NBER WORKING PAPER SERIES

# IMPROVING POLICY FUNCTIONS IN HIGH-DIMENSIONAL DYNAMIC GAMES

Carlos A. Manzanares
Ying Jiang
Patrick Bajari

Working Paper 21124
http://www.nber.org/papers/w21124

NATIONAL BUREAU OF ECONOMIC RESEARCH
1050 Massachusetts Avenue
Cambridge, MA 02138
April 2015

Improving Policy Functions in High-Dimensional Dynamic Games
Carlos A. Manzanares, Ying Jiang, and Patrick Bajari
NBER Working Paper No. 21124
April 2015
JEL No. C44,C55,C57,C73,L1

## ABSTRACT

In this paper, we propose a method for finding policy function improvements for a single agent in high-dimensional Markov dynamic optimization problems, focusing in particular on dynamic games. Our approach combines ideas from literatures in Machine Learning and the econometric analysis of games to derive a one-step improvement policy over any given benchmark policy. In order to reduce the dimensionality of the game, our method selects a parsimonious subset of state variables in a data-driven manner using a Machine Learning estimator. This one-step improvement policy can in turn be improved upon until a suitable stopping rule is met as in the classical policy function iteration approach. We illustrate our algorithm in a high-dimensional entry game similar to that studied by Holmes (2011) and show that it results in a nearly 300 percent improvement in expected profits as compared with a benchmark policy.

Carlos A. Manzanares
Department of Economics
Vanderbilt University
VU Station B, Box #351819
2301 Vanderbilt Place, Nashville, TN 37235
carlos.a.manzanares@Vanderbilt.Edu

Ying Jiang
Department of Economics
University of Washington
305 Savery Hall, Box 353330
Seattle, WA 98195-3330
jiangy82@uw.edu

Patrick Bajari
University of Washington
331 Savery Hall
UW Economics Box 353330
Seattle, Washington 98195-3330
and NBER
Bajari@uw.edu

## 1 Introduction

Many dynamic games of interest in economics have state spaces that are potentially very large, and solution algorithms considered in the economics literature do not scale to problems of this size. Consider the game of Chess, which is a two-player board game involving sequential moves on a board consisting of 64 squares and which can be characterized as Markovian, with the existing board configuration serving as the current state. Since games end after the maximum allowable 50 number of moves, solving for pure Markov-perfect equilibrium strategies is in principle achievable using backward induction, since all allowable positions of pieces and moves could be mapped into an extensive form game tree.[1] In practice, however, there are at least two challenges to implementing this type of solution method.

The first challenge is the high number of possible branches in the game tree. For example, an upper bound on the number of possible terminal nodes is on the order of $10^{46}$.[2] Fully solving for equilibrium strategies requires computing and storing state transition probabilities at each of a very large number of nodes, which is both analytically and computationally intractable.

The second challenge is deriving the strategies of opponents. Equilibrium reasoning motivates fixed-point methods for deriving equilibrium best responses. However, it is not clear that equilibrium assumptions will generate good approximations of opponent play in Chess, since players may engage in suboptimal strategies, making Nash-style best responses derived *a priori* possibly suboptimal. Similarly, it is not clear whether developing and solving a stylized version of Chess would produce strategies relevant for playing the game.

Recently, researchers in computer science and artificial intelligence have made considerable progress deriving strategies for high-dimensional dynamic games such as Chess using a general approach very different from that used by economists, which has two broad themes. First, to derive player strategies, they rely more heavily on data of past game play than on equilibrium assumptions. Second, instead of focusing on deriving optimal strategies, they focus on continually improving upon the best strategies previously implemented by other researchers or game practitioners. This general approach has provided a series of successful strategies for high-dimensional games.[3]

In this paper, we propose an approach which proceeds in this spirit, combining ideas developed by researchers in computer science and artificial intelligence with those developed by econometricians for studying dynamic games to solve for policy improvements for a single agent in high-

[1] Recently, researchers have found that this is even more complicated for games like Chess, which may have no uniform Nash equilibria in pure or even mixed positional strategies. See Boros, Gurvich and Yamangil (2013) for this assertion.

[2] See Chinchalkar (1996).

[3] These include, *inter-alia*, the strategy of the well-publicized computer program "Deep Blue" (developed by IBM), which was the first machine to beat a reigning World Chess Champion, and the counterfactual regret-minimization algorithm for the complex multi-player game Texas Hold'em developed by Bowling, Burch, Johanson and Tammelin (2015), which has been shown to beat successful players in practice.

dimensional dynamic games, where strategies are restricted to be Markovian. For an illustration, we consider the problem of computing a one-step improvement policy for a single retailer in the game considered in Holmes (2011). He considers the decision by chain store retailers of where to locate physical stores. We add to his model the decision of where to locate distribution centers as well. In our game, there are 227 physical locations in the United States and two rival retailers, which each seek to maximize nation-wide profits over seven years by choosing locations for distribution centers and stores.

This game is complicated for several reasons. First, store location decisions generate both own firm and competitor firm spillovers. On the one hand, for a given firm, clustering stores in locations near distribution centers lowers distribution costs. On the other hand, it also cannibalizes own store revenues, since consumers substitute between nearby stores. For the same reason, nearby competitor stores lower revenues for a given store. Second, the game is complicated because it is dynamic, since we make distribution center and store decisions irreversible. This forces firms to consider strategies such as spatial preemption, whereby firm entry in earlier time periods influences the profitability of these locations in future time periods.

Using our algorithm, we derive a one-step improvement policy for a hypothetical retailer and show that our algorithm generates a 289 percent improvement over a strategy designed to approximate the actual facility location patterns of Wal-Mart. This algorithm can be characterized by two attributes that make it useful for deriving strategies to play high-dimensional games. First, instead of deriving player strategies using equilibrium assumptions, we utilize data on a large number of previous plays of the game. Second, we employ an estimation technique from Machine Learning that reduces the dimensionality of the game in a data-driven manner, which simultaneously makes estimation feasible.

The data provides us with sequences of actions and states describing many plays of the game, indexed by time, and the assumption that strategies are Markovian allows us to model play in any particular period as a function of a set of payoff relevant state variables.[4] Using this data, we estimate opponent strategies as a function of the state, as well as a law of motion. We also borrow from the literature on the econometrics of games and estimate the choice-specific value function, making the choice-specific value function the dependent variable in an econometric model.[5] After fixing the strategy of the agent for all time periods beyond the current time period using a benchmark strategy, we use the estimated opponent strategies and law of motion to simulate and then estimate the value of a one-period deviation from the agent's strategy in the current period.[6]

[4]We note that in principle there is some scope to test the Markov assumption. For example, we could do a hypothesis test of whether information realized prior to the current period is significant after controlling for all payoff relevant states in the current period.

[5]See Pesendorfer and Schmidt-Dengler (2008) for an example using this approach. Also see Bajari, Hong, and Nekipelov (2013) for a survey on recent advances in game theory and econometrics.

[6]In practice, the benchmark strategy could represent a previously proposed strategy that represents the highest

This estimate is used to construct a one-step improvement policy by maximizing the choice-specific value function in each period as a function of the state, conditional on playing the benchmark strategy in all time periods beyond the current one.

Since the settings we consider involve a large number of state variables, estimating opponent strategies, the law of motion, and the choice-specific value function in this algorithm is infeasible using conventional methods. For example, in our spatial location game, one way to enumerate the current state is to define it as a vector of indicator variables representing the national network of distribution center and store locations for both firms. This results in a state vector that contains 1817 variables and achieves an average cardinality in each time period on the order of $10^{85}$.[7] Although this enumeration allows us to characterize opponent strategies, the law of motion, and choice-specific value functions of this game as completely non-parametric functions of the state variables, it is potentially computationally wasteful and generates three estimation issues. First, the large cardinality of the state vector makes it unlikely that these models are identified. Second, if they are identified, they are often inefficiently estimated since there are usually very few observations for any given permutation of the state vector. Moreover, when estimating the choice-specific value function, remedying these issues by increasing the scale of the simulation is computationally infeasible. Third, when the number of regressors is large, researchers often find in practice that many of these regressors are highly multicollinear, and in the context of collinear regressors, out-of-sample predictive accuracy under most norms is often maximized using a relatively small number of regressors. To the extent that some state variables are relatively unimportant, these estimation and computational issues motivate the use of well-specified approximations. However, in high-dimensional settings, it is often difficult to know *a priori* which state variables are important.

As a consequence, we utilize a technique from Machine Learning which makes estimation and simulation in high-dimensional contexts feasible through an approximation algorithm that selects the parsimonious set of state variables that minimizes the loss associated with predicting our out-

payoffs agents have been able to find in practice. For example, in spectrum auctions, we might use the well known "straightforward bidding" strategy or the strategy proposed by Bulow, Levin, and Milgrom (2009).

[7] $1817 = 227 * 2$ (own distribution center indicators for two merchandise classes) + $227 * 2$ (own store indicators for two types of stores) + $227 * 2$ (opponent distribution center indicators for two merchandise classes) + $227 * 2$ (opponent store indicators for two types of stores) + 1 (location-specific population variable). The state space cardinality for the second time period is calculated as follows. In each time period, we constrain the number of distribution centers and stores that each firm can open, and at the start of the game (in the first time period), we allocate firm facilities randomly as described in the Appendix. Only locations not currently occupied by firm $i$ facilities of the same type are feasible. In the first time period, the number of feasible locations for placing facilities of each type in the second time period include: 220, 226, 211, and 203, and among available locations, each firm chooses 4 distribution centers and 8 stores of each type. The order of the resulting cardinality of the state space in the second period (only including the cardinality of the state attributable to firm $i$ facilities; also not including the cardinality of the population variable) is the product of the possible combinations of distribution centers and store locations of each type, i.e. $\begin{pmatrix} 220 \\ 4 \end{pmatrix} * \begin{pmatrix} 226 \\ 4 \end{pmatrix} * \begin{pmatrix} 211 \\ 8 \end{pmatrix} * \begin{pmatrix} 203 \\ 8 \end{pmatrix} \approx 10^7 * 10^7 * 10^{13} * 10^{13} = 10^{43}$. The cardinality of the state attributable to firm $-i$ facilities is calculated in a similar manner, and the total cardinality of the state (not considering the population variable) is the product of the cardinality attributable to firm $i$ and $-i$ facilities. State space cardinality calculations attributable to firm $i$ facilities for all time periods are available in the Appendix.

comes of interest using a fixed metric. Machine Learning refers to a set of methods developed and used by computer scientists and statisticians to estimate models when both the number of observations and controls is large, and these methods have proven very useful in practice for predicting accurately in cross-sectional settings. See Hastie *et al.* (2009) for a survey. There has been relatively little attention to the problem of estimation when the number of controls is very large in econometrics until recently. See Belloni, Chernozhukov, and Hansen (2010) for a survey of some recent work. In our illustration, we utilize a Machine Learning method known as Component Wise Gradient Boosting (CWGB), which we describe in detail in Section 2.2. This technique was developed and characterized theoretically in a series of articles by Breiman (1998, 1999), Friedman *et al.* (2000), and Friedman (2001). Also see Hastie *et al.* (2009) for an introduction to the method.[8] As with many other ML methods, CWGB works by projecting the estimand functions of interest onto a low-dimensional set of parametric basis functions of regressors, with the regressors and basis functions chosen in a data-driven manner. CWGB methods can accommodate non-linearity in the data generating process, are computationally simple, and, unlike many other non-linear estimators, are not subject to problems with convergence in practice. As a result of the estimation process, CWGB often reduces the number of state variables dramatically, and we find that these parsimonious approximations perform well in our application as compared with other variable and model selection procedures, suggesting that many state representations in economics might be computationally wasteful.[9] For example, we find that choice-specific value functions in our spatial location game are well-approximated by between 6 and 7 state variables (chosen from the original 1817).

Our algorithm contributes a data-driven method for deriving policy improvements in high-dimensional dynamic Markov games which can be used to play these games in practice. High-dimensional dynamic games include, for example, Chess, Go, Texas Hold 'em, spectrum auctions, and the entry game we study in this paper. It also extends a related literature in approximate dynamic programming (ADP). ADP is a set of methods developed primarily by engineers to study Markov decision processes in high-dimensional settings. See Bertsekas (2012) for an extensive survey of this field. Within this literature, our approach is most related to the rollout algorithm, which is a technique that also generates a one-step improvement policy based on a choice-specific value function estimated using simulation. See Bertsekas (2013) for a survey of these algorithms. Although originally developed to solve for improvement policies in dynamic engineering applications, the main idea of rollout algorithms–obtaining an improved policy starting from another subopti-

[8] We choose this method because among the methods considered, it had the highest level of accuracy in out-of-sample prediction. We note that there are relatively few results about "optimal" estimators in high-dimensional settings. In practice, researchers most often use out-of-sample fit as a criteria for deciding between estimators.

[9] As with other Machine Learning estimators, the relative performance of CWGB as compared with other methods may depend on the application considered. In general, Machine Learning methods do not necessarily dominate existing estimators in econometrics. For example, Hansen (forthcoming) shows that whether the Lasso estimator generates a lower mean-squared error than OLS depends on the extent to which many of the true coefficients on regressors are equal to zero, i.e. the extent to which the parameter space is "sparse."

mal policy using a one-time improvement–has been applied to Markov games by Abramson (1990) and Tesauro and Galperin (1996). We appear to be the first to formalize the idea of estimating opponent strategies and the law of motion as inputs into the simulation and estimation of the choice-specific value function when applying rollout to multi-agent Markov games. This is facilitated by separating the impact of opponent strategies on state transitions from the payoff function in the continuation value term of the choice-specific value function, which is a separation commonly employed in the econometrics of games literature. Additionally, we extend the rollout literature by using a recently developed Machine Learning estimator to select regressors in high-dimensional contexts in a data-driven manner.

We note that in practice there are several limitations to the approach we describe. A first is that we do not derive an equilibrium of the game. Hence we are unable to address the classic questions of comparative statics if we change the environment. That said, to the best of our knowledge, the problem of how to derive equilibria in games with very large state spaces has not been solved in general. We do suspect that finding a computationally feasible way to derive policy improvements in this setting may be useful as researchers make first steps in attacking this problem. A second limitation is that we assume opponent strategies are fixed. In practice, competitors might reoptimize their strategies after observing our play.[10] A third limitation is that we do not derive theoretical characterizations of the optimality properties of our Machine Learning estimator or policy function improvements, i.e. whether our policy function improvements converge generally. Many Machine Learning estimators, including the one we use in this paper, simultaneously perform model selection and estimation at the same time. This feature can generate corner solutions, making the derivation of fundamental estimator properties, such as consistency and asymptotic distributions, potentially more challenging. Although Machine Learning estimators are typically used on datasets that are very large, often making sampling distributions a less important criteria than predictive accuracy, sampling distributions may influence the convergence of our policy function improvements in the context of smaller samples.

That said, it is not clear that equilibrium theory is a particularly useful guide to play in these settings, even if theory tells us that equilibrium exists and is unique. In economics, much of the guidance has been based on solving very stylized versions of these games analytically or examining the behavior of subjects in laboratory experiments. Our method complements these approaches by providing strategies useful for playing high-dimensional games in practice. Artificial intelligence and computer science researchers, along with decision makers in industry and policy have used data as an important input into deriving strategies to play games. We believe that our example shows that certain economic problems may benefit from the intensive use of data and modeling based on

[10]It may be possible to mitigate this problem to some extent in practice. For example, if researchers can observe newly reoptimized opponent play, they can reestimate opponent strategies and use our method to derive new policy improvements to compete against these strategies.

econometrics and Machine Learning.

The rest of the paper proceeds as follows. The next section describes the class of games for which our algorithm is useful and then describes Component-Wise Gradient Boosting as well as an algorithm for generating policy function improvements. Section 3 describes two applications, the chain store entry game described previously as well as an application in progress where we seek to improve-upon a recently developed strategy for Texas Hold'em. Section 4 concludes.

## 2 Method Characterization

### 2.1 Game Model

In this section, we formally characterize a class of games for which our method is useful for finding policy function improvements.

#### 2.1.1 Preliminaries

We define a discrete number of time periods, denoted as $t = 1, ..., T$ with $T < \infty$, and a discrete number of players, denoted as $i \in \mathcal{I} \equiv \{1, ..., N\}$. We refer to the competitors of a reference player $i$ as players $-i$, where $-i \equiv \{\neg (i \cap \mathcal{I})\}$. Finally, we denote observations of player actions and states (defined below) found within data using the index $l = 1, ..., L$ with $L < \infty$.

#### 2.1.2 State

Define a state vector, denoted as $\mathbf{s}_t$ for each $t$, as

$$\mathbf{s}_t \equiv (s_{1t}, ..., s_{K_s t}) \in \mathcal{S}_t \subseteq \mathbb{R}^{K_s}$$

where $s_{1t}, ..., s_{K_s t}$ represent a set of $K_s$ state variables at time $t$, $\mathcal{S}_t$ represents the support of $\mathbf{s}_t$, and $\mathbb{R}^{K_s}$ represents the $K_s$-ary Cartesian product over $K_s$ sets of real numbers $\mathbb{R}$. The set $\mathcal{S}_t$ can be discrete, continuous, or both. In practice, the number of state variables, i.e. $K_s$, can be large. At time $t$, the state at time $t+1$ is random and is denoted as $\mathbf{S}_{t+1}$ with realization $\mathbf{S}_{t+1} = \mathbf{s}_{t+1}$.

Importantly, in the econometrics of games literature, researchers often seek to model functions of the state, such as payoffs, opponent strategies, and the law of motion (defined formally in the subsections that follow), using general functional forms. In these settings, the cardinality of $\mathcal{S}_t$, denoted as $|\mathcal{S}_t|$, becomes important, and this cardinality is potentially very large. For example, in our entry game application, although $K_s = 1817$, the average $|\mathcal{S}_t|$ (average by time period) is greater than $10^{85}$. See the Appendix for a derivation of the cardinality of the state in our entry game application.

We also define the dimension-reduced state vector that remains as a result of the Component-Wise Gradient Boosting (CWGB) estimation process described in Section 2.2. Define this state vector, denoted as $\widetilde{\mathbf{s}}_t$ for all $t$, as

$$\widetilde{\mathbf{s}}_t \equiv \left(s_{1t}, ..., s_{K_{s,GB}t}\right) \in \widetilde{\mathcal{S}}_t \subseteq \mathbb{R}^{K_{s,GB}}$$

where $K_{s,GB}$ represents the number of state variables that remain after CWGB, such that $K_{s,GB} \leq K_s$. In practice, it is often the case that the dimension of $\widetilde{\mathbf{s}}_t$ is much smaller than the dimension of the original state vector $\mathbf{s}_t$, i.e. $K_{s,GB}$ is much smaller than $K_s$, making the cardinality of $\widetilde{\mathcal{S}}_t$ much smaller than the cardinality of $\mathcal{S}_t$. This cardinality reduction plays an important role in making the forward simulation step of our algorithm feasible (see Section 2.2), since we only simulate realizations of the dimension-reduced state rather than realizations of the original state.

### 2.1.3 Actions

Each player chooses a vector of feasible actions, denoted as $\mathbf{a}_{it}$ for all $t$, and defined as

$$\mathbf{a}_{it} \equiv (a_{1it}, ..., a_{K_a it}) \in \mathcal{A}_{it} \subseteq \mathbb{R}^{K_a}$$

where $a_{1it}, ..., a_{K_a it}$ represent a set of $K_a$ action variables at time $t$, $\mathcal{A}_{it}$ represents the discrete support of $\mathbf{a}_{it}$, and $\mathbb{R}^{K_a}$ represents the $K_a$-ary Cartesian product over $K_a$ sets of real numbers $\mathbb{R}$. We abuse the notation of $\mathbf{a}_{it}$ by suppressing its possible dependence on $\mathbf{s}_t$. We further define the profile of actions across all competitors $-i$ as $\mathbf{a}_{-it} \equiv (\mathbf{a}_{1t}, ..., \mathbf{a}_{i-1t}, \mathbf{a}_{i+1t}, ..., \mathbf{a}_{Nt})$ with support $\mathcal{A}_{-it} \subseteq \mathbb{R}^{K_a(N-1)}$, and a profile of actions across all players including $i$ as $\mathbf{a}_t \equiv (\mathbf{a}_{1t}, ..., \mathbf{a}_{Nt})$ with support $\mathcal{A}_t \subseteq \mathbb{R}^{K_a N}$.

The action vector serves as either an outcome variable or set of regressors in the models estimated in Section 2.2. In practice, when actions represent an outcome, we redefine $\mathcal{A}_{it}$ so as to constrain its cardinality to be sufficiently small, since our method requires us to evaluate policies at each action in $\mathcal{A}_{it}$ for a subset of states. This often means that we will redefine the problem such that each action we consider is a scalar rather than a vector, which is separately denoted as $a_{it}$ to distinguish it from $\mathbf{a}_{it}$. For example, in our entry game application, $K_a = 1$ and $a_{it}$ is a scalar 0, 1 indicator over the choice of placing a facility in a given location, which gives $|\mathcal{A}_{it}| = 2$ for all $t$.

When actions represent a set of regressors, as is the case when estimating the law of motion in Section 2.2, we allow the dimensionality of the action vector to remain high. As is the case with the state vector, the CWGB estimation process selects a subset of the original action variables, which we define as

$$\widetilde{\mathbf{a}}_{it} \equiv \left(a_{1it}, ..., a_{K_{a,GB}it}\right) \in \widetilde{\mathcal{A}}_{it} \subseteq \mathbb{R}^{K_{a,GB}}$$

where $K_{a,GB} \leq K_a$.

### 2.1.4 Law of Motion

We make the following assumption on the evolution of states over time.

**Assumption (A1)**. States evolve according to the Markov property, i.e. for all $\mathbf{s}_{t+1} \in \mathcal{S}_{t+1}$, $\mathbf{s}_t \in \mathcal{S}_t$, and $\mathbf{a}_t \in \mathcal{A}_t$,

$$F\left(\mathbf{S}_{t+1} \leq \mathbf{s}_{t+1} | \mathbf{s}_1, ..., \mathbf{s}_t, \mathbf{a}_1, ..., \mathbf{a}_t\right) = F\left(\mathbf{S}_{t+1} \leq \mathbf{s}_{t+1} | \mathbf{s}_t, \mathbf{a}_t\right)$$

where,

$$F\left(\mathbf{S}_{t+1} \leq \mathbf{s}_{t+1} | \mathbf{s}_t, \mathbf{a}_t\right) \equiv \Pr\left(S_{1t+1} \leq s_{1t+1}, ..., S_{K_s t+1} \leq s_{K_s t+1} | \mathbf{s}_t, \mathbf{a}_t\right)$$

Here, $F(.)$ represents the cdf of $\mathbf{S}_{t+1}$, which we deem the law of motion. In some applications, the law of motion may vary across time periods, although we abstract away from this possibility for expositional simplicity. We allow the transitions for a subset of state variables to be independent of player actions.

### 2.1.5 Period Return

The von Neumann-Morgenstern utility function for player $i$ at time $t$ is:

$$u_i\left(\mathbf{s}_t, \mathbf{a}_{it}, \mathbf{a}_{-it}\right) \equiv \pi_i\left(\mathbf{s}_t, \mathbf{a}_{it}, \mathbf{a}_{-it}\right) + \epsilon_{it}$$

where $\epsilon_{it}$ is a continuous random variable observed only by player $i$ at time $t$ and which has a density $f\left(\epsilon_{it}\right)$ and cumulative distribution function $F\left(\epsilon_{it}\right)$. The error $\epsilon_{it}$ can be interpreted as a preference shock unobserved by both the econometrician as well as by the other players and which makes player strategies as a function of the state random (see Section 2.1.6). It can also be interpreted as an unobserved state variable. See Rust (1987) for a discussion of this interpretation within the context of dynamic optimization problems. We make the following assumption on the distribution of $\epsilon_{it}$.

**Assumption (A2)**. The private shock $\epsilon_{it}$ is distributed *iid* across agents, actions, states, and time.

The term $\pi_i\left(\mathbf{s}_t, \mathbf{a}_{it}, \mathbf{a}_{-it}\right)$ is a scalar which is a function of the current state $\mathbf{s}_t$ and the action vector for players $i$ and $-i$, i.e. $\pi_i\left(\mathbf{s}_t, \mathbf{a}_{it}, \mathbf{a}_{-it}\right) : \mathcal{S}_t \times \mathcal{A}_{it} \times \mathcal{A}_{-it} \rightarrow \mathbb{R}$, where $\mathbb{R}$ is the set of real numbers. We assume payoffs are additively separable over time.

### 2.1.6 Strategies

We assume that players choose actions simultaneously at each time $t$. A strategy for agent $i$ is a vector-valued function $\mathbf{a}_{it} = \delta_i\left(\mathbf{s}_t, \epsilon_{it}\right)$, which maps the state at time $t$ and agent $i$'s time $t$ private shock to agent $i$'s time $t$ action vector $\mathbf{a}_{it}$. From the perspective of all other players $-i$, player $i$'s policy function as a function of the state can be represented by the conditional probability function $\sigma_i(\mathbf{A}_{it} = \mathbf{a}_{it}|\mathbf{s}_t)$, such that:

$$\sigma_i(\mathbf{A}_{it} = \mathbf{a}_{it}|\mathbf{s}_t) = \int \mathbb{I}\{\delta_i(\mathbf{s}_t, \epsilon_{it}) = \mathbf{a}_{it}\} \, dF(\epsilon_{it})$$

where $dF(\epsilon_{it}) \equiv f(\epsilon_{it})\, d\epsilon_{it}$ and where the notation $\mathbf{A}_{it}$ emphasizes that actions are random from the perspective of other players (we use the notation $A_{it}$ when actions are a random variable rather than a random vector). Abusing notation, we often abbreviate $\sigma_i(\mathbf{A}_{it} = \mathbf{a}_{it}|\mathbf{s}_t)$ as $\sigma_i$. Further define the product of the profile of policy functions for all players $-i$ at time $t$ as $\boldsymbol{\sigma}_{-i}(\mathbf{A}_{-it} = \mathbf{a}_{-it}|\mathbf{s}_t) \equiv \prod_{j\in -i} \sigma_j(\mathbf{A}_{jt} = \mathbf{a}_{jt}|\mathbf{s}_t)$, which we abbreviate as $\boldsymbol{\sigma}_{-i}(\mathbf{a}_{-it}|\mathbf{s}_t)$. Finally, we separately denote a potentially suboptimal policy function for player $i$ at time $t$ as $\overline{\sigma}_i$, which plays a special role in our policy improvement algorithm detailed in Section 2.2. For simplicity, we abstract away from the possibility that each player's policy function changes over time. It is straightforward to relax this simplification in what follows.

### 2.1.7 Value Function and Choice-Specific Value Function

**Value Function**. Let $\beta$ be a common discount factor. We define the following *ex ante* value function for player $i$ at time $t$,

$$V_i(\mathbf{s}_t, \epsilon_{it}) \equiv \max_{\mathbf{a}_{it}\in\mathcal{A}_{it}} \left\{ \sum_{\mathbf{a}_{-it}\in\mathcal{A}_{-it}} \left(\pi_i(\mathbf{s}_t, \mathbf{a}_{it}, \mathbf{a}_{-it}) + \epsilon_{it} + \beta \mathbb{E}_{\mathbf{S}_{t+1},\epsilon_{it+1}}\left[V_i(\mathbf{s}_{t+1}, \epsilon_{it+1})|\mathbf{s}_t, \mathbf{a}_{it}, \mathbf{a}_{-it}\right]\right) \boldsymbol{\sigma}_{-i}(\mathbf{a}_{-it}|\mathbf{s}_t) \right\} \tag{1}$$

where,

$$\mathbb{E}_{\mathbf{S}_{t+1},\epsilon_{it+1}}\left[V_i(\mathbf{s}_{t+1}, \epsilon_{it+1})|\mathbf{s}_t, \mathbf{a}_{it}, \mathbf{a}_{-it}\right] = \int_{\mathbf{s}_{t+1}\in\mathcal{S}_{t+1}} \int_{\epsilon_{it+1}} V_i(\mathbf{s}_{t+1}, \epsilon_{it+1}) dF(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_{it}, \mathbf{a}_{-it})\, dF(\epsilon_{it+1})$$

where it is assumed agent $i$ makes the maximizing choice $\mathbf{a}_{it}$ in each period $t = 1, ..., T$ and that the value function is implicitly indexed by the profile of policy functions for all agents. We allow opponent strategies to be optimal or suboptimal, which facilitates the use of our method in practical game settings. The expectation $\mathbb{E}_{\mathbf{S}_{t+1},\epsilon_{it+1}}$ is taken over all realizations of the state $\mathbf{S}_{t+1}$, conditional on the current state and actions, and the unobserved private shock for agent $i$ in period $t + 1$.

**Choice-Specific Value Function**. We also define the following *ex ante* choice-specific value function for player $i$:

$$V_i\left(\mathbf{s}_t, \epsilon_{it}; \mathbf{a}_{it}, \overline{\sigma}_i\right) \equiv \sum_{\mathbf{a}_{-it}\in\mathcal{A}_{-it}} \left(\pi_i\left(\mathbf{s}_t, \mathbf{a}_{it}, \mathbf{a}_{-it}\right) + \epsilon_{it} + \beta\mathbb{E}_{\mathbf{s}_{t+1},\epsilon_{it+1}}\left[V_i(\mathbf{s}_{t+1}, \epsilon_{it+1}; \overline{\sigma}_i)|\mathbf{s}_t, \mathbf{a}_{it}, \mathbf{a}_{-it}\right]\right)\boldsymbol{\sigma}_{-i}\left(\mathbf{a}_{-it}|\mathbf{s}_t\right) \quad (2)$$

where,

$$\mathbb{E}_{\mathbf{s}_{t+1},\epsilon_{it+1}}\left[V_i(\mathbf{s}_{t+1}, \epsilon_{it+1}; \overline{\sigma}_i)|\mathbf{s}_t, \mathbf{a}_{it}, \mathbf{a}_{-it}\right] = \int_{\mathbf{s}_{t+1}\in\mathcal{S}_{t+1}} \int_{\epsilon_{it+1}} V_i(\mathbf{s}_{t+1}, \epsilon_{it+1}; \overline{\sigma}_i) dF\left(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_{it}, \mathbf{a}_{-it}\right) dF\left(\epsilon_{it+1}\right)$$

Our choice-specific value function represents the value of a particular action choice $\mathbf{a}_{it}$, conditional on the state $\mathbf{s}_t$, the agent's private shock, $\epsilon_{it}$, and a potentially suboptimal strategy played by agent $i$ beyond the current time period, $\overline{\sigma}_i$. Both value functions we define abstract away from the possibility that the value function changes over time. This simplification is not necessary for implementing our method and can be relaxed if researchers have access to enough data to efficiently estimate separate choice-specific value functions per time period.

## 2.2 Policy Function Improvement

In this section, we outline our algorithm for generating a one-step improvement policy as well as our Machine Learning estimator of choice.

**Algorithm 1** *We generate a one-step improvement policy according to the following steps:*

1. *Estimate the strategies of competitors, and, if necessary, the law of motion and the payoff function, using a Machine Learning estimator to select a parsimonious subset of state variables.*

2. *Fix an initial strategy for the agent.*

3. *Use the estimated competitor strategies, payoff function, law of motion, and fixed agent strategy to simulate play.*

4. *Use this simulated data to estimate the choice-specific value function.*

5. *Obtain a one-step improvement policy.*

6. *Repeat by returning to step 1 and using the one-step improvement policy as the fixed initial agent strategy.*

Since we seek to improve a strategy of a reference agent $i$, we assume the researcher knows $F(\epsilon_{it+1})$ for this agent. In particular, for expositional clarity, we set $\epsilon_{it} = 0$ for all $t = 1, ..., T$ for agent $i$. Prior to describing the steps of Algorithm 1 in detail, we describe our Machine Learning estimator.

### 2.2.1 Component-Wise Gradient Boosting

We use Component-Wise Gradient Boosting (CWGB) in Algorithm 1, Step 1, to estimate models corresponding to opponent strategies, and, if necessary, the law of motion and payoff function for a reference agent $i$. CWGB is a specific variant of boosting methods, which are a popular class of Machine Learning methods that accommodate the estimation of both linear and nonlinear models. Boosting methods work by sequentially estimating a series of simple models, deemed "base learners," and then forming a "committee" of predictions from these models through weighted averaging. See Hastie *et al.* (2009) for a survey of boosting methods.

We present the linear variant of CWGB we employ and then briefly discuss how this setup can be generalized to nonlinear contexts. To facilitate the description of CWGB, we show how it can be used to estimate the opponents' strategy functions, i.e. $\sigma_j$ for $j \in -i$, which are estimations employed in Algorithm 1, Step 1. We assume researchers have access to a random sample of previous plays of the game for each player $j \in -i$, i.e. $\{a_{jlt}, \mathbf{s}_{lt}\}_{l=1,t=1}^{L,T}$, where the subscript $l = 1, ..., L$ indexes individual observations of play attributable to player $j$. For the purposes of this description, we assume the support of $a_{jt}$ is binary, 0, 1, which means that the linear CWGB estimator we employ effectively estimates a linear probability model of the probability of choice $a_{jt}$ conditional on the dimension-reduced state vector $\widetilde{\mathbf{s}}_t$.[11] The estimator works according to the following steps.

**Algorithm 2** ***CWGB Estimator (Linear)***

1. *Initialize the iteration* 0 *model, denoted as* $\widehat{\sigma}_j^{c=0}$, *by setting* $\widehat{\sigma}_j^{c=0} = \frac{1}{LT}\sum_{l=1}^{L}\sum_{t=1}^{T} a_{jlt}$, *i.e. initializing the model with the empirical mean of the outcome variable.*[12]

2. *In the first step, estimate* $K_s$ *univariate linear regression models (without intercepts) of the relationship between* $a_{jt}$ *and each* $s_{kt}$ *as the sole regressor, denoted as* $\widehat{b}(s_{1t}) = \widehat{\beta}_{s_{1t}} s_{1t}, ..., \widehat{b}(s_{K_s t}) = \widehat{\beta}_{s_{K_s t}} s_{K_s t}$ *where each* $b(.)$ *is a linear base learner and each* $\widehat{\beta}_{s_{kt}}$ *is a univariate linear regression parameter.*[13]

[11] Section 2.2.2 discusses the case where the support of $a_{jt}$ is not binary.

[12] We abuse notation slightly here, since in principle, $L$ can vary by time period.

[13] These linear regression models could also be estimated with an intercept term, which would vary for each of the $K_s$ models.

3. *Choose the model with the best OLS fit, denoted as* $\widehat{b}_{W1}(s_{W1t})$ *for some* $W1 \in \{1, ..., K_s\}$*. Update the iteration 1 model as* $\widehat{\sigma}_j(A_{jt} = a_{jt}|s_{W1t})^{c=1} = \widehat{\sigma}_j^{c=0} + \widehat{b}_{W1}(s_{W1t})$ *and use it to calculate the iteration 1 fitted residuals.*

4. *Using the iteration 1 fitted residuals as the new outcome variable, estimate an individual univariate linear regression model (without an intercept) for each individual regressor* $s_{kt}$ *as in iteration 1. Choose the model with the best OLS fit, denoted as* $\widehat{b}_{W2}(s_{W2t})$ *for some* $W2 \in \{1, ..., K_s\}$*. Update the iteration 2 model as:*

$$\widehat{\sigma}_j(A_{jt} = a_{jt}|s_{W1t}, s_{W2t})^{c=2} = \widehat{\sigma}_j(A_{jt} = a_{jt}|s_{W1t})^{c=1} + \lambda\widehat{b}_{W2}(s_{W2t})$$

   *where* $\lambda$ *is called the "step-length factor," which is often chosen using k-fold cross-validation (we set* $\lambda = 0.01$*). Use* $\widehat{\sigma}_j(A_{jt} = a_{jt}|s_{W1t}, s_{W2t})^{c=2}$ *to calculate iteration 2 residuals.*

5. *Continue in a similar manner for a fixed number of iterations to obtain the final model (we use* $C = 1000$ *iterations). The number of iterations is often chosen using k-fold cross-validation.*

As a consequence of this estimation process, it is usually the case that some regressors never comprise the best fit model in any iteration. If so, then this variable is excluded from the final model, yielding the dimension-reduced state vector $\widetilde{\mathbf{s}}_t$ defined in section 2.1.2 and the estimated opponent strategy models $\widehat{\sigma}_j(A_{jt} = a_{jt}|\widetilde{\mathbf{s}}_t)$ for each $a_{jt} \in \mathcal{A}_{jt}$ and $j \in -i$. CWGB estimates are easily computed using one of several available open-source packages, including $H_2O$ as well as mboost and gbm in R.[14] For the linear variant of CWGB, we use the glmboost function available in the mboost package of R. See Hofner *et al.* (2014) for an introduction to implementing CWGB in R using the mboost package.

The total number of iterations $C$ and the step length factor $\lambda$ are tuning parameters for the algorithm, typically chosen using k-fold cross-validation. Cross-validation is a subset of the out-of-sample testing that is used as the primary criteria for judging the performance of Machine Learning models in practice. Out-of-sample testing involves the creation of a training dataset, which is used to estimate the models of interest, and a testing dataset (a "holdout" sample), which is used to evaluate the performance of these estimators. The separation of training and testing datasets is important for evaluating estimators, since in general, adding regressors to a model often reduces training sample prediction error but does not necessarily improve out-of-sample prediction error. A common criteria for evaluating estimator performance on the testing dataset is the Mean-Squared Error (MSE) criteria. Training and testing models is feasible in settings where the number of observations is large, since this allows both datasets to have a sufficient number of observations to

[14] $H_2O$ is an open source software developed for implementing Machine Learning methods on particularly large datasets and is available from http://0xdata.com/. The documentation for the gbm and mboost packages in R, respectively, are available from http://cran.r-project.org/web/packages/gbm/index.html and http://cran.r-project.org/web/packages/mboost/index.html.

generate precise estimates. See Hastie *et al.* (2009) for an introduction to the common practice of training and testing in Machine Learning.

Of $C$ and $\lambda$, the number of iterations ($C$) has proven to be the most important tuning parameter in CWGB models, and the most useful practical criterion for choosing $\lambda$ is that it should be small (e.g., $\lambda = 0.01$ or $\lambda = 0.1$, see Hofner *et al.* (2014) and Schmid and Hothorn (2008)). On the one hand, choosing a $C$ that is too large may result in overfitting, i.e. low MSE in sample, but poor MSE out-of-sample. On the other hand, choosing a $C$ that is too low also results in poor out-of-sample performance. As a consequence, $C$ is often chosen by minimizing cross-validation error on randomly chosen holdout samples. For example, when performing 10-fold cross validation for a given value of $C$ in this context, the researcher randomly chooses 10% of the observations to include in a holdout sample. Then Algorithm 2 is run on the remaining 90% of the data, i.e. the training sample, to obtain the estimated model, which used to compute the MSE on the 10% testing sample. This process is repeated nine additional times using the same value of $C$, each with a different randomly chosen holdout and training sample, and the total MSE across all 10 folds is recorded. A 10-fold cross-validation procedure is carried out for every candidate value of $C$, and the value of $C$ that generates the lowest total MSE is chosen. More generally, K-fold cross-validation generates $K$ testing samples. The mboost package provides a simple command for implementing K-fold cross-validation automatically.[15]

Generalizations of CWGB are achieved primarily through the choice of alternative base learners $b(.)$, subsets of regressors included in each base learner model, and loss functions. For example, nonlinear versions of gradient boosting might employ regression trees instead of linear $b(.)$, or they might use subsets of regressors larger than one as part of the base learning models to accommodate interactions among regressors. We direct readers interested in a more comprehensive introduction to boosting methods to Hastie *et al.* (2009) and Hofner *et al.* (2014).

### 2.2.2 Opponent Strategies, Period Return, and the Law of Motion (Step 1)

The first step of Algorithm 1 involves estimating opponent strategy functions, and if needed, the payoff function for agent $i$ and the law of motion. To do so, we make the following assumption.

**Assumption (A3)**. Researchers have access to *iid* random samples of the form **(i)** $\{a_{jlt}, \mathbf{s}_{lt}\}_{l=1,t=1}^{L,T}$ for each $j \in -i$, **(ii)** $\{\pi_i(\mathbf{s}_{lt}, \mathbf{a}_{lt}), \mathbf{s}_{lt}, \mathbf{a}_{lt}\}_{l=1,t=1}^{L,T}$, and **(iii)** $\{\widetilde{\mathbf{s}}_{lt+1}, \widetilde{\mathbf{s}}_{lt}, \widetilde{\mathbf{a}}_{lt}\}_{l=1,t=1}^{L,T}$.

We invoke (A3)(i) to estimate a separate strategy function model for each $j \in -i$, with each model denoted as $\widehat{\sigma}_j(A_{jt} = a_{jt}|\widetilde{\mathbf{s}}_t)$.[16] As a prerequisite to estimation, we assume the action space can be redefined in a way that makes it low-dimensional, as described in Section 2.1.3. In our

[15]See Hofner *et al.* (2014) for details on selecting $C$ using cross-validation.

[16]If feasible, in some contexts it may be desirable to estimate separate strategy function models for each feasible action, i.e. $\widehat{\sigma}_j(A_{jt} = a_{jt}|\widetilde{\mathbf{s}}_t)$ for $a_{jt} \in \mathcal{A}_{jt}$. We employ this estimation strategy in our entry game application, described in Section 3.

application described in Section 3, we assume the support of the action is binary 0,1, noting that there are more general forms of boosting estimators capable of classification in the case of discrete categorical variables with more than two choices.[17] For the binary case, we propose estimating a linear probability model using CWGB as demonstrated in Algorithm 2. We abuse the notation of $\widetilde{\mathbf{s}}_t$, since in practice, the state variables included in $\widetilde{\mathbf{s}}_t$ may vary across models.

Often, the payoff function for agent $i$ may be known. However, in many settings, it may be desirable and feasible to estimate these payoff functions. Under (A3)(ii), we assume researchers have access to a random sample of scalar payoffs for agent $i$ along with corresponding states and actions. We propose estimating the payoff function using CWGB and denote this estimate as $\widehat{\pi}_i\left(\widetilde{\mathbf{s}}_t, \widetilde{\mathbf{a}}_t\right)$, where again, $\widetilde{\mathbf{s}}_t$ and $\widetilde{\mathbf{a}}_t$ represent the dimension-reduced state and action vectors selected by CWGB, keeping in mind that the selected state variables may be different from those selected by CWGB to produce the opponent strategy function estimates.

Under some circumstances, such as in the entry game application we study in Section 3, the law of motion is deterministic and need not be estimated. In settings where the law of motion must be estimated, the outcomes $(\mathbf{s}_{t+1})$ will be high-dimensional, making the estimation of the law of motion infeasible or at least computationally burdensome. We therefore propose estimating only the evolution of the state variables collected across all dimension-reduced states selected by the CWGB estimation processes for all opponent strategy functions and the payoff function. We abuse the notation of this state vector by also denoting it as $\widetilde{\mathbf{s}}_t = (s_{1t}, ..., s_{Mt})$, where $M$ is the total number of state variables retained across all CWGB-estimated opponent strategy and payoff function models. This restricts attention only to those state variables selected under the CWGB selection criteria for all other estimands of interest, rather than the state variables that comprise the original state vector $\mathbf{s}_t$. If the action vector is also high-dimensional, we use the dimension-reduced action vector $\widetilde{\mathbf{a}}_t$ selected in the payoff function estimation process. Using (A3)(iii) we assume researchers have access to a random sample of these state and action variables. Estimation of the law of motion using the retained state and action variables can proceed flexibly and the exact estimator used depends on the application and on the nature of the state variables. We propose estimating a separate model for each outcome state variable included in $\widetilde{\mathbf{s}}_{t+1}$. These estimated models are denoted as $\widehat{f}_k\left(\widetilde{\mathbf{s}}_t, \widetilde{\mathbf{a}}_t\right)$ for each $k = 1, ..., M$. If $s_{kt+1}$ is continuous, $\widehat{f}_k\left(\widetilde{\mathbf{s}}_t, \widetilde{\mathbf{a}}_t\right)$ can be estimated using linear regressions, which generates models that takes the form $s_{kt+1} = \widehat{f}_k\left(\widetilde{\mathbf{s}}_t, \widetilde{\mathbf{a}}_t\right) + \widehat{m}_t$ for $k = 1, ..., M$, where $\widehat{m}_t$ is a residual. If $s_{kt+1}$ is discrete and it's support is binary $0, 1$, then a parametric or semiparametric estimator of the probability of this state transition as a function of $\widetilde{\mathbf{s}}_t$ and $\widetilde{\mathbf{a}}_t$ can be used (for example, a probit model or again, OLS). If $s_{kt+1}$ is discrete and its support is categorical, then an estimator for categorical variables can be used (for example, the multinomial logit).

[17]This includes, e.g., the recently proposed gradient boosted feature selection algorithm of Zheng *et al.* (2014). We note that the implementation of this algorithm requires large datasets, i.e. those where the number of observations is much larger than the number of regressors.

### 2.2.3 Initial Strategy for Agent (Step 2)

The second step involves fixing an initial strategy for the agent, i.e., choosing the potentially suboptimal policy function for player $i$, i.e. $\overline{\sigma}_i$. In practice, this policy function can be any optimal or suboptimal policy, including previously proposed strategy that represents the highest payoffs researchers or game players have been able to find in practice. For example, if we were studying a game such as multi-player Texas Hold'em, we might start with the counterfactual regret minimization strategy recently proposed by Bowling *et al.* (2015). Regardless of the choice of $\overline{\sigma}_i$, Algorithm 1 is designed to weakly improve upon this strategy. However, a particularly suboptimal choice for $\overline{\sigma}_i$ may slow the convergence of our one-step improvements in subsequent iterations.

### 2.2.4 Simulating Play (Step 3)

We simulate play for the game using $\widehat{\sigma}_j\left(A_{jt} = a_{jt}|\widetilde{\mathbf{s}}_t\right)$ for all $j \in -i$, the law of motion, the payoff function, and $\overline{\sigma}_i$. We describe the case where both the law of motion and period return functions are estimated since it is straightforward to implement what follows when these functions are known and deterministic. Our simulation focuses only on the state variables selected across all CWGB-estimated models for opponent strategies and the period return function, i.e. $\widetilde{\mathbf{s}}_t$ as introduced in Section 2.2.2. Denote simulated variables with the superscript $*$. First, we generate an initial state $\widetilde{\mathbf{s}}_1^*$. This can be done by either randomly selecting a state from the support of $\widetilde{\mathbf{s}}_1$ or by restricting attention to particular "focal" states. For example, in an industry game, focal states of interest might include the current state of the industry or states likely to arise under certain policy proposals. We then generate period $t = 1$ actions. For competitors, we choose actions $\mathbf{a}_{-i1}^*$ by drawing upon the estimated probability models generated by opponent strategies $\widehat{\sigma}_j\left(A_{jt} = a_{jt}|\widetilde{\mathbf{s}}_t\right)$ for each $j \in -i$. For the agent, we choose actions $a_{i1}^*$ by using the fixed agent strategy $\overline{\sigma}_i$ generally while randomly permuting a deviation to this action in some simulation runs or time periods. Given choices for $a_{i1}^*$ and $\mathbf{a}_{-i1}^*$, and also given $\mathbf{s}_1^*$, we draw upon the estimated law of motion models $\widehat{f}_k\left(\widetilde{\mathbf{s}}_1^*, \widetilde{\mathbf{a}}_1^*\right)$ for $k = 1, ..., M$ to generate $\mathbf{s}_2^*$. We draw $\widetilde{s}_{k2}^*$ from these models in two ways, depending upon whether $\widetilde{s}_{k2}$ is discrete or continuous. For discrete state variables, $\widehat{f}_k\left(.\right)$ is a probability distribution, and we draw upon this probability distribution to choose $\widetilde{s}_{k2}^*$. For continuous state variables, $\widehat{f}_k\left(.\right)$ is a linear regression model. We use this linear regression model to generate a prediction for the next period state variable, which represents it's mean value. We then draw upon the empirical distribution of estimated residuals generated by our original sample (see Section 2.2.2) to select a residual to add to the model prediction. This gives $\widetilde{s}_{k2}^* = \widehat{f}_k\left(\widetilde{\mathbf{s}}_t^*, \widetilde{\mathbf{a}}_t^*\right) + \widehat{m}_t^*$. We choose each $a_{it}^*$, $\mathbf{a}_{-it}^*$, and $\widetilde{\mathbf{s}}_{t+1}^*$ for $t = 2, ..., T$ similarly by randomly deviating from $\overline{\sigma}_i$, and also by drawing upon $\widehat{\sigma}_j\left(.\right)$ and $\widehat{f}\left(.\right)$, respectively. This simulation sequence produces data used to estimate the choice-specific value of a one-period deviation from $\overline{\sigma}_i$. In all time periods, we compute payoffs and generate the simulated sums for each $t = 1, ..., T$:

$$V_i\left(\widetilde{\mathbf{s}}_{lt}^{*}; a_{ilt}^{*}; \overline{\sigma}_i\right) = \sum_{\tau=t}^{T} (\beta)^{\tau-t}\, \pi_i\left(\widetilde{\mathbf{s}}_{l\tau}^{*}, \widetilde{\mathbf{a}}_{l\tau}^{*}\right) \tag{3}$$

The simulated sums represent the discounted payoffs of choice $a_{ilt}^{*}$, given that the agent plays $\overline{\sigma}_i$, each opponent $j \in -i$ plays according to $\widehat{\sigma}_j(.)$, and the law of motion evolves as dictated by $\widehat{f}_k(.)$ for $k = 1, ..., M$. These simulated sums provide us with outcome variables for estimation of the choice-specific value functions in Step 4.

The low-dimensionality of each $\widetilde{\mathbf{s}}_t$ greatly reduces the simulation burden in two ways. First, the simulation only needs to reach points in the support of each $\widetilde{\mathbf{s}}_t$, rather than all points in the full support of $\mathbf{s}_t$, which is computationally infeasible. Second, reducing the number of regressors may lead to more reliable estimates of $\widehat{\sigma}_j(.)$, and $\widehat{f}_k(.)$ due to a variety of potential statistical issues encountered in settings where the number of regressors is large. When the number of regressors is large, researchers often find in practice that many of these regressors are highly multicollinear, and in the context of collinear regressors, out-of-sample prediction is often maximized using a relatively small number of regressors. A large number of regressors may also cause identification issues using conventional models. Good estimates of these models lead to better predictions, which in turn allow the simulation to reliably search across the space of state variables that are strategically likely to arise when forming data for the choice-specific value function estimates. This in turn generates more reliable estimates of the choice-specific value functions, which leads to better improvements in Step 5.

The simulation process provides us with a sample of simulated data of the form:

$$\left\{V_i\left(\widetilde{\mathbf{s}}_{lt}^{*}, a_{ilt}^{*}; \overline{\sigma}_i\right), \widetilde{\mathbf{s}}_{lt}^{*}, a_{ilt}^{*}\right\}_{l=1,t=1}^{L,T}$$

for player $i$. We use this simulated data to estimate the choice-specific value function for agent $i$ in the next step.

### 2.2.5 Estimating Choice-Specific Value Function (Step 4)

If there is smoothness in the value function, this allows us to pool information from across our simulations in Step 3 to reduce variance in our estimator. Note that the simulated choice specific value function will be equal to the choice specific value function plus a random error due to simulation. If we have an unbiased estimator, adding error to the dependent variable of a regression does not result in a biased estimator.

We propose pooling the simulated data $\left\{V_i\left(\widetilde{\mathbf{s}}_{lt}^{*}, a_{ilt}^{*}; \overline{\sigma}_i\right), \widetilde{\mathbf{s}}_{lt}^{*}, a_{ilt}^{*}\right\}_{l=1,t=1}^{L,T}$ over time and estimating separate choice-specific value functions for each action $a_{it} \in A_{it}$ using linear regressions (with intercepts), where each $V_i\left(\widetilde{\mathbf{s}}_{lt}^{*}, a_{ilt}^{*}; \overline{\sigma}_i\right)$ is the outcome variable and $\widetilde{\mathbf{s}}_{lt}^{*}$ are the regressors. We denote each estimated model as $\widehat{V}_i\left(\widetilde{\mathbf{s}}_t, a_{it}; \overline{\sigma}_i\right)$.

### 2.2.6 One-Step Improvement Policy (Step 5)

We generate a one-step improvement policy for player $i$, denoted as $\widehat{\sigma}_i^1$, which represents policy function which maximizes the estimated choice-specific value function in the corresponding period $t$ conditional on $\overline{\sigma}_i$, i.e. we seek the policy function vector $\widehat{\sigma}_i^1$[18] such that, for all $t = 1, ..., T-1$:

$$\widehat{\sigma}_i^1 \equiv \left\{ \sigma_i : \widetilde{S}_t \rightarrow \mathcal{A}_{it} \,\middle|\, \begin{array}{c} \sigma_i = \arg\max_{a_{it} \in \mathcal{A}_{it}} \left\{ \widehat{V}_i \left( \widetilde{\mathbf{s}}_t, a_{it}; \overline{\sigma}_i \right) \right\} \\ \text{for all } \widetilde{\mathbf{s}}_t \in \mathcal{S}_t \end{array} \right\} \tag{4}$$

Each $\widehat{\sigma}_i^1$ is "greedy" in that it searches only for the action choices that maximize the estimated choice-specific value function in the current period conditional on the agent's fixed strategy $\overline{\sigma}_i$, rather than the actions that maximize the value of choices across all time periods. Once $\widehat{\sigma}_i^1$ is obtained, this strategy vector can be used to generate $\overline{\sigma}_i$ in the following iteration, repeating Algorithm 1 again to obtain a second-step improvement $\widehat{\sigma}_i^2$, and so forth until a suitable stopping rule is met.

## 3 Empirical Illustrations

### 3.1 Application In Progress (Texas Hold'Em)

In an application in progress, we apply our algorithm to the popular poker game Texas Hold'em, which is a game of imperfect information with a state space cardinality of $3.16 \times 10^{17}$. Bowling *et al.* (2015) developed an iterative counterfactual regret minimization (CFR) algorithm, which is the best performing strategy developed for Texas Hold'em to date. We use the CFR strategy as an input to our method, inserting it as the initial fixed strategy for our reference agent and also as the strategy for opponents. Using these inputs, we attempt to derive a policy that improves-upon the CFR strategy.

### 3.2 Chain Store Entry Game Institutional Background and Data

According to the U.S. Census, U.S. retail sales in 2012 totaled \$4.87 trillion, representing 30 percent of nominal U.S. GDP. The largest retailer, Wal-Mart, dominates retail trade, with sales accounting for 7 percent of the U.S. total in 2012.[19] Wal-Mart is not only the largest global retailer, it is also the largest company by total revenues of any kind in the world.[20] Notwithstanding their importance in the global economy, there has been a relative scarcity of papers in the literature

[18] As with the other policy functions, we abuse notation by suppressing the dependence of $\widehat{\sigma}_i^1$ on the corresponding states.

[19] Total U.S. retail sales collected from the Annual Retail Trade Survey (1992-2012), available: http://www.census.gov/retail/. Wal-Mart share of retail sales collected from the National Retail Federation, Top 100 Retailers (2013), available: https://nrf.com/resources/top-retailers-list/top-100-retailers-2013.

[20] Fortune Global 500 list (2014), available: http://fortune.com/global500/.

studying chain store retailers in a way that explicitly models the multi-store dimension of chain store networks, primarily due to modeling difficulties.[21]

Wal-Mart, along with other large chain store retailers such as Target, Costco or Kmart, operate large networks of physical store and distribution center locations around the world and compete in several product lines, including general merchandise and groceries, and via several store types, including regular stores, supercenters, and discount warehouse club stores. For example, by the end of 2014, Wal-Mart had 42 distribution centers and 4203 stores in the U.S., with each distribution center supporting from 90 to 100 stores within a 200-mile radius.[22]

In our illustration, we model a game similar to the one considered by Holmes (2011), who studies the physical store location decisions of Wal-Mart. Our game consists of two competing chain store retailers which seek to open a network of stores and distribution centers from the years $t = 2000, ..., 2006$ across a finite set of possible physical locations in the United States.[23] One location corresponds to a metropolitan statistical area (MSA) as defined by the U.S. Census Bureau and is indexed by $l = 1, ..., L$ with support $\mathcal{L}$ and $L = 227$ possible locations.[24] We extend the game in Holmes (2011) by modeling the decision of where to locate distribution centers as well as stores. Each firm sells both food and general merchandise and can open two types of distribution centers–food and general merchandise–and two types of stores–supercenters and regular stores. Supercenters sell both food and general merchandise and are supplied by both types of distribution centers, while regular stores sell only general merchandise and are supplied only by general merchandise distribution centers.[25]

At a given time period $t$, each firm $i$ will have stores and distribution centers in a subset of locations, observes the facility network of the competitor as well as the current population of each MSA, and decides in which locations to open new distribution centers and stores in period $t + 1$. We collect MSA population and population density data from the US Census Bureau.[26] As in Holmes (2011), we focus on location decisions and abstract away from the decision of how many

[21] For recent exceptions, see Aguirregabiria and Vicentini (2014), Holmes (2011), Jia (2008), Ellickson, Houghton, and Timmins (2013), and Nishida (2014).

[22] The total number of stores figure excludes Wal-Mart's 632 Sam's Club discount warehouse club stores.

[23] Throughout the paper, we use the notation $t = 2000, ..., 2006$ and $t = 1, ..., T$ with $T = 7$ interchangeably.

[24] Census Bureau, County Business Patterns, Metropolitan Statistical Areas, 1998 to 2012. Available at: http://www.census.gov/econ/cbp/. All raw data used in this paper, which includes a list of MSA's used, is available from: http://abv8.me/4bL.

[25] Additionally, each firm operates import distribution centers located around the country, where each import distribution center supplies both food and general merchandise distribution centers. We abstract away from decisions regarding import distribution center placement, fixing and making identical the number and location of import distribution centers for both firms. Specifically, we place import distribution centers for each competitor in the locations in our sample closest to the actual import distribution center locations of Wal-Mart during the same time period. See the Appendix for details.

[26] Our population density measure is constructed using MSA population divided by MSA land area by square miles in 2010, both collected from the U.S. Census Bureau. Population data by MSA was obtained from the Metropolitan Population Statistics, available: http://www.census.gov/population/metro/data/index.html. Land area in square miles by MSA in 2010 was obtained from the Patterns of Metropolitan and Micropolitan Population Change: 2000 to 2010, available: http://www.census.gov/population/metro/data/pop_data.html.

facilities to open in each period. Instead, we constrain each competitor to open the same number of distribution centers of each category actually opened by Wal-Mart annually in the United States from 2000 to 2006, with the exact locations and opening dates collected from data made publicly available by an operational logistics consulting firm.[27] We also constrain each competitor to open two supercenters for each newly opened food distribution center and two regular stores for each newly opened general merchandise distribution center.[28] Finally, we use the distribution center data to endow our competitor with a location strategy meant to approximate Wal-Mart's actual expansion patterns as documented by Holmes (2011), which involved opening a store in a relatively central location in the U.S., opening additional stores in a pattern that radiated from this central location out, and never placing a store in a far-off location and filling the gap in between. This pattern is illustrated in Figure 1.[29]

## 3.3 Chain Store Entry Game Model

In this section, we adapt our theoretical game model developed in Section 2.1 to the chain store entry game characterized in Section 3.2.

**State.** Our state vector is comprised of indicator variables over facility placement decisions by firm $i$ and firm $-i$ across all locations, as well as a location-specific characteristic (population), resulting in $K_s = 8L + 1 = 1817$ variables. For example, the first $L$ indicator variables take a value of 1 if firm $i$ has placed a general merchandise distribution center in location $l$, 0 otherwise. The next $L$ indicators work similarly with respect to firm $i$ food distribution centers, and so forth for firm $i$ regular stores and supercenters. Similarly, the final $4L$ indicators represent facility placements by firm $-i$. Finally, we include a discrete variable representing the population for a given location $l$.

**Actions.** As introduced in Section 3.2, we force each competitor to open a pre-specified aggregate number of distribution centers and stores in each period.[30] The set of feasible locations is constrained by the period $t$ state, since for a given facility type $q$, firm $i$ can open at most one facility per location.[31] Further, we restrict firms to open at most one own store of any kind in each MSA, with firms each choosing regular stores prior to supercenters in period $t$. Given these constraints and also given the designated number of aggregate facility openings in each period, at time

[27] This included thirty food distribution centers and fifteen general merchandise distribution centers. Wal-Mart distribution center locations with opening dates were obtained from MWPVL International, available: http://www.mwpvl.com/html/walmart.html. We also provide a list in the Appendix.

[28] This results in a total of sixty supercenters and thirty regular stores opened over the course of the game by each firm.

[29] Data prepared by Holmes (2011), available: http://www.econ.umn.edu/~holmes/data/WalMart/.

[30] We force firms to open the following number of facilities in each period (food distribution centers, general merchandise distribution centers, regular stores, and supercenters): $(4, 4, 8, 8)$ in $t = 2000$, $(5, 2, 4, 10)$ in $t = 2001$, $(6, 1, 2, 12)$ in $t = 2002$, $(2, 3, 6, 4)$ in $t = 2003$, $(3, 3, 6, 6)$ in $t = 2004$, and $(3, 1, 2, 6)$ in $t = 2005$. Note that these vectors each represent facility openings for the next period, e.g. $(4, 4, 8, 8)$ in $t = 2000$ designates the number of openings to be realized in $t = 2001$.

[31] See the Appendix for details.

$t$, firm $i$ chooses a vector of feasible actions (locations) $\mathbf{a}_{it}$ with $K_a = 4L = 908$. This action vector is comprised of indicator variables over facility placement choices by firm $i$ across all locations. For example, the first $L$ indicator variables take a value of 1 if firm $i$ chooses to place a general merchandise distribution center in location $l$ at time $t+1$, 0 otherwise. Similarly, the remaining $3L$ indicator variables represent placement decisions for food distribution centers, regular stores, and supercenters, respectively. We assume that once opened, distribution centers and stores are never closed. As documented by Holmes (2011), Wal-Mart rarely closes stores and distribution centers once opened, making this assumption a reasonable approximation for large chain store retailers.[32]

**Law of Motion.** Since we assume the state is comprised of only the current network of facilities and populations, rather than their entire history, this game is Markov. Since we assume that all players have perfect foresight with respect to MSA-level population, the law of motion is a deterministic mapping from the current state and the current actions taken by players $i$ and $-i$ to the state in period $t+1$.

**Strategies.** The policy function for each agent maps the current state to location choices in the following period. The probabilities induced by the strategy of opponent $-i$ are also defined as before, since our agent $i$ does not observe the period $t$ location choices of opponent player $-i$ until time period $t+1$.[33]

**Period Return.** The period $t$ payoffs for firm $i$ represents operating profits for location $l$. In this game, operating profits are parametric and deterministic functions of the current location-specific state and are similar to the operating profits specified by Holmes (2011). Since customers substitute demand among nearby stores, operating profits in a given location are a function of both own and competitor facility presence in nearby locations. They are also a function of location-specific variable costs, distribution costs, population, and population density. For simplicity of exposition, we ignore the separate contribution of population density in the period return when defining the state and instead use population as the lone non-indicator location-specific characteristic of interest. The Appendix provides the details of our profit specification.

**Choice-Specific Value Function.** The choice-specific value function for agent $i$ in this game is a "local" facility and choice-specific value function, which is defined as the period $t$ location-specific discounted expected operating profits of opening facility $q \in \{f, g, r, sc\}$ in location $l$ for firm $i$, where $f$ represents food distribution centers, $g$ represents general merchandise distribution centers,

[32] Also see MWPVL International's list of WalMart distribution center openings and closing for additional support for this assertion, available from: http://www.mwpvl.com/html/walmart.html.

[33] Since in our illustration our state is "location-specific" in that it includes only the population of a particular location $l$ (rather than the vector of populations across all locations), we ignore the effect of populations across locations on opponent strategies. Although this is likely a misspecification, we define the state in this way to take advantage of cross-sectional differences in location populations when estimating the choice-specific value function, rather than relying only on across-time variation. We show in our Results section that our state is well-approximated by our specification. In practice, researchers with access to large datasets might include the entire vector of populations or other location-specific characteristics in the state.

$r$ represents regular stores, and $sc$ represents supercenters. We denote this facility-specific choice-specific value function as $V_i\left(\mathbf{s}_t, a^q_{ilt}; \overline{\sigma}_i\right)$, where $a^q_{ilt}$ replaces the action vector $\mathbf{a}_{it}$ and represents the decision by firm $i$ of whether to locate facility $q$ in location $l$, 0 otherwise. We focus on facility and location-specific value functions in order to take advantage of cross-sectional differences in value when estimating the choice-specific value function in the next section. The choice-specific value function is also conditional on a set of profit parameters, which is a dependence we suppress to simplify the notation. Details regarding all parameters are presented in the Appendix.[34]

## 3.4 Chain Store Entry Game Policy Function Improvement

We adapt our algorithm to derive a one-step improvement policy over a benchmark strategy in our chain store entry game.

**Opponent Strategies and the Law of Motion (Step 1).** In our illustration, we do not estimate models corresponding to opponent strategies. Instead, we force the competitor to open distribution centers in the exact locations and at the exact times chosen by Wal-Mart from 2000 to 2006, placing stores in the MSA's closest to these distribution centers. Specifically, for $\widehat{\sigma}_{-i}$ for all simulations, we force our competitor to place food and general merchandise distribution centers in the MSA's in our sample closest to the exact locations of newly opened Wal-Mart distribution centers of each kind during the years 2000 to 2006, as detailed in Tables 4 and 5. We then open regular stores in the two closest feasible MSA's to each newly opened firm $i$ general merchandise distribution center. After making this decision, we determine the closest firm $i$ general merchandise distribution center to each newly opened firm $i$ food distribution center and open supercenters in the two feasible MSA's closest to the centroid of each of these distribution center pairs. Additionally, we do not estimate a law of motion, since it is deterministic in our example. We also do not estimate

[34] There are two primary differences between the model developed in Section 2.1 and the model implied by our chain store game. The first difference is the timing of actions. In the chain store application, in period $t$, agents decide on store locations in period $t+1$. This makes the time $t$ period return deterministic, since all player actions have already been realized. The second difference is that the law of motion is deterministic, since the state is comprised of indicators over location choices, and period $t$ actions deterministically determine period $t+1$ location choices. Also, we assume perfect foresight by all competitors on the population variable, which represents the only variable in the state vector that is not a location indicator. As in Section 2.2, we assume that $\epsilon_{it} = 0$ for $t = 1, ..., T$ for our reference agent. As a result, the choice-specific value function for the value of placing facility $q$ in location $l$ in our chain store entry game for the reference agent $i$ takes the form:

$$V_i\left(\mathbf{s}_t; a^q_{ilt}, \overline{\sigma}_i\right) = \pi_i\left(\mathbf{s}_t\right) + \beta \mathbb{E}_{\mathbf{S}_{t+1}}\left[V_i(\mathbf{s}_{t+1}; \overline{\sigma}_i) | \mathbf{s}_t, \mathbf{a}_{it}\left(a^q_{ilt}\right)\right] \tag{5}$$

where,

$$\mathbb{E}_{\mathbf{S}_{t+1}}\left[V_i(\mathbf{s}_{t+1}; \overline{\sigma}_i) | \mathbf{s}_t, \mathbf{a}_{it}\left(a^q_{ilt}\right)\right] = \sum_{\mathbf{a}_{-tt} \in \mathcal{A}_{-it}} V_i(\mathbf{s}_{t+1}\left(\mathbf{s}_t, \mathbf{a}_{it}\left(a^q_{ilt}\right), \mathbf{a}_{-it}\right); \overline{\sigma}_i) \sigma_{-i}\left(\mathbf{a}_{-it} | \mathbf{s}_t\right)$$

where the randomness in $\mathbf{S}_{t+1}$ is due only to the randomness in the opponent's strategy $\boldsymbol{\sigma}_{-i}\left(\mathbf{a}_{-it} | \mathbf{s}_t\right)$, the notation $\mathbf{a}_{it}\left(a^q_{ilt}\right)$ indicates that facility choices by agent $i$ across all locations at time $t$ are conditional on the facility and location-specific choice $a^q_{ilt}$, and the notation $\mathbf{s}_{t+1}\left(\mathbf{s}_t, \mathbf{a}_{it}\left(a^q_{ilt}\right), \mathbf{a}_{-it}\right)$ indicates that $\mathbf{s}_{t+1}$ is conditional on $\mathbf{s}_t$, $\mathbf{a}_{it}\left(a^q_{ilt}\right)$, and $\mathbf{a}_{-it}$.

the payoff function for agent $i$, since we assume that it is known.

**Initial Strategy for Agent (Step 2).** In our illustration, for the first-step policy improvement, we choose distribution center locations randomly over all remaining MSA's not currently occupied by an own-firm general merchandise or food distribution center, respectively (the number chosen per period is constrained as previously described). We then open regular stores and supercenters in the closest feasible MSA's to these distribution centers exactly as described for the competitor. For second-step policy improvements and beyond, we use the previous step's improvement strategy as the fixed agent strategy.

**Simulating Play (Step 3).** We simulate play for the game using the opponent's strategy as described in Step 1, the law of motion, and $\overline{\sigma}_i$. We generate an initial state $\mathbf{s}_1^*$ by 1) for the agent, randomly placing distribution centers around the country and placing stores in the MSA's closest to these distribution centers, and 2) for the competitor, placing distribution centers in the exact locations chosen by Wal-Mart in the year 2000 and placing stores in the MSA's closest to these distribution centers. This results in seven food distribution centers, one general merchandise distribution center, two regular stores, and fourteen supercenters allocated in the initial state ($t = 2000$). In all specifications, store placement proceeds as follows. We open regular stores in the two closest feasible MSA's to each newly opened firm $i$ general merchandise distribution center. After making this decision, we determine the closest firm $i$ general merchandise distribution center to each newly opened firm $i$ food distribution center and open supercenters in the two feasible MSA's closest to the centroid of each of these distribution center pairs. We then generate period $t = 1$ actions. For the competitor, we choose locations $\mathbf{a}_{-i1}^*$ according to the opponent strategy from Step 1. For the agent, we choose a subset of facility locations using the fixed agent strategy $\overline{\sigma}_i$, and the remaining facilities randomly, i.e. by choosing $a_{il1}^{q*} = 1$ or $a_{il1}^{q*} = 0$ for each facility $q \in \{f, g, r, sc\}$ and each location $l = 1, ..., L$ by drawing from a uniform random variable. For example, in $t = 2000$, of the 8 supercenters agent $i$ must choose to enter in $t = 2001$, we choose 6 using $\overline{\sigma}_i$ and 2 randomly (i.e., we place supercenters in the two feasible locations with the highest random draws). These choices specify the state in period $t = 2$, i.e. $\mathbf{s}_2^*$. We choose each $a_{ilt}^{q*}$ and $\mathbf{a}_{-it}^*$ for $t = 2, ..., T-1$ similarly using $\overline{\sigma}_i$, a subset of random location draws, and the opponent strategy. For each location $l$ and period $t = 1, ..., T-1$, we calculate the expected profits generated by each choice $a_{ilt}^{q*} \in \{0, 1\}$, i.e. the simulated sums presented in definition 3, substituting $a_{ilt}^{q*}$ for $a_{ilt}^*$. This provides us with a sample of simulated data of the form $\left\{V_i\left(\mathbf{s}_{lt}^*, a_{ilt}^{q*}; \overline{\sigma}_i\right), \mathbf{s}_{lt}^*, a_{ilt}^{q*}\right\}_{l=1,t=1}^{L,T}$ for firm $i$ and each simulation run.

**Estimating Choice-Specific Value Function (Step 4).** We focus on eight estimands, $V_i\left(\mathbf{s}_t, a_{ilt}^q; \overline{\sigma}_i\right)$ for each $q \in \{f, g, r, sc\}$ and choice $a_{ilt}^q \in \{0, 1\}$. Defining the state as "location-specific" through the location-specific population variable allows us to exploit differences in value across locations when estimating the choice-specific value functions. This simplification is not necessary for implementing Algorithm 1 but greatly reduces the simulation burden, since each individual simulation effectively provides 227 sample observations rather than 1. We employ CWGB Algorithm 2, with outcomes $V_i\left(\mathbf{s}_{lt}^{*}, a_{ilt}^{q*}; \overline{\sigma}_i\right)$ and regressors $\mathbf{s}_{lt}^{*}$, and we pool observations across simulation runs, locations, and time. This estimation process produces eight models, denoted as $\widehat{V}_i\left(\widetilde{\mathbf{s}}_t, a_{ilt}^{q}; \overline{\sigma}_i\right)$ for $q \in \{f, g, r, sc\}$ and $a_{ilt}^{q} \in \{0,1\}$, where we abuse notation by not acknowledging the potential differences in the dimension-reduced vectors $\widetilde{\mathbf{s}}_t$ across models, which need not include the same state variables.[35]

**One-Step Improvement Policy (Step 5).** To derive each $\widehat{\sigma}_i^1$, we first compute the difference in the CWGB estimated local choice and facility-specific value functions between placing a facility $q$ in location $l$ versus not, i.e. $\widehat{V}_i\left(\widetilde{\mathbf{s}}_t, a_{ilt}^{q} = 1; \overline{\sigma}_i\right) - \widehat{V}_i\left(\widetilde{\mathbf{s}}_t, a_{ilt}^{q} = 0; \overline{\sigma}_i\right)$, for each facility type $q \in \{f, g, r, sc\}$ and location $l = 1, ..., L$. Then, for each $q$, we rank these differences over all locations and choose the highest ranking locations to place the pre-specified number of new facilities allowed in each period. This algorithm for choosing facility locations over all time periods represents our one-step policy improvement policy $\widehat{\sigma}_i^1$.[36] A second-step policy improvement is obtained by using $\widehat{\sigma}_i^1$ to generate $\overline{\sigma}_i$, and repeating the steps of Algorithm 1.[37]

## 3.5 Chain Store Entry Game Results

The models resulting from using the CWGB procedure are presented in Table 1. Table 1 lists both the final coefficients associated with selected state variables in each model, as well as the proportion of CWGB iterations for which univariate models of these state variables resulted in the best fit (i.e. the selection frequency). For example, during the CWGB estimation process which generated the model for general merchandise distribution centers and $a_{ilt}^{g} = 1$, i.e. $\widehat{V}_i\left(\widetilde{\mathbf{s}}_t, a_{ilt}^{g} = 1; \overline{\sigma}_i\right)$, univariate models of the population variable were selected in 53 percent of the iterations.

This table reveals three salient features of these models. The first is that the CWGB procedure *drastically* reduces the number of state variables for each model, from 1817 to an average of 7 variables. For example, one of the most parsimonious models estimated is that for regular stores with $a_{ilt}^{r} = 1$, i.e. $\widehat{V}_i\left(\widetilde{\mathbf{s}}_t, a_{ilt}^{r} = 1; \overline{\sigma}_i\right)$, which consists of a constant, the population covariate, and indicators for own regular store entry in five markets: Allentown, PA, Hartford, CT, Kansas City, MO, San Francisco, CA, and Augusta, GA. This reduces the average state space cardinality per

[35] In our chain store entry game application, we estimate the choice-specific value function using CWGB rather than OLS, where OLS is proposed in Section 2.2. This is necessary because our state vector remains high-dimensional in the chain store game, since we do not estimate opponent policy functions, our agent's payoff function is known, and since the law of motion is deterministic. In settings where the policy functions of opponents and (if necessary) the agent's payoff function are estimated using CWGB, the CWGB estimator typically reduces the dimension of the state vector sufficiently, making further model selection unnecessary when estimating the choice-specific value function.

[36] We note that by choosing $\widehat{\sigma}_i^1$ in this way, we do not choose a true greedy maximum action vector $\mathbf{a}_{it}$ in each period $t$, since focusing on location and facility-specific choice-specific value functions effectively assumes that facilities in all other locations are chosen according to $\overline{\sigma}_i$. Nonetheless, we show in the next Section that $\widehat{\sigma}_i^1$ generates a substantial improvement in our illustration.

[37] Our code for implementing the chain store application is available at: http://abv8.me/4g8.

time period from more than $10^{85}$ (not including population) to 32 ($2^5$) multiplied by the cardinality of the population variable.

The second and related feature is that all models draw from a relatively small subset of the original 1816 own and competitor facility presence indicators. It is also interesting to observe which MSA indicators comprise this subset, which is made up primarily of indicators associated with medium-sized MSA's in our sample scattered across the country. What explains this pattern is that in the simulated data used for estimation, even across many simulations, only a subset of MSA's are occupied by firm facilities. Among those occupied, occasionally, the agent experiences either heavy gains or heavy losses, which are compounded over time, since we do not allow firms to close facilities once they are opened. These particularly successful or painful facility placements tend to produce univariate models that explain levels of the choice-specific value function well, which results in their selection by the CWGB procedure, typically across several models. For example, a series of particularly heavy losses were sustained by the agent as a result of placing a regular store in Augusta, GA, which induced the CWGB procedure to choose this indicator at a high frequency–25 percent, 15 percent, and 18 percent of iterations–across three different models, with each model associating this indicator with a large negative coefficient. As a result, our one-step improvement policy $\widehat{\sigma}_i^1$ tended to avoid placing distribution centers and stores in this location.

The third salient feature apparent from Table 1 is that population is the state variable selected most consistently. Across all CWGB models, population is selected with a frequency of roughly 53 percent in each model, while facility presence indicator variables are selected at much smaller rates.[38]

For a variety of parameter specifications, Table 2 compares per-store revenues, operating income, margins, and costs, averaged over all time periods and simulations, for three strategies: 1) the one-step improvement policy for the agent, 2) a random choice agent strategy, where distribution centers and stores are chosen as specified for $\overline{\sigma}_i$ (in all time periods $t, ..., T-1$), and 3) the competitor's strategy (benchmark). The three parameter specifications correspond to three scenarios: a baseline specification, a high penalty for urban locations, and high distribution costs.[39] As shown in this table when comparing revenues, in the baseline scenario, the one-step improvement policy outperforms the random choice strategy by 354 percent. Similarly, it outperforms the competitor's strategy by 293 percent. In the high urban penalty and high distribution cost specifications, the one-step improvement policy outperforms the random choice strategy by 355 percent and 350

[38]For comparison, in the Appendix (Table 8), we estimate OLS models of the choice-specific value functions of interest by using only the state variables selected by the corresponding boosted regression model from Table 1. Overall, the post selection OLS models have similar coefficients to the boosted regression models.

[39]The parameter values in the baseline specification were chosen to calibrate competitor per-store returns to those actually received by Wal-Mart in the U.S. during the same time period. The high urban penalty and high distribution cost specifications were chosen to explore the sensitivity of the relative returns generated by our one-step improvement policy to these parameters.

percent, respectively, and the competitor strategy by 293 percent and 294 percent, respectively. The relative returns of the one-step improvement policies seem fairly invariant to the parameter specifications, which is understandable since each is constructed using a choice-specific value function estimated under each respective parameter specification. The one-step improvement policies appear to adjust the agent's behavior accordingly in response to these parameter changes.

Table 3 provides a comparison of per-store revenues, operating income, margins, and costs by revenue type and strategy, averaged over all time periods and simulations in the baseline scenario, and compares these to Wal-Mart's revenue and operating income figures for 2005. The competitor's strategy generates average operating income per store (of both types) of \$4.40 million, which is similar to that actually generated by Wal-Mart in 2005 of \$4.49 million, and larger than the of the random choice strategy, which generates \$3.26 million. The one-step improvement policy does much better, with an operating income per store of over \$18 million, corresponding to revenues per store of \$244 million, versus \$62 million for the competitor and \$54 million for the random choice strategy. Moreover, the one-step improvement policy achieves a slightly higher operating margin than the other two strategies: 7.55 percent versus 7.09 percent for the competitor and 6.07 percent for the random choice strategy. One reason for the success of the improvement strategy appears to be that it targets higher population areas than the other strategies, which generates higher revenues in our simulation. Specifically, it targets MSA's with an average population of 2.39 million versus 0.84 million for the random choice strategy and 1.01 million for the competitor. That the average population of competitor locations is relatively small is understandable, since the competitor progresses as Wal-Mart did, placing distribution centers and stores primarily in the Midwest and radiating out towards the east coast, while the improvement strategy searches for value-improving locations for distribution centers and stores in a less restricted manner across the country.

These facility placement pattern differences are visually detectable in Figure 2, which shows distribution center and store location patterns for the agent and the competitor in a representative simulation, with the agent using the one-step improvement policy. As shown in these figures, the agent scatters distribution centers and stores across the population dense MSA's in the United States, while the competitor has a concentration of distribution centers and stores primarily in the Midwest and east coast. By the end of 2006, the agent has a strong presence on the West coast with eight facilities in California, while the competitor only opens four facilities in this region. Although visually these pattern differences seem subtle, they generate large differences in revenues and operating income, as highlighted by Table 3.

Finally, we generate additional policy improvements after the first one-step policy improvement by randomly deviating from each one-step policy improvement and otherwise repeating the steps of Algorithm 1. As shown in Figure 3, the first one-step policy improvement generates almost all gains in payoffs, and all subsequent policy improvements generate returns very close to the first one-step policy improvement. For example, using the baseline specification, while the first one-step policy improvement generates total revenues per store for the agent of \$244.44 million, up from the total revenues per store generated by the random choice strategy of \$53.61 million, the second through fifth-step policy improvements generate total revenues per store of between \$244.43 and \$244.60 million. Similarly, while the first one-step policy improvement generates operating income per store of \$11.55 million, up from the operating income per store generated by the random choice strategy of \$1.94 million, the second through fifth-step policy improvements generate operating income per store of between \$11.23 million and \$11.55 million.

## 4 Conclusion

This paper develops a method for deriving policy function improvements for a single agent in high-dimensional Markov dynamic optimization problems and in particular dynamic games. The approach has two attributes that make it useful for deriving policies in realistic game settings. The first is that we impose no equilibrium restrictions on opponent behavior and instead estimate opponent strategies directly from data on past game play. This allows us to accommodate a richer set of opponent strategies than equilibrium assumptions would imply. A second is that we use a Machine Learning method to estimate opponent strategies and, if needed, the payoff function for a reference agent and the law of motion. This method makes estimation of the agent's choice-specific value function feasible in high-dimensional settings, since as a consequence of estimation, the estimator reduces the dimension of the state space in a data-driven manner. Data-driven dimension-reduction proceeds by choosing the state variables that minimize the loss associated with predicting the outcomes of interest according to a fixed metric, making the estimates low-dimensional approximations of the original functions. In our illustration, we show that our functions of interest are well-approximated by these low-dimensional representations, suggesting that data-driven dimension-reduction might serve as a helpful tool for economists seeking to make their models less computationally wasteful.

We use the method to derive policy function improvements for a single retailer in a dynamic spatial competition game among two chain store retailers similar to the one considered by Holmes (2011). This game involves location choices for stores and distribution centers over a finite number of time periods. This game becomes high-dimensional primarily because location choices involve complementarities across locations. For example, clustering own stores closer together can lower distribution costs but also can cannibalize own store revenues, since consumers substitute demand between nearby stores. For the same reason, nearby competitor stores lower revenues for a given store. Since we characterize the state as a vector enumerating the current network of stores and distribution centers for both competitors, the cardinality of the state becomes extremely large (on the order of $> 10^{85}$ per time period), even given a relatively small number of possible locations (227). We derive an improvement policy and show that this policy generates a nearly 300 percent improvement over a strategy designed to approximate Wal-Mart's actual facility placement during the same time period (2000 to 2006).

## References

[1] Abramson, Bruce (1990), "Expected-Outcome: A General Model of Static Evaluation," *IEEE Transactions on Pattern Analysis and Machine Intelligence* 12(2), 182-193.

[2] Aguirregabiria, Victor and Gustavo Vicentini (2014), "Dynamic Spatial Competition Between Multi-Store Firms," Working Paper, February.

[3] Bajari, Patrick, Ying Jiang, and Carlos A. Manzanares (2015),"Improving Policy Functions in High-Dimensional Dynamic Games: An Entry Game Example," Working Paper, March.

[4] Bajari, Patrick, Han Hong, and Denis Nekipelov (2013), "Game Theory and Econometrics: A Survey of Some Recent Research," *Advances in Economics and Econometrics*, 10th World Congress, Vol. 3, Econometrics, 3-52.

[5] Belloni, Alexandre, Victor Chernozhukov, and Christian Hansen (2010), "Inference Methods for High-Dimensional Sparse Econometric Models," *Advances in Economics & Econometrics*, ES World Congress 2010, ArXiv 2011.

[6] Bertsekas, Dimitri P. (2012). *Dynamic Programming and Optimal Control*, Vol. 2, 4th ed. Nashua, NH: Athena Scientific.

[7] ——— (2013), "Rollout Algorithms for Discrete Optimization: A Survey," In Pardalos, Panos M., Ding-Zhu Du, and Ronald L. Graham, eds., *Handbook of Combinatorial Optimization*, 2nd ed., Vol. 21, New York: Springer, 2989-3013.

[8] Boros, Endre, Vladimir Gurvich, and Emre Yamangil (2013), "Chess-Like Games May Have No Uniform Nash Equilibria Even in Mixed Strategies," *Game Theory* 2013, 1-10.

[9] Bowling, Michael, Neil Burch, Michael Johanson, and Oskari Tammelin (2015), "Heads-Up Limit Hold'em Poker is Solved," *Science* 347(6218), 145-149.

[10] Breiman, Leo (1998), "Arcing Classifiers (with discussion)," *The Annals of Statistics* 26(3), 801-849.

[11] ——— (1999), "Prediction Games and Arcing Algorithms," *Neural Competition* 11(7), 1493-1517.

[12] Bulow, Jeremy, Jonathan Levin, and Paul Milgrom (2009), "Winning Play in Spectrum Auctions," Working Paper, February.

[13] Chinchalkar, Shirish S. (1996), "An Upper Bound for the Number of Reachable Positions", *ICCA Journal* 19(3), 181–183.

[14] Ellickson, Paul B., Stephanie Houghton, and Christopher Timmins (2013), "Estimating network economies in retail chains: a revealed preference approach," *The RAND Journal of Economics* 44(2), 169-193.

[15] Friedman, Jerome H. (2001), "Greedy Function Approximation: A Gradient Boosting Machine," *The Annals of Statistics* 29(5), 1189-1232.

[16] Friedman, Jerome H., Trevor Hastie, and Robert Tibshirani (2000), "Additive Logistic Regression: A Statistical View of Boosting," *The Annals of Statistics* 28(2), 337-407.

[17] Hansen, Bruce, (2015), "The Risk of James-Stein and Lasso Shrinkage," *Econometric Reviews*, forthcoming.

[18] Hastie, Trevor, Robert Tibshirani, and Jerome H. Friedman (2009). *The Elements of Statistical Learning (2nd ed.)*, New York: Springer Inc.

[19] Hofner, Benjamin, Andreas Mayr, Nikolay Robinzonov, and Matthias Schmid (2014), "Model-based Boosting in R," *Computational Statistics* 29(1-2), 3-35.

[20] Holmes, Thomas J. (2011), "The Diffusion of Wal-Mart and Economies of Density," *Econometrica* 79(1), 253-302.

[21] Jia, Panle (2008), "What Happens When Wal-Mart Comes to Town: An Empirical Analysis of the Discount Retailing Industry," *Econometrica* 76(6), 1263-1316.

[22] Nishida, Mitsukuni (2014), "Estimating a Model of Strategic Network Choice:
The Convenience-Store Industry in Okinawa," *Marketing Science* 34(1), 20-38.

[23] Pesendorfer, Mardin, and Philipp Schmidt-Dengler (2008), "Asymptotic Least Squares Estimators for Dynamic Games," *Review of Economic Studies* 75(3), 901-928.

[24] Rust, John (1987), "Optimal Replacement of GMC Bus Engines: An Empirical Model of Harold Zurcher," *Econometrica* 55(5), 999-1033.

[25] Schmid, Matthias and Torsten Hothorn (2008), "Boosting Additive Models Using Component-Wise P-Splines," *Computational Statistics & Data Analysis* 53(2), 298-311.

[26] Xu, Zhixiang, Gao Huang, Kilian Q. Weinberger, and Alice X. Zheng (2014), "Gradient boosted feature selection," In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, 522-531. ACM.

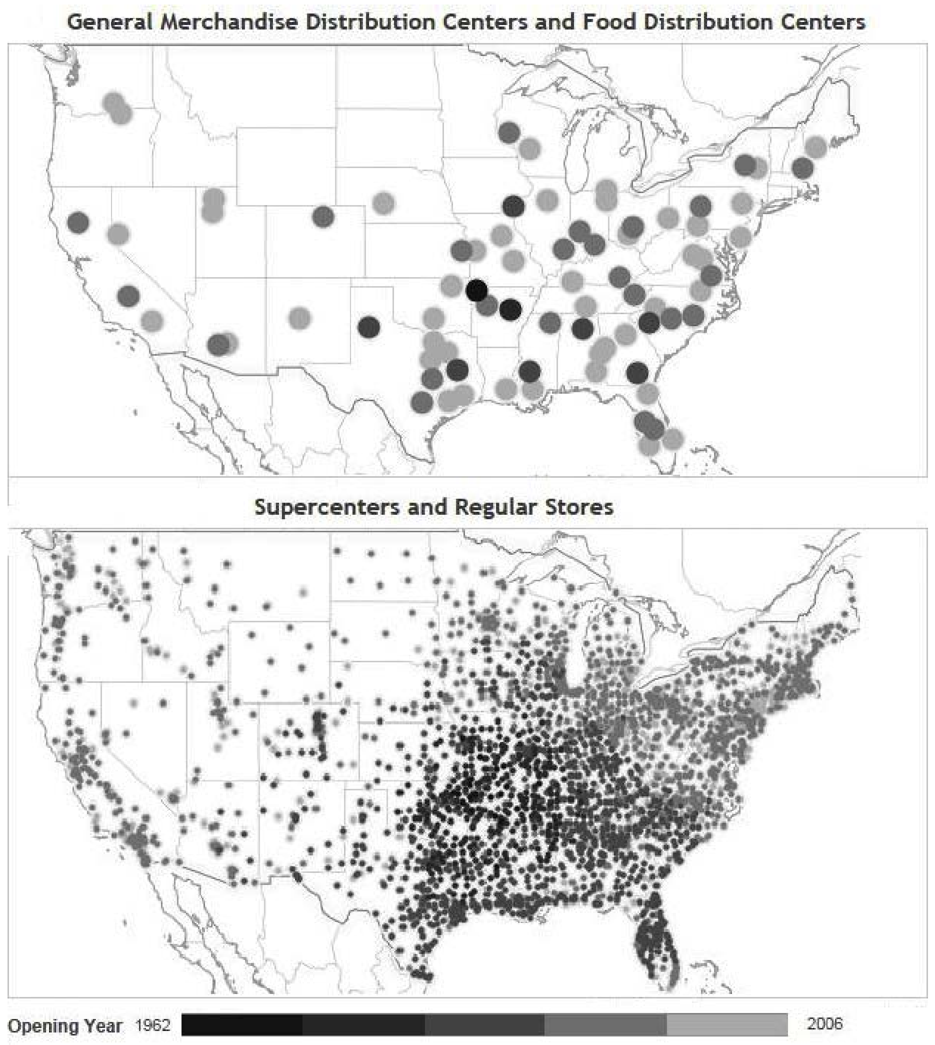

General Merchandise Distribution Centers and Food Distribution Centers
Supercenters and Regular Stores
Opening Year
1962
2006

Figure 1: Wal-Mart Distribution Center and Store Diffusion Map (1962 to 2006).

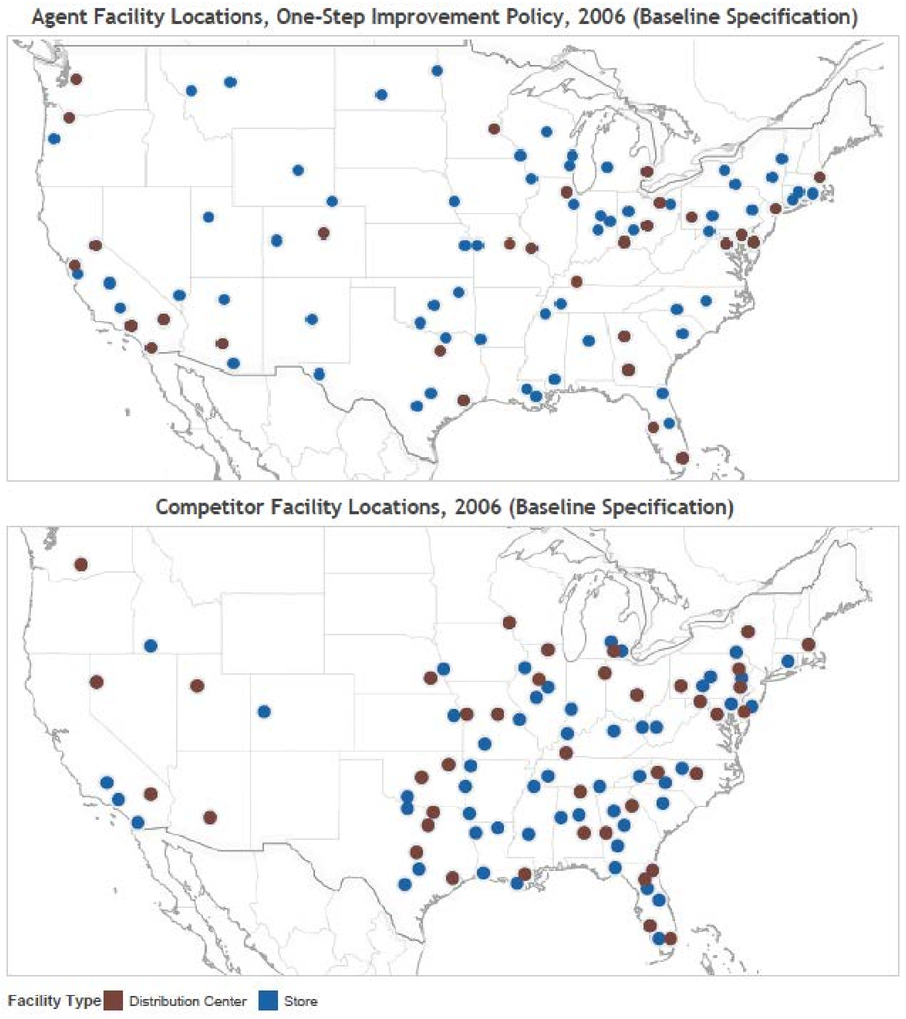
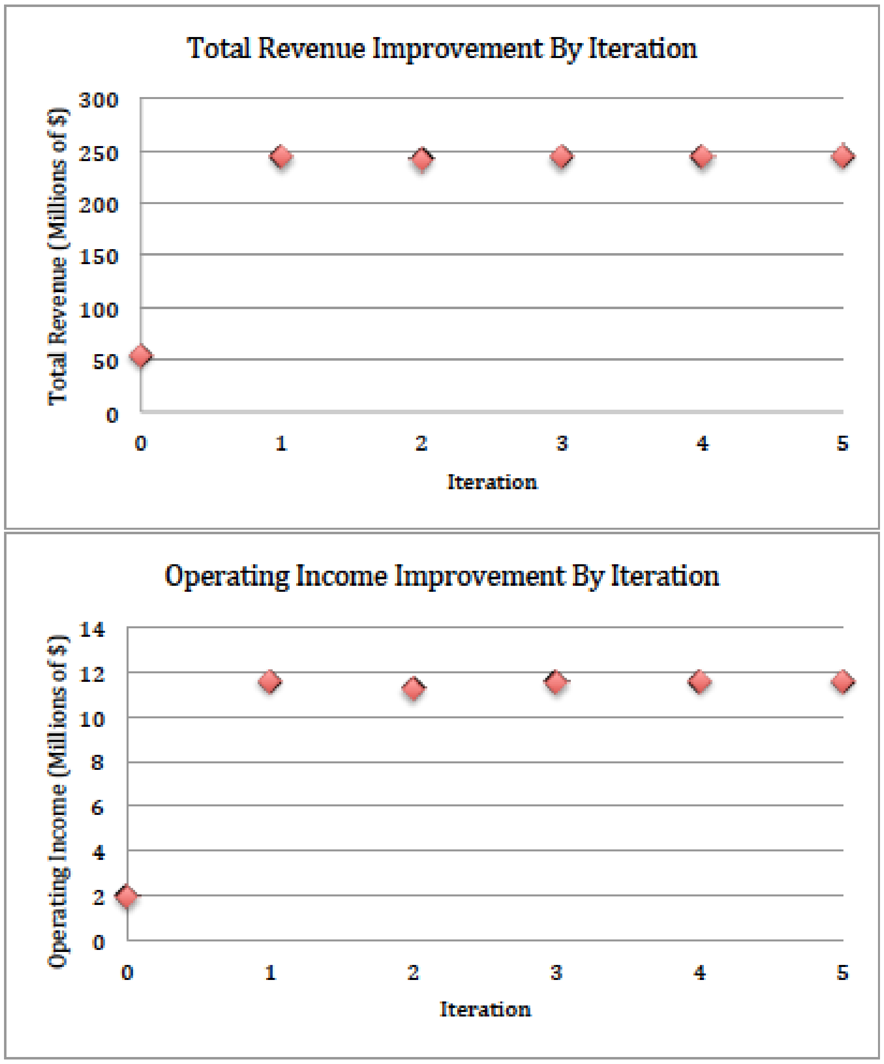

Agent Facility Locations, One-Step Improvement Policy, 2006 (Baseline Specification)
Competitor Facility Locations, 2006 (Baseline Specification)
Facility Type
Distribution Center
Store

Figure 2: Simulation Results, Representative Simulation (2000 to 2006).



**Note: Training samples based on 500 simulation runs. Test samples based on one simulation run. Iteration 0 represents return from random facility strategy. Iterations 1 to 5 trained using one-step policy improvements upon strategy of previous iteration. All iterations use baseline parameter specification.**

Figure 3: Multi-step Policy Improvement.

Table 1: Choice-Specific Value Function Estimates, Boosted Regression Models (Baseline Specification)

| Choice-Specifc Value Function | $\widehat{V}_i(a^g_{ilt}=1)$ | $\widehat{V}_i(a^g_{ilt}=0)$ | $\widehat{V}_i(a^f_{ilt}=1)$ | $\widehat{V}_i(a^f_{ilt}=0)$ | $\widehat{V}_i(a^r_{ilt}=1)$ | $\widehat{V}_i(a^r_{ilt}=0)$ | $\widehat{V}_i(a^{sc}_{ilt}=1)$ | $\widehat{V}_i(a^{sc}_{ilt}=0)$ |
|---|---|---|---|---|---|---|---|---|
| Population | $1.64\times10^1$ | $1.42\times10^1$ | $1.93\times10^1$ | $1.42\times10^1$ | $1.18\times10^1$ | $1.42\times10^1$ | $1.95\times10^1$ | $1.42\times10^1$ |
| | (0.530) | (0.526) | (0.526) | (0.526) | (0.531) | (0.526) | (0.533) | (0.526) |
| Own Entry Regstore Allentown, PA | $-1.69\times10^7$ | | $-1.53\times10^7$ | | $-2.09\times10^6$ | | $-7.81\times10^6$ | |
| | (0.113) | | (0.072) | | (0.058) | | (0.053) | |
| Own Entry Regstore Boulder, CO | $-4.85\times10^6$ | | $-4.23\times10^6$ | | | | $-2.37\times10^6$ | |
| | (0.061) | | (0.065) | | | | (0.064) | |
| Own Entry Regstore Hartford, CT | $-8.70\times10^6$ | | $-6.39\times10^6$ | | $-1.15\times10^6$ | | $-3.57\times10^6$ | |
| | (0.051) | | (0.059) | | (0.054) | | (0.059) | |
| Own Entry Regstore Kansas City, MO | $-1.55\times10^7$ | | $-5.42\times10^6$ | | $-2.13\times10^6$ | | $-3.54\times10^6$ | |
| | (0.049) | | (0.026) | | (0.045) | | (0.035) | |
| Own Entry Regstore San Francisco, CA | $-1.08\times10^7$ | | | | $-1.77\times10^6$ | | $-1.11\times10^6$ | |
| | (0.196) | | | | (0.159) | | (0.079) | |
| Own Entry Regstore Augusta, GA | | | $-1.48\times10^7$ | | $-3.57\times10^6$ | | $-9.28\times10^6$ | |
| | | | (0.252) | | (0.153) | | (0.177) | |
| Rival Entry Regstore Albany, GA | | $-8.47\times10^6$ | | $-8.47\times10^6$ | | $-8.48\times10^6$ | | $-8.47\times10^6$ |
| | | (0.302) | | (0.303) | | (0.303) | | (0.302) |
| Rival Entry GM Dist Clarksville, TN | | $-1.34\times10^6$ | | $-1.35\times10^6$ | | $-1.35\times10^6$ | | $-1.34\times10^6$ |
| | | (0.032) | | (0.033) | | (0.033) | | (0.032) |
| Rival Entry GM Dist Columbia, MO | | $-5.09\times10^5$ | | $-5.05\times10^5$ | | $-5.06\times10^5$ | | $-5.05\times10^5$ |
| | | (0.015) | | (0.015) | | (0.015) | | (0.015) |
| Rival Entry GM Dist Cumberland, MD | | $-1.35\times10^6$ | | $-1.35\times10^6$ | | $-1.35\times10^6$ | | $-1.35\times10^6$ |
| | | (0.050) | | (0.050) | | (0.050) | | (0.050) |
| Rival Entry GM Dist Dover, DE | | $-1.81\times10^6$ | | $-1.80\times10^6$ | | $-1.80\times10^6$ | | $-1.81\times10^6$ |
| | | (0.051) | | (0.050) | | (0.050) | | (0.051) |
| Rival Entry GM Dist Hickory, NC | | $-9.74\times10^5$ | | $-9.65\times10^5$ | | $-9.66\times10^5$ | | $-9.74\times10^5$ |
| | | (0.024) | | (0.023) | | (0.023) | | (0.024) |
| Constant | $4.12\times10^7$ | $9.37\times10^6$ | $3.11\times10^7$ | $9.37\times10^6$ | $6.24\times10^6$ | $9.38\times10^6$ | $1.72\times10^7$ | $9.37\times10^6$ |

Note: Selection frequencies are shown in parentheses. Results are based on 1000 simulation runs. We use *glmboost* function in *mboost* package in R with linear base-learners, a squared-error loss function used for observation-weighting, 1000 iterations per boosted regression model, and a step size of 0.01. The covariate *Own Entry Regstore City, State* represents own-firm regular store entry in the listed MSA; similarly, *Rival Entry Regstore City, State* represents competitor regular store entry in the listed MSA, and *Rival Entry GM Dist City, State* represents competitor general merchandise distribution center entry in the listed MSA.

Table 2: Simulation Results by Specification (Per-Store Average)

| Model | CWGB 1<br>(baseline) | CWGB 2<br>(high-urban-penalty) | CWGB 3<br>(high-dist-cost) |
|---|---|---|---|
| **Revenue (millions of $)** | | | |
| Agent, One-Step Improvement | 244.23 | 244.09 | 245.87 |
| Agent, Random Choice | 53.79 | 53.67 | 54.64 |
| Competitor | 62.14 | 62.15 | 62.47 |
| **Operating Income (millions of $)** | | | |
| Agent, One-Step Improvement | 18.44 | 18.29 | 17.91 |
| Agent, Random Choice | 3.26 | 3.09 | 2.78 |
| Competitor | 4.40 | 4.25 | 4.01 |
| **Operating Margin** | | | |
| Agent, One-Step Improvement | 7.55% | 7.49% | 7.28% |
| Agent, Random Choice | 6.07% | 5.77% | 5.08% |
| Competitor | 7.09% | 6.83% | 6.42% |
| **Variable Cost Labor (millions of $)** | | | |
| Agent, One-Step Improvement | 21.28 | 21.26 | 21.42 |
| Agent, Random Choice | 5.64 | 5.62 | 5.73 |
| Competitor | 5.91 | 5.91 | 5.93 |
| **Variable Cost Land (millions of $)** | | | |
| Agent, One-Step Improvement | 0.20 | 0.20 | 0.20 |
| Agent, Random Choice | 0.05 | 0.05 | 0.05 |
| Competitor | 0.04 | 0.04 | 0.04 |
| **Variable Cost Other (millions of $)** | | | |
| Agent, One-Step Improvement | 17.23 | 17.21 | 17.33 |
| Agent, Random Choice | 4.78 | 4.76 | 4.84 |
| Competitor | 5.16 | 5.16 | 5.18 |
| **Import Distribution Cost (millions of $)** | | | |
| Agent, One-Step Improvement | 0.79 | 0.80 | 1.21 |
| Agent, Random Choice | 0.74 | 0.73 | 1.10 |
| Competitor | 0.56 | 0.56 | 0.85 |
| **Domestic Distribution Cost (millions of $)** | | | |
| Agent, One-Step Improvement | 0.60 | 0.56 | 0.84 |
| Agent, Random Choice | 0.37 | 0.37 | 0.55 |
| Competitor | 0.28 | 0.28 | 0.42 |
| **Urban Cost Penalty (millions of $)** | | | |
| Agent, One-Step Improvement | 0.34 | 0.51 | 0.34 |
| Agent, Random Choice | 0.32 | 0.48 | 0.32 |
| Competitor | 0.32 | 0.48 | 0.32 |

Note: Results are based on 1000 simulation runs for each specification. The parameter values for each specification are available in the Appendix.

Table 3: Simulation Results by Merchandise Type (Baseline Specification, Per-Store Average)

| Statistic | One-Step Improvement | Random Choice | Competitor | Wal-Mart (2005) |
|---|---|---|---|---|
| **Revenue (millions of $)** | | | | |
| All Goods | 244.23 | 53.79 | 62.14 | 60.88 |
| General Merchandise | 119.19 | 30.81 | 36.61 | – |
| Food | 180.71 | 33.13 | 36.89 | – |
| **Operating Income (millions of $)** | | | | |
| All Goods | 18.44 | 3.26 | 4.40 | 4.49 |
| General Merchandise | 8.46 | 1.64 | 2.43 | – |
| Food | 14.43 | 2.34 | 2.85 | – |
| **Operating Margin (millions of $)** | | | | |
| All Goods | 7.55% | 6.07% | 7.09% | 7.38% |
| General Merchandise | 7.10% | 5.33% | 6.64% | – |
| Food | 7.99% | 7.05% | 7.73% | – |
| **Import Distribution Cost (millions of $)** | | | | |
| All Goods | 0.79 | 0.74 | 0.56 | – |
| General Merchandise | 0.55 | 0.43 | 0.30 | – |
| Food | 0.35 | 0.44 | 0.38 | – |
| **Domestic Distribution Cost (millions of $)** | | | | |
| All Goods | 0.60 | 0.37 | 0.28 | – |
| General Merchandise | 0.51 | 0.28 | 0.21 | – |
| Food | 0.13 | 0.12 | 0.10 | – |
| **Variable Costs and Urban Penalty (millions of $)** | | | | |
| Labor Cost, All Goods | 21.28 | 5.64 | 5.91 | – |
| Land Cost, All Goods | 0.20 | 0.05 | 0.04 | – |
| Other Cost, All Goods | 17.23 | 4.78 | 5.16 | – |
| Urban Cost Penalty, All Goods | 0.34 | 0.32 | 0.32 | – |
| **MSA Population** | | | | |
| Population (millions) | 2.39 | 0.84 | 1.01 | – |
| Population Density (Population/Square Miles) | 528 | 296 | 264 | – |

Note: Results are based on 1000 simulation runs. The baseline specification parameter values are available in the Appendix.

## 5 Appendix

### 5.1 Section 3.2 Details

**Import Distribution Centers.** During the 2000 to 2006 time period, Wal-Mart operated import distribution centers in Mira Loma, CA, Statesboro, GA, Elwood, IL, Baytown, TX, and Williamsburg, VA. See http://www.mwpvl.com/html/walmart.html for this list. For our simulation (over all time periods), we endow each firm with an import distribution center in each MSA in our sample physically closest to the above listed cities. These include [with Wal-Mart's corresponding import distribution center location in brackets]: Riverside, CA [Mira Loma, CA], Savannah, GA [Statesboro, GA], Kankakee, IL [Elwood, IL], Houston, TX [Baytown, TX], and Washington, DC [Williamsburg, VA].

**General Merchandise and Food Distribution Centers**. We constrain each competitor to open the same number of general merchandise and food distribution centers in each period as actually opened by Wal-Mart. Additionally, we constrain the competitor to open general merchandise and food distribution centers in the MSA's in our sample closest to the actual distribution centers opened by Wal-Mart in the same period. Tables 4 and 5 present both the distribution centers opened by Wal-Mart, as well as the distribution center locations opened by the competitor in all simulations.

Table 4: General Merchandise Distribution Centers

| Year | Wal-Mart Location | Competitor's Location |
|---|---|---|
| 2000 | LaGrange, GA | Columbus, GA |
| 2001 | Coldwater, MI | Jackson, MI |
| 2001 | Sanger, TX | Sherman, TX |
| 2001 | Spring Valley, IL | Peoria, IL |
| 2001 | St. James, MO | Columbia, MO |
| 2002 | Shelby, NC | Hickory, NC |
| 2002 | Tobyhanna, PA | Scranton, PA |
| 2003 | Hopkinsville, KY | Clarksville, TN |
| 2004 | Apple Valley, CA | Riverside, CA |
| 2004 | Smyrna, DE | Dover, DE |
| 2004 | St. Lucie County, FL | Miami, FL |
| 2005 | Grantsville, UT | Salt Lake City, UT |
| 2005 | Mount Crawford, VA | Cumberland, MD |
| 2005 | Sealy, TX | Houston, TX |
| 2006 | Alachua, FL | Gainesville, FL |

Table 5: Food Distribution Centers

| Year | Wal-Mart Location | Competitor's Location |
|---|---|---|
| 2000 | Corinne, UT | Salt Lake City, UT |
| 2000 | Johnstown, NY | Utica-Rome, NY |
| 2000 | Monroe, GA | Athens, GA |
| 2000 | Opelika, AL | Columbus, GA |
| 2000 | Pauls Valley, OK | Oklahoma City, OK |
| 2000 | Terrell, TX | Dallas, TX |
| 2000 | Tomah, WI | LaCrosse, WI |
| 2001 | Auburn, IN | FortWayne, IN |
| 2001 | Harrisonville, MO | KansasCity, MO |
| 2001 | Robert, LA | New Orleans, LA |
| 2001 | Shelbyville, TN | Huntsville, AL |
| 2002 | Cleburne, TX | Dallas, TX |
| 2002 | Henderson, NC | Raleigh, NC |
| 2002 | MacClenny, FL | Jacksonville, FL |
| 2002 | Moberly, MO | Columbia, MO |
| 2002 | Washington Court House, OH | Columbus, OH |
| 2003 | Brundidge, AL | Montgomery, AL |
| 2003 | Casa Grande, AZ | Phoenix, AZ |
| 2003 | Gordonsville, VA | Washington, DC |
| 2003 | New Caney, TX | Houston, TX |
| 2003 | Platte, NE | Cheyenne, WY |
| 2003 | Wintersville (Steubenville), OH | Pittburgh, PA |
| 2004 | Fontana, CA | Riverside,CA |
| 2004 | Grandview, WA | Yakima,WA |
| 2005 | Arcadia, FL | PuntaGorda,FL |
| 2005 | Lewiston, ME | Boston,MA |
| 2005 | Ochelata, OK | Tulsa,OK |
| 2006 | Pottsville, PA | Reading,PA |
| 2006 | Sparks, NV | Reno,NV |
| 2006 | Sterling, IL | Rockford,IL |
| 2007 | Cheyenne, WY | Cheyenne,WY |
| 2007 | Gas City, IN | Muncie,IN |

## 5.2 Section 3.3 Details

**State Space Cardinality Calculation**. Calculations for the cardinality of the part of the state space attributable to firm $i$ as defined in our illustration are listed in Table 6.

Table 6: State Space Cardinality Calculation

| Year | 2000 | 2001 | 2002 | 2003 | 2004 | 2005 | 2006 | Total facilities |
|---|---|---|---|---|---|---|---|---|
| **Number of location decisions** | | | | | | | | |
| Regular Stores | 2 | 8 | 4 | 2 | 6 | 6 | 2 | 30 |
| Supercenters | 14 | 8 | 10 | 12 | 4 | 6 | 6 | 60 |
| Food DC | 7 | 4 | 5 | 6 | 2 | 3 | 3 | 30 |
| General Merchandise DC | 1 | 4 | 2 | 1 | 3 | 3 | 1 | 15 |
| **Number of feasible locations** | | | | | | | | |
| Regular Stores | 227 | 211 | 195 | 181 | 167 | 157 | 145 | – |
| Supercenters | 225 | 203 | 191 | 179 | 161 | 151 | 143 | – |
| Food DC | 227 | 220 | 216 | 211 | 205 | 203 | 200 | – |
| General Merchandise DC | 227 | 226 | 222 | 220 | 219 | 216 | 213 | – |
| **Number of possible combinations** | | | | | | | | |
| Regular Stores | $2.57\times10^{04}$ | $8.52\times10^{13}$ | $5.84\times10^{07}$ | $1.63\times10^{04}$ | $2.75\times10^{10}$ | $1.89\times10^{10}$ | $1.04\times10^{04}$ | – |
| Supercenters | $6.47\times10^{21}$ | $6.22\times10^{13}$ | $1.40\times10^{16}$ | $1.55\times10^{18}$ | $2.70\times10^{07}$ | $1.49\times10^{10}$ | $1.07\times10^{10}$ | – |
| Food DC | $5.61\times10^{12}$ | $9.50\times10^{07}$ | $3.74\times10^{09}$ | $1.14\times10^{11}$ | $2.09\times10^{04}$ | $1.37\times10^{06}$ | $1.31\times10^{06}$ | – |
| General Merchandise DC | $2.27\times10^{02}$ | $1.06\times10^{08}$ | $2.45\times10^{04}$ | $2.20\times10^{02}$ | $1.73\times10^{06}$ | $1.66\times10^{06}$ | $2.13\times10^{02}$ | – |
| **State space cardinality** | $2.11\times10^{41}$ | $5.33\times10^{43}$ | $7.51\times10^{37}$ | $6.34\times10^{35}$ | $2.68\times10^{28}$ | $6.40\times10^{32}$ | $3.12\times10^{22}$ | – |
| **Total number of possible terminal nodes (generated by state)** | | | | | $2.86\times10^{242}$ | | | |
| **Average state space cardinality (over all time periods)** | | | | | $7.64\times10^{42}$ | | | |

Note: The cardinality calculations represent the cardinality of the state attributable to firm i facilities only. The cardinality attributable to firm -i facilities is the same that attributable to firm i facilities. The total cardinality of the state is the product of the cardinality attributable to firm i facilities, firm -i facilities, and the population variable.

**Constraints on firm location choices**. For a given firm $i$, we allow each location $l$ to accommodate up to four firm $i$ facilities at one time: one import distribution center, one food distribution center, one general merchandise distribution center, and one store of either type. Symmetrically, the competitor firm can also place up to four facilities in the same location $l$, for a maximum number of eight facilities per location. We assume that neither firm can place two of its own stores (regardless of type) in one location. This approximates actual store placement patterns by big box retailers such as Wal-Mart well for small MSA's, which usually accommodate only one own-store at a time, but less so for larger MSA's, which might contain several own-stores. One additional constraint we impose is that in each period $t$, each firm chooses regular stores prior to choosing supercenters. Since we allow only one firm $i$ store of any type per MSA, each firm's constrained set of possible supercenter locations are a function of period $t$ regular store location choices.

**Profit Specification**. For a given firm $i$, sales revenues for a store in location $l$ depend on the proximity of other firm $i$ stores and firm $-i$ stores, where $-i$ denotes the competitor firm. Note that since we allow only one store of any kind per MSA, we can refer to a store by its location, i.e. we refer to a store in location $l$ as store $l$. Let the portion of the state vector attributable to locations for food distribution centers ($f$), general merchandise distribution centers ($g$), regular stores ($r$), and supercenters ($sc$) be denoted as $\mathbf{s}_{it}^{f}$, $\mathbf{s}_{it}^{g}$,$\mathbf{s}_{it}^{r}$, and $\mathbf{s}_{it}^{sc}$, where each vector is of length $L$, and $\mathbf{s}_{it}^{q} \equiv \left(s_{1t}^{q}, ..., s_{Lt}^{q}\right)$ for $q \in \{f, g, r, sc\}$. Also denote the population for location $l$ at time $t$ as $pop_{lt}$. For store $l$ of firm $i$ at time $t$, denote food revenues as $R_{ilt}^{f}\left(\mathbf{s}_{it}^{sc}, \mathbf{s}_{-it}^{sc}, pop_{lt}\right)$ and general merchandise revenues as $R_{ilt}^{g}\left(\mathbf{s}_{it}, \mathbf{s}_{-it}, pop_{lt}\right)$, where $\mathbf{s}_{it} \equiv \mathbb{I}\left(\mathbf{s}_{it}^{r} + \mathbf{s}_{it}^{sc} > 0\right)$ with support $\mathcal{S}_{it}$, $\mathbb{I}(.)$ represents the indicator function, each element of $\mathbf{s}_{it}$ is denoted as $s_{ilt}$, food revenues are a function of the proximity of supercenter locations for both firms, general merchandise revenues are a function of the proximity of store locations of both types for both firms, and both classes of revenue are a function location-specific population $pop_{lt}$.[40] Although we do not model consumer choice explicitly, our revenue specification implies that consumers view other own-stores and competitor-stores as substitutes for any given store.[41]

We assume that revenues are a function of the parameter vector $\boldsymbol{\vartheta}_i = (\alpha_i, \delta_{i,-i}, \delta_{i,i})$ and specify total revenues for store $l$ and firm $i$ at time $t$ in the following way:

$$R_{ilt}\left(\mathbf{s}_{it}^{sc}, \mathbf{s}_{-it}^{sc}, \mathbf{s}_{it}, \mathbf{s}_{-it}, pop_{lt}; \boldsymbol{\vartheta}_i\right) = R_{ilt}^{f}\left(\mathbf{s}_{it}^{sc}, \mathbf{s}_{-it}^{sc}, pop_{lt}; \boldsymbol{\vartheta}_i\right) + R_{ilt}^{g}\left(\mathbf{s}_{it}, \mathbf{s}_{-it}, pop_{lt}; \boldsymbol{\vartheta}_i\right) \tag{6}$$

where,

[40] It is conceivable that close-proximity regular stores could cannibalize food revenues of a given supercenter store $l$ to the extent that consumers buy food incidentally while shopping for general merchandise. In that case, a nearby regular store might attract the business of these consumers, who could refrain from making the incidental food purchases they might have made at supercenter store $l$. Because we expect this effect to be small, we model food revenues as conditional only on the presence of nearby supercenters.

[41] Holmes (2011) specifies revenue in a similar way but derives consumers' store substitution patterns from demand estimates obtained using data on Wal-Mart sales.

$$R^{f}_{ilt}\left(\mathbf{s}^{sc}_{it},\mathbf{s}^{sc}_{-it},pop_{lt};\boldsymbol{\vartheta}_i\right)=$$
$$s^{sc}_{ilt}*\left[\alpha_i pop_{lt}\left(1+\delta_{i,-i}\sum_{m\neq l}\frac{s^{sc}_{-imt}}{d_{lm}}*\mathbb{I}\left\{d_{lm}\leq 60\right\}+\delta_{i,i}\sum_{m\neq l}\frac{s^{sc}_{imt}}{d_{lm}}*\mathbb{I}\left\{d_{lm}\leq 60\right\}\right)\right]$$

$$R^{g}_{ilt}\left(\mathbf{s}_{it},\mathbf{s}_{-it},pop_{lt};\boldsymbol{\vartheta}_i\right)=$$
$$s_{ilt}*\left[\alpha_i pop_{lt}\left(1+\delta_{i,-i}\sum_{m\neq l}\frac{s_{-imt}}{d_{lm}}*\mathbb{I}\left\{d_{lm}\leq 60\right\}+\delta_{i,i}\sum_{m\neq l}\frac{s_{imt}}{d_{lm}}*\mathbb{I}\left\{d_{lm}\leq 60\right\}\right)\right]$$

In this specification, both classes of revenue depend on the proximity of own-stores and competitor-stores through the terms $\delta_{i,i}\sum_{m\neq l}\frac{s^{y}_{imt}}{d_{lm}}*\mathbb{I}\left\{d_{lm}\leq 60\right\}$ and $\delta_{i,-i}\sum_{m\neq l}\frac{s^{y}_{-imt}}{d_{lm}}*\mathbb{I}\left\{d_{lm}\leq 60\right\}$ for $y\in\{sc,\oslash\}$, respectively, where $m$ indexes a location different from location $l$, and $d_{lm}$ represents the distance from location $l$ to a different location $m$. The parameters $\delta_{i,i}$ and $\delta_{i,-i}$ represent the average effect on revenues of close-proximity own-stores and competitor-stores, respectively. Since we assume that the parameters $\delta_{i,i}$ and $\delta_{i,-i}$ are negative, intuitively, these terms represent a deduction to revenues induced by own-stores or competitor-stores that are close in proximity to store $l$, since we assume that consumers view these stores as substitutes for store $l$. With respect to own-stores, this revenue substitution effect is deemed own-store "cannibalization," which is an important dimension of chain-store location decisions as documented by Holmes (2011) for the case of Wal-Mart. With respect to competitor stores, this effect reflects competition. The strength of the effect is weighted by $d_{lm}$, with stores in locations that are farther away from store $l$ having a smaller effect on revenues than those that are close by. The indicators $\mathbb{I}\left\{d_{lm}\leq 60\right\}$ take a value of 1 if location $m$ is closer than 60 miles away from location $l$, 0 otherwise, which imposes the assumption that stores located farther than 60 miles have no effect on store $l$ revenues. This assumption is slightly unrealistic, but we impose it since our sample only includes 227 MSA's in the U.S., which means there are few MSA's within, for example, a 30 mile radius of any MSA in our sample. With more MSA's, this cutoff distance can be reduced. We assume that the parameters $\delta_{i,i}$ and $\delta_{i,-i}$ are the same across revenue categories to simplify the exposition. Both types of revenue are dependent on population at time $t$, $x_{lt}$, through a common scalar parameter $\alpha_i$. Additionally, since regular stores don't sell food, $R^{f}_{ilt}=0$ for all regular stores.

As in Holmes (2011), we abstract from price variation and assume each firm sets constant prices across all own-stores and time, which is motivated by simplicity and is not necessarily far from reality for a chain-store retailer like Wal-Mart, which is known to set prices according to an every-day-low-price strategy. Denoting $\mu$ as the proportion of sales revenue that is net of the cost of goods sold (COGS), then $\mu R^e_{ilt}(.)$ represents revenues net of COGS for firm $i$, store $l$, time $t$, and revenue type $e \in \{g, f\}$.

Firms incur three types of additional costs: 1) distribution costs attributable to store sales, 2) store-level variable costs, and store-level fixed costs. In order to sell a given set of goods in time period $t$ at store $l$, as in Holmes (2011), we assume that each firm incurs distribution costs to deliver these goods from general merchandise or food distribution centers (or both for supercenters) to store $l$. In addition, we assume that firms incur distribution costs when transporting these goods from import distribution centers to either general merchandise or food distribution centers. We introduce these latter distribution costs in order to model location decisions for general merchandise and food distribution centers. Denote the distribution costs incurred by firm $i$ to sell goods from store $l$ at time $t$ as $DC_{ilt}$, which take the form: $DC_{ilt} = \varsigma d^g_{lt} + \iota d^{imp}_{lgt} + \varsigma d^f_{lt} + \iota d^{imp}_{lft}$. Here, $d^g_{lt}$ and $d^f_{lt}$ represent the distance from store $l$ to the nearest firm $i$ general merchandise distribution center or food distribution center, respectively. In our game simulation, if store $l$ is a regular store, we assume that it is supplied exclusively by the own-general merchandise distribution center in the MSA physically closest to store $l$. Similarly, if store $l$ is a supercenter, it is supplied exclusively by the own-food distribution center and own-general merchandise distribution center in the MSA('s) closest to store $l$. Further, $d^{imp}_{lgt}$ represents the distance between the general merchandise distribution center that supplies store $l$ and the nearest import distribution center, while $d^{imp}_{lft}$ represents the distance between the food distribution center that supplies store $l$ (if store $l$ is a supercenter) and the nearest import distribution center. We assume that distribution costs are a fixed proportion of these distances, captured by the parameters $\varsigma$ and $\iota$, and interpret fixed distribution costs as the costs incurred to operate a truck over the course of one delivery of goods per day, aggregated over one year. This model approximates the daily truck delivery distribution model actually employed by Wal-Mart, as documented by Holmes (2011). Finally, if store $l$ is a regular store, $\varsigma d^f_{lt} + \iota d^{imp}_{lft} = 0$ since regular stores do not sell food.

The remainder of our costs for both firms are specified almost exactly as in Holmes (2011) for the case of Wal-Mart, so we describe them succinctly and direct the interested reader to that work for additional description. Firms incur variable costs in the form of labor, land, and other costs (all costs not attributable to land or labor). Variable land costs are motivated by the store modification patterns of Wal-Mart, which frequently changes parking lot size, building size, and shelf space to accommodate changes in sales patterns. The quantity of labor, land, and other inputs needed are assumed to be a fixed proportion of total store revenues, such that for firm $i$, store $l$, and time $t$, $Labor^e_{ilt} = \nu^{Labor} R^e_{ilt}$, $Land^e_{ilt} = \nu^{Land} R^e_{ilt}$, and $Other^e_{ilt} = \nu^{Other} R^e_{ilt}$, for merchandise segment $e \in \{g, f\}$. The prices of land and labor per unit of input are represented by wages and rents specific to store $l$ at time $t$, denoted as $wage_{lt}$ and $rent_{lt}$. We collect data on rents and wages for each time period and each MSA. We define rents as the median (per-MSA) residential home value per square-foot from Zillow, and wages as the annual retail sector payroll divided by the total number of employees (per-MSA), provided by the U.S. Census County Business Patterns dataset (MSA level).[42] The price of the other input is normalized to 1. We focus only on fixed costs that vary by location, since costs that are constant across locations do not matter for the decision of where to locate stores and distribution centers. As documented by Holmes (2011), there are disadvantages for big box retailers like Wal-Mart of locating stores in urban locations, including, for example, increased non big box retailer shopping options for consumers. The fixed-cost disadvantage of locating stores in urban locations is modeled as a as a function of the population density at time $t$ of the location hosting store $l$, denoted as $Popden_{lt}$.[43] This function, $u\left(Popden_{lt}\right)$, is quadratic in logs, e.g.:

$$u\left(Popden_{lt}\right) = \omega_0 + \omega_1 \ln\left(Popden_{lt}\right) + \omega_2 \ln\left(Popden_{lt}\right)^2$$

Given this specification for revenues and costs, firm $i$ operating profits for store $l$ at time $t$ take the following form:

$$\pi_i\left(\mathbf{s}_t\right) \approx \pi_{ilt} \equiv \left[\left[\psi^g_{ilt} - \varsigma d^g_{lt} - \iota d^{imp}_{lgt}\right] + \left[\psi^f_{ilt} - \varsigma d^f_{lt} - \iota d^{imp}_{lft}\right] - u\left(Popden_{lt}\right)\right] \tag{7}$$

where,

$$\psi^e_{ilt} = \mu R^e_{ilt} - Wage_{lt} Labor^e_{ilt} - Rent_{lt} Land^e_{ilt} - Other^e_{ilt} \text{ for merchandise segment } e \in \{g, f\}$$

If store $l$ is a regular store, the profit component $\left[\psi^f_{ilt} - \varsigma d^f_{lt} - \iota d^{imp}_{lft}\right] = 0$, since regular stores sell only general merchandise. We assume that if firm $i$ operates no store in location $l$ at time $t$, then $\pi_{ilt} = 0$. Note that we use the $\approx$ notation to make clear that the we omit population density from the location-specific state described in Section 3.3 and instead only include location-specific population.

We define a discount factor $\beta$ and set it to $\beta = 0.95$. As in Holmes (2011), we define an exogenous productivity parameter $\rho$ that represents gradual increases in average sales per-store, motivated by gradual increases in average sales per-store experienced by Wal-Mart.[44] Profit parameter values for each specification are presented in Table 7.

[42] The Zillow data is available from http://www.zillow.com/, and the Census data is available from http://www.census.gov/econ/cbp/.

[43] See Section 3.2 for details on our population density definition and data source.

[44] Unlike in Holmes (2011), for simplicity of exposition, we make this productivity parameter constant over time. One source of these increases is an expansion in the variety of products offered for sale.

Table 7: Parameter Values by Specification

| Model | CWGB Specification 1 (baseline) | CWGB Specification 2 (high-urban-penalty) | CWGB Specification 3 (high-dist-cost) |
|---|---|---|---|
| Revenue parameter $\alpha^g$ | 60 | 60 | 60 |
| Revenue parameter $\alpha^f$ | 60 | 60 | 60 |
| Revenue parameter $\delta_{(i,-i)}$ | -0.5 | -0.5 | -0.5 |
| Revenue parameter $\delta_{(i,i)}$ | -0.5 | -0.5 | -0.5 |
| Distribution cost parameters $\varsigma$ | 1400 | 1400 | 2100 |
| Distribution cost parameters $\iota$ | 1400 | 1400 | 2100 |
| Input parameters $\nu^{labor}$ | 3.61 | 3.61 | 3.61 |
| Input parameters $\nu^{land}$ | $5 \times 10^{-6}$ | $5 \times 10^{-6}$ | $5 \times 10^{-6}$ |
| Input parameters $\nu^{other}$ | 0.07 | 0.07 | 0.07 |
| Urban location quadratic cost parameter $\omega_0$ | 0 | 0 | 0 |
| Urban location quadratic cost parameter $\omega_1$ | 20000 | 30000 | 20000 |
| Urban location quadratic cost parameter $\omega_2$ | 20000 | 30000 | 20000 |
| Discount factor $\beta$ | 0.95 | 0.95 | 0.95 |
| Productivity parameