cost Function as a Functional

State equations (1) imply the equivalence of finding $u_0(y_0)$ and finding $x_1(x_0)$. To avoid messy expressions, we state our results in terms of $x_1(x_0)$ instead of $u_0(y_0)$.

By (1), the cost function can be written as a sum of integrals

$$\begin{aligned}\mathcal{J}\left[x_1,u_1\right] &= \mathbb{E}\left[k^2u_0(y_0)^2 + x_2^2\right]\\ &= k^2\int \left(x_1(x_0)-x_0\right)^2 f_X(x_0)\ dx_0\\ &+ \iint \left(x_1(x_0)-u_1(y_1)\right)^2 f_X(x_0)f_W(w)\ dx_0\ dy_1\end{aligned}$$

where all integrals are taken from $-\infty$ to ∞. For simplicity of notation, we omit the limits of the integrals in the following context when integrating over $\mathbb{R}$. Notice that we integrate over y_1 instead of $w = y_1 - x_1(x_0)$ in the second term, which results from variable substitution at each x_0.

B. Minimizers and Local Nash Minimizers

We say (x_1, u_1) is a *minimizer* of $\mathcal{J}\left[x_1,u_1\right]$ if the pair of functions (x_1, u_1) attains the minimum of the functional $\mathcal{J}$. In other words, for arbitrary functions δx_1 and δu_1,

$$\mathcal{J}\left[x_1+\delta x_1, u_1+\delta u_1\right] \geq \mathcal{J}\left[x_1,u_1\right]. \tag{2}$$

Our goal can be translated as finding a minimizer of $\mathcal{J}$.

Inspired by Nash equilibrium, (x_1, u_1) is a *local Nash minimizer* of $\mathcal{J}\left[x_1,u_1\right]$ if

$$\begin{aligned}&\mathcal{J}\left[x_1+\delta x_1, u_1\right] \geq \mathcal{J}\left[x_1,u_1\right],\\ &\mathcal{J}\left[x_1, u_1+\delta u_1\right] \geq \mathcal{J}\left[x_1,u_1\right]\end{aligned} \tag{3}$$

for arbitrary δx_1 and δu_1. (3) is a weaker condition than (2): Clearly, a minimizer is a local Nash minimizer, but a local Nash minimizer is not necessarily a minimizer. Instead of finding minimizers directly, we aim to find "good" local Nash minimizers, which may be "similar" to minimizers.

C. Necessary Conditions for Local Nash Minimizers

To obtain a local Nash minimizer, we derive the necessary conditions for local Nash minimizers.

Let $\epsilon \ll 1$ be a small constant and δx_1 be a bounded function. We can Taylor-expand the functional $\mathcal{J}\left[x_1+\epsilon\delta x_1,u_1\right]$:

$$\begin{aligned}\mathcal{J}\left[x_1+\epsilon\delta x_1,u_1\right] &= \mathcal{J}\left[x_1,u_1\right]\\ &+ \epsilon\int \frac{\delta\mathcal{J}}{\delta x_1}\left[x_1,u_1\right](x_0)\delta x_1(x_0)\ dx_0\\ &+ \frac{\epsilon^2}{2}\int \frac{\partial}{\partial x_1}\frac{\delta\mathcal{J}}{\delta x_1}\left[x_1,u_1\right](x_0)\delta x_1^2(x_0)\ dx_0\\ &+ O(\epsilon^3).\end{aligned}$$

We remark that the expansion is taken against analytical functions of ϵ, instead of $x_1(x_0)$, at each x_0. Also, each notation is chosen to be consistent with the definitions of functional derivatives in the literature.

If (x_1, u_1) is a local Nash minimizer of $\mathcal{J}\left[x_1,u_1\right]$, we have

$$\mathcal{J}\left[x_1+\epsilon\delta x_1,u_1\right] - \mathcal{J}\left[x_1,u_1\right] \geq 0.$$

Since $\epsilon \ll 1$, we can ignore the higher order terms and obtain the first order condition (FOC)

$$\int \frac{\delta\mathcal{J}}{\delta x_1}\left[x_1,u_1\right](x_0)\delta x_1(x_0)\ dx_0 = 0,$$

and the second order condition (SOC)

$$\int \frac{\partial}{\partial x_1}\frac{\delta\mathcal{J}}{\delta x_1}\left[x_1,u_1\right](x_0)\delta x_1^2(x_0)\ dx_0 \geq 0.$$

The FOC and the SOC of u_1 are derived through the same procedure. We defer the calculation details to Appendix A.

If (x_1, u_1) is a local Nash minimizer of $\mathcal{J}\left[x_1,u_1\right]$, FOC requires

$$\begin{aligned}\int \frac{\delta\mathcal{J}}{\delta x_1}\left[x_1,u_1\right](x_0)\delta x_1(x_0)\ dx_0 = 0,\\ \int \frac{\delta\mathcal{J}}{\delta u_1}\left[x_1,u_1\right](y_1)\delta u_1(y_1)\ dy_1 = 0.\end{aligned}$$

Since δx_1 and δu_1 are arbitrary, FOC implies

$$\frac{\delta\mathcal{J}}{\delta x_1}\left[x_1,u_1\right](x_0) = 0, \qquad \frac{\delta\mathcal{J}}{\delta u_1}\left[x_1,u_1\right](y_1) = 0 \tag{4}$$

almost everywhere with respect to Lebesgue measure.

Similarly, the second order condition (SOC) requires

$$\begin{aligned}\int \frac{\partial}{\partial x_1}\frac{\delta\mathcal{J}}{\delta x_1}\left[x_1,u_1\right](x_0)\delta x_1^2(x_0)\ dx_0 \geq 0,\\ \int \frac{\partial}{\partial u_1}\frac{\delta\mathcal{J}}{\delta u_1}\left[x_1,u_1\right](y_1)\delta u_1^2(y_1)\ dy_1 \geq 0.\end{aligned}$$

Since δx_1 and δu_1 are arbitrary, SOC implies

$$\frac{\partial}{\partial x_1}\frac{\delta\mathcal{J}}{\delta x_1}\left[x_1,u_1\right](x_0) \geq 0, \qquad \frac{\partial}{\partial u_1}\frac{\delta\mathcal{J}}{\delta u_1}\left[x_1,u_1\right](y_1) \geq 0 \tag{5}$$

almost everywhere with respect to Lebesgue measure.

Algorithm 1: Alternative Update

1: Fix $x_1(x_0)$ and compute $u_1(y_1)$ through (6).
2: Fix $u_1(y_1)$ and update $x_1(x_0)$ using (7) or (8).

D. Closed Form Expression of $u_1(y_1)$

(4) leads to a closed form expression of $u_1(y_1)$:

$$\begin{aligned} u_1(y_1) &= \frac{\int x_1(x_0) f_X(x_0) f_W(y_1 - x_1(x_0))\ dx_0}{\int f_X(x_0) f_W(y_1 - x_1(x_0))\ dx_0} \\ &= \mathbb{E}\left[x_1(x_0) | x_1(x_0) + w = y_1\right]. \end{aligned} \tag{6}$$

Given $x_1(x_0)$, (6) leads to a unique $u_1(y_1)$, which attains the minimum of $\mathcal{J}[x_1, u_1]$ since

$$\frac{\partial}{\partial u_1} \frac{\delta \mathcal{J}}{\delta u_1} [x_1, u_1](y_1) = 2 \int f_X(x_0) f_W(w)\ dx_0 > 0.$$

The result is also known as MMSE estimator in the literature.

Here the analysis leads to the same result without knowing MMSE estimator in advance. Essentially, the analysis provides a systematic way to explore the properties of a cost function, which makes it applicable for general problems.

III. Algorithms

We develop algorithms based on (4) and (5) to explore local Nash minimizers. The basic idea is to update x_1 and u_1 alternatively. However, the alternative update suffers from initialization sensitivity and sampling granularity issues. As a result, we introduce local denoising procedure and propose our local search algorithm. The convergence of the algorithm is then briefly discussed.

A. Alternative Update

To minimize $\mathcal{J}[x_1, u_1]$, we alternatively fix x_1 and u_1 and search for a local Nash minimizer as in Algorithm 1.

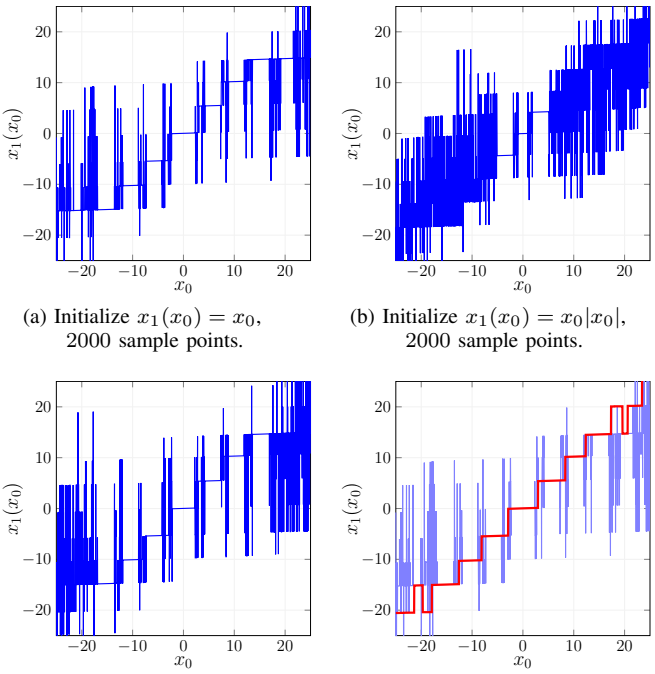
Given $x_1(x_0)$, local Nash minimizer $u_1(y_1)$ can be found by (6). However, local Nash minimizer $x_1(x_0)$ cannot be derived directly from a given $u_1(y_1)$. As such, we apply revised Newton's Method to update and find local Nash minimizer $x_1(x_0)$:

$$x_1(x_0) \leftarrow x_1(x_0) - \frac{\frac{\delta \mathcal{J}}{\delta x_1}[x_1, u_1](x_0)}{\left| \frac{\partial}{\partial x_1} \frac{\delta \mathcal{J}}{\delta x_1} [x_1, u_1](x_0) \right|}. \tag{7}$$

If $\frac{\partial}{\partial x_1} \frac{\delta \mathcal{J}}{\delta x_1} [x_1, u_1](x_0) = 0$, we select a small constant τ as the step size and update x_1 by Gradient Method:

$$x_1(x_0) \leftarrow x_1(x_0) - \tau \frac{\delta \mathcal{J}}{\delta x_1} [x_1, u_1](x_0). \tag{8}$$

The step size τ should be small enough such that it will not overshoot, while choosing τ too small can lead to slow convergence. In the simulations, we choose τ to be the smallest distance between two sample points.



(a) Initialize $x_1(x_0) = x_0$, 2000 sample points.
(b) Initialize $x_1(x_0) = x_0|x_0|$, 2000 sample points.
(c) Initialize $x_1(x_0) = x_0$, 3000 sample points.
(d) $x_1(x_0)$ should be "noiseless" as the thick line.

Fig. 1. Algorithm 1 is sensitive to the initial function $x_1(x_0)$ and the sampling granularity (number of samples procured over the support to approximate continuous functions).

B. Local Denoising and Local Search Algorithm

Ideally, we want to start from an initial $x_1(x_0)$ and repeat Algorithm 1 to obtain a local Nash minimizer of $\mathcal{J}[x_1, u_1]$, which may be close to a (global) minimizer. However, Algorithm 1 is sensitive to the initial function $x_1(x_0)$ and the sampling granularity (Fig. 1(a) – (c)).

The sampling granularity issue results from the nature of numerical methods: Target functions are approximated by step functions, which take values at points sampled from the support. In general, the more the samples, the less is the difference between the target and the approximation. Nevertheless, more samples can also include more noise during the update of the approximating function.

Algorithm 1 generates $x_1(x_0)$, which looks like a function mixed with some noise. Intuitively, $x_1(x_0)$ should be "similar" within a local neighborhood, i.e., left- or right-continuous, and we can hence get rid of the noise (Fig. 1(d)). Thus, we denoise the resulting x_1 by finding the best possible x_1 locally. To do so, we first define

$$\begin{aligned} C_X(a, x_0) &= k^2 (a - x_0)^2 f_X(x_0) \\ &\quad + \int (a - u_1(y_1))^2 f_X(x_0) f_W(y_1 - a)\ dy_1 \end{aligned}$$

for a given $u_1(y_1)$, which allows us to express $\mathcal{J}[x_1, u_1]$ as

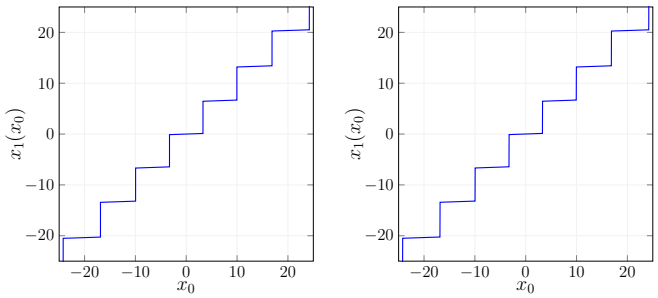
$$\mathcal{J}[x_1, u_1] = \int C_X(x_1(x_0), x_0)\ dx_0.$$

If (x_1, u_1) is a minimizer of $\mathcal{J}[x_1, u_1]$, we have

$$x_1(x_0) = \operatorname*{argmin}_a C_X(a, x_0).$$

Algorithm 2: Local Search Algorithm

Given $x_1(x_0)$, the local interval radius $r \in \mathbb{R}$, the number of repetition $N \in \mathbb{N}$, and the precision p.
repeat
  Repeat Algorithm 1 to update (x_1, u_1) N times.
  for all x_0 **do**
    $x_1(x_0) \leftarrow \operatorname{argmin}_{x' \in B_r(x_0)} C_X(x_1(x'), x_0)$.
  end for
until $p > \int \left| \frac{\delta \mathcal{J}}{\delta x_1}[x_1, u_1] \right| \, dx_0$



(a) Initialize $x_1(x_0) = x_0$. (b) Initialize $x_1(x_0) = x_0|x_0|$.

Fig. 2. Algorithm 2 takes different initial functions and gives the same result.

Especially, let $B_r(x_0) = [x_0 - r, x_0 + r]$, we have $x_0 \in B_r(x_0)$ and

$$x_1(x_0) = \operatorname*{argmin}_{x_1(x'):x' \in B_r(x_0)} C_X(x_1(x'), x_0). \tag{9}$$

(9) means that $x_1(x_0)$ must perform the best at x_0 comparing to any other $x_1(x')$ where x' is within a neighborhood of x_0. Otherwise, if there exists $x_1(x')$ that performs better than $x_1(x_0)$, we can improve the solution by setting $x_1(x_0) \leftarrow x_1(x')$. This property enables us to "denoise" and improve the solution (x_1, u_1). Applying the local denoising condition (9) to Algorithm 1, we create Algorithm 2. In Fig. 2, it is shown that Algorithm 2 avoids the issues in Fig. 1.

Though we don't have a convergence proof for Algorithm 2, it converges numerically. However, we observe that repeated Algorithm 1 can sometimes get stuck in some repetitive states, regardless of the size of τ. Choosing smaller τ only reduces the difference between the repetitive functions. Algorithm 2 introduces the local denoising condition (9) which resolves this problem.

IV. Numerical Results

Under $\sigma = 5$ and various k, we evaluate Algorithm 2 with the precision parameter $p = 10^{-9}$, the local interval radius $r = 0.25$, the number of repetition $N = 15$, and the initial function $x_1(x_0) = x_0$.

x_1 and u_1 are supported on $[-25, 25]$ and $[-30, 30]$ in the simulations. The supports are partitioned evenly by 16000 points to approximate x_1 and u_1 by step functions.[1]

We first compare the performance of Algorithm 2 with major prior results on the benchmark case $k = 0.2$ in Table I. Our result 0.166897 outperforms all major prior results.

[1]Notice that we do not *assume* x_1 and u_1 to be step functions.

TABLE I
Our result and major prior results ($k = 0.2$).

Source	Total Cost $\mathcal{J}$
Our result	0.166897
Mehmetoglu et al. [14]	0.16692291
Karlsson et al. [12]	0.16692462
Li et al. [11]	0.1670790
Baglietto et al. [13]	0.1701
Witsenhausen [1]	0.40425320

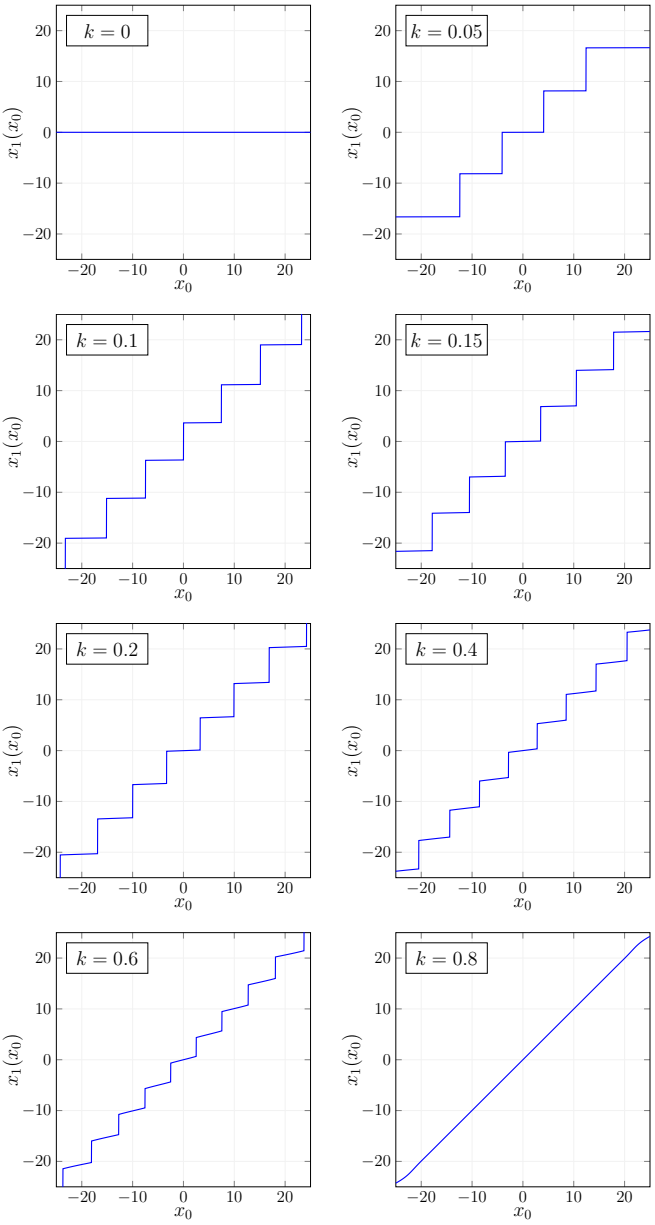


Fig. 3. The resulting $x_1(x_0)$ given by Algorithm 2 under different k.

Besides the benchmark case, we also vary k and apply Algorithm 2 to find the local Nash minimizers and the corresponding costs $\mathcal{J}$. The resulting $x_1(x_0)$ are plotted in Fig. 3 and the costs are listed in Table II. To the best of our knowledge, the results in Table II are the best known thus far.

We remark that even though each $x_1(x_0)$ in Fig. 3 looks like a piecewise affine function, it is actually not except for

TABLE II

THE COST $\mathcal{J}$ OBTAINED BY ALGORITHM 2 UNDER DIFFERENT k.

k	Cost $\mathcal{J}$	k	Cost $\mathcal{J}$	k	Cost $\mathcal{J}$
0	0	0.5	0.640974	1	0.961467
0.05	0.015517	0.55	0.718803	1.5	0.961498
0.1	0.052292	0.6	0.791935	2	0.961509
0.15	0.104019	0.65	0.858579	2.5	0.961514
0.2	0.166897	0.7	0.916458	3	0.961516
0.25	0.237978	0.75	0.961424	4	0.961519
0.3	0.314824	0.8	0.961436	5	0.961520
0.35	0.395337	0.85	0.961446	6	0.961521
0.4	0.477652	0.9	0.961454	7	0.961521
0.45	0.560067	0.95	0.961461	8	0.961522

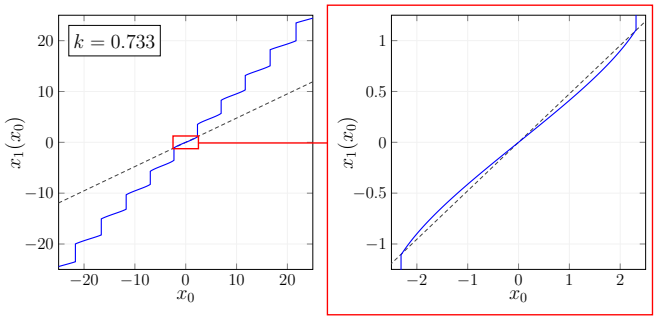
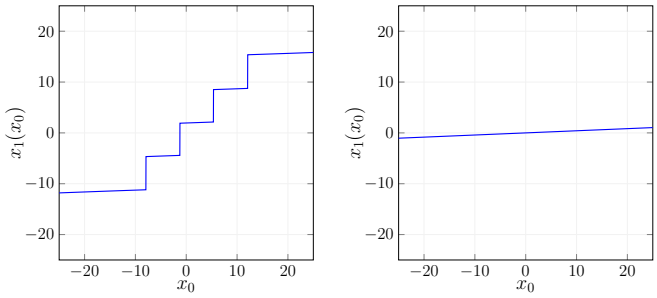


Fig. 4. $x_1(x_0)$ is not piecewise affine ($k = 0.733$ as an example).



(a) Initialize $x_1(x_0) = e^{x_0}$, resulting cost $\mathcal{J}$: 0.168075.

(b) Initialize $x_1(x_0) = 0$, resulting cost $\mathcal{J}$: 0.959991.

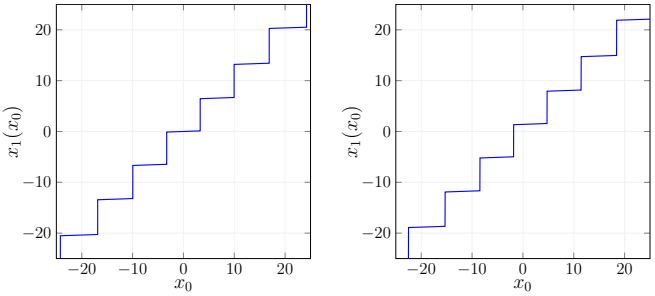
Fig. 5. Initial functions that do not have range $\mathbb{R}$ can lead to disparate local Nash minimizers ($k = 0.2$).

$k = 0$. To exaggerate the difference, we zoom in the case $k = 0.733$ to examine a step in Fig. 4. The dashed line is a linear function, and $x_1(x_0)$ is clearly not affine.

V. DISCUSSION

Although Algorithm 2 is shown effective in Section IV, the choice of the initial function is actually critical to its performance. Feeding the functions that do not have range $\mathbb{R}$ as the initials can trap Algorithm 2 in disparate local Nash minimizers whose associated costs $\mathcal{J}$ may be far from the optimal. Two examples are shown in Fig. 5: $x_1(x_0) = e^{x_0}$ with range $(0, \infty]$ and $x_1(x_0) = 0$.

Several continuous functions with range $\mathbb{R}$, such as x_0, $x_0|x_0|$, and x_0^3, can be guided to the best known local Nash minimizer by Algorithm 2. But having range $\mathbb{R}$ does not guarantee the convergence to the same local Nash minimizer. For instance, initializing $x_1(x_0) = x_0 + 2$ leads to a different local Nash minimizer with similar cost (Fig. 6).



(a) Initialize $x_1(x_0) = x_0$, resulting cost $\mathcal{J}$: 0.166897.

(b) Initialize $x_1(x_0) = x_0 + 2$, resulting cost $\mathcal{J}$: 0.166898.

Fig. 6. Algorithm 2 converges to local Nash minimizers with similar costs ($k = 0.2$).

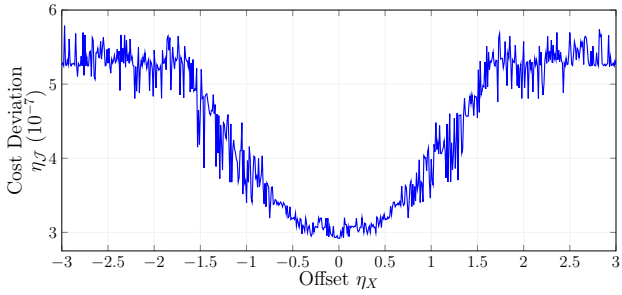


Fig. 7. Initializing Algorithm 2 with $x_1(x_0) = x_0 + \eta_X$ results in similar cost $\mathcal{J}[x_1, u_1] = 0.166897 + \eta_{\mathcal{J}}$ ($k = 0.2$).

It is questionable if there exists an offset η_X such that the initial function $x_1(x_0) = x_0 + \eta_X$ yields a better local Nash minimizer. Therefore, we initialize Algorithm 2 with $x_1(x_0) = x_0 + \eta_X$ and collect the resulting cost $\mathcal{J}$ in Fig. 7. It turns out the difference among the costs is not significant enough (10^{-7}) to determine if those local optima are different in consideration of possible simulation granularity bias. However, $\eta_X = 0$ gives the lowest cost $\mathcal{J}$ based on the experiments.

VI. APPLICATION TO INVENTORY CONTROL

Algorithm 2 can be applied to not only the Witsenhausen's counterexample but also some other problems of similar structures. As an example, we apply our method to approach the well-known inventory control problem [18, Chapter 4.2]. For simplicity of presentation, we consider the 2-stage version under the state dynamic

$$x_{m+1} = x_m + u_m(x_m) - w_m$$

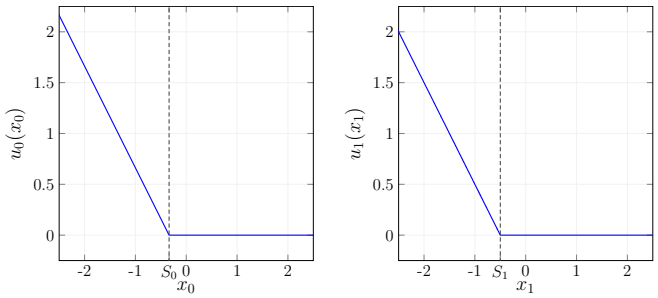
for $m = 0, 1$, where x_0 and w_m are distributed uniformly over $[-1, 1]$, and the controllers $u_m(x_m)$ are non-negative.

The objective is to minimize the following functional:

$$\mathcal{J}[u_0, u_1] = \mathbb{E}\left[\sum_{m=0}^{1} \xi u_m(x_m) + \gamma\left(x_m + u_m(x_m) - w_m\right)\right]$$

where $\gamma(a) = h\max(0, a) + l\max(0, -a)$. In this example, we set $\xi = 1$ and $h = l = 2$.

It is known that the optimal controllers $u_0(x_0)$ and $u_1(x_1)$ can be computed by dynamic programming [18, Chapter



(a) First stage controller $u_0(x_0)$. (b) Second stage controller $u_1(x_1)$.

Fig. 8. Algorithm 2 finds the optimal controllers of the inventory control problem.

4.2], which are given by

$$u_m(x_m) = \begin{cases} S_m - x_m & x_m \le S_m, \\ 0 & x_m > S_m \end{cases}$$

for $m = 0, 1$ with $S_0 \approx -0.338$ and $S_1 = -0.5$.

Instead of using dynamic programming, we find the optimal controllers by Algorithm 2. Since $\frac{\partial}{\partial u_m}\frac{\delta \mathcal{J}}{\delta u_m}[u_0, u_1](x_m) = 0$ almost everywhere with respect to Lebesgue measure, we update $u_m(x_m)$ for $m = 0, 1$ alternatively by

$$u_m(x_m) \leftarrow \max\left(0, u_m(x_m) - \tau \frac{\delta \mathcal{J}}{\delta u_m}[u_0, u_1](x_m)\right),$$

and then denoise them by

$$u_m(x_m) \leftarrow \operatorname*{argmin}_{u_m(x'):x' \in B_r(x_m)} C_{U_m}(u_m(x'), x_m)$$

with C_{U_m} defined similar to the way that C_X is defined.

Initializing $u_m(x_m) = \max(0, x_m)$, Algorithm 2 successfully finds the optimal controllers as in Fig. 8 (under suitable choices of τ, p, r, and N).

VII. Conclusion

A local search algorithm (Algorithm 2) is presented, which outperforms all previous results and discovers $x_1(x_0)$ that is non-linear even within each step. $x_1(x_0)$ transforms from a piecewise continuous function to a sigmoid-like function when k increases. Unlike the previous attempts, the algorithm is based on variational analysis instead of heuristics. The whole analysis does not require prior knowledge about the cost function property such as MMSE estimator. Also, no assumption about the functional form of the controller is required for our method. As a result, the method can be applied to not only the Witsenhausen's counterexample but also a wide range of problems which have similar structures.

Appendix

A. FOC and SOC

In FOC, $\frac{\delta \mathcal{J}}{\delta x_1}$ and $\frac{\delta \mathcal{J}}{\delta u_1}$ are called the first order functional derivatives, which are given by

$$\begin{aligned}\frac{\delta \mathcal{J}}{\delta x_1}[x_1, u_1](x_0) &= 2k^2 (x_1(x_0) - x_0) f_X(x_0) \\ &\quad + \int D'_X(x_0, y_1) f_X(x_0) f_W(w)\, dy_1, \\ \frac{\delta \mathcal{J}}{\delta u_1}[x_1, u_1](y_1) &= \int 2 (u_1(y_1) - x_1(x_0)) f_X(x_0) f_W(w)\, dx_0,\end{aligned}$$

where

$$\begin{aligned}D'_X(x_0, y_1) &= 2 (x_1(x_0) - u_1(y_1)) \\ &\quad + (y_1 - x_1(x_0)) (x_1(x_0) - u_1(y_1))^2.\end{aligned}$$

And the partial derivatives of the first order functional derivatives in SOC are

$$\begin{aligned}\frac{\partial}{\partial x_1}\frac{\delta \mathcal{J}}{\delta x_1}[x_1, u_1](x_0) &= 2(k^2 - 1) f_X(x_0) \\ &\quad + \int D''_X(x_0, y_1) f_X(x_0) f_W(w)\, dy_1, \\ \frac{\partial}{\partial u_1}\frac{\delta \mathcal{J}}{\delta u_1}[x_1, u_1](y_1) &= 2 \int f_X(x_0) f_W(w)\, dx_0,\end{aligned}$$

where

$$\begin{aligned}D''_X(x_0, y_1) = &\ ((y_1 - x_1(x_0)) (x_1(x_0) - u_1(y_1)) + 2)^2 \\ &- (x_1(x_0) - u_1(y_1))^2.\end{aligned}$$

References

[1] H. S. Witsenhausen, "A counterexample in stochastic optimum control," *SIAM J. Control*, vol. 6, no. 1, pp. 131–147, 1968.

[2] S. Mitter and A. Sahai, "Information and control: Witsenhausen revisited," in *Learning, control and hybrid systems*. Springer, 1999, pp. 281–293.

[3] R. Bansal and T. Basar, "Stochastic teams with nonclassical information revisited: When is an affine law optimal?" in *Proc. IEEE ACC*, 1986, pp. 45–50.

[4] Y. Wu and S. Verdú, "Witsenhausen's counterexample: A view from optimal transport theory," in *Proc. IEEE CDC*, 2011, pp. 5732–5737.

[5] W. M. McEneaney and S. H. Han, "Optimization formulation and monotonic solution method for the Witsenhausen problem," *Automatica*, vol. 55, pp. 55–65, 2015.

[6] S. Y. Park, P. Grover, and A. Sahai, "A constant-factor approximately optimal solution to the Witsenhausen counterexample," in *Proc. IEEE CDC*, 2009, pp. 2881–2886.

[7] P. Grover, S. Y. Park, and A. Sahai, "Approximately optimal solutions to the finite-dimensional Witsenhausen counterexample," *IEEE Trans. Autom. Control*, vol. 58, no. 9, pp. 2189–2204, 2013.

[8] C. H. Papadimitriou and J. Tsitsiklis, "Intractable problems in control theory," *SIAM J. Control Optim.*, vol. 24, no. 4, pp. 639–654, 1986.

[9] M. Deng and Y.-C. Ho, "An ordinal optimization approach to optimal control problems," *Automatica*, vol. 35, no. 2, pp. 331–338, 1999.

[10] J. T. Lee, E. Lau, and Y.-C. Ho, "The Witsenhausen counterexample: A hierarchical search approach for nonconvex optimization problems," *IEEE Trans. Autom. Control*, vol. 46, no. 3, pp. 382–397, 2001.

[11] N. Li, J. R. Marden, and J. S. Shamma, "Learning approaches to the Witsenhausen counterexample from a view of potential games," in *Proc. IEEE CDC*, 2009, pp. 157–162.

[12] J. Karlsson, A. Gattami, T. J. Oechtering, and M. Skoglund, "Iterative source-channel coding approach to Witsenhausen's counterexample," in *Proc. IEEE ACC*, 2011, pp. 5348–5353.

[13] M. Baglietto, T. Parisini, and R. Zoppoli, "Numerical solutions to the Witsenhausen counterexample by approximating networks," *IEEE Trans. Autom. Control*, vol. 46, no. 9, pp. 1471–1477, 2001.

[14] M. Mehmetoglu, E. Akyol, and K. Rose, "A deterministic annealing approach to Witsenhausen's counterexample," in *Proc. IEEE ISIT*, 2014, pp. 3032–3036.

[15] Y.-C. Ho, "Review of the Witsenhausen problem," in *Proc. IEEE CDC*, 2008, pp. 1611–1613.

[16] T. Başar, "Variations on the theme of the Witsenhausen counterexample," in *Proc. IEEE CDC*, 2008, pp. 1614–1619.

[17] M. Rotkowitz, "On information structures, convexity, and linear optimality," in *Proc. IEEE CDC*, 2008, pp. 1642–1647.

[18] D. P. Bertsekas, *Dynamic Programming and Optimal Control*. Athena Scientific, 2005, vol. 1.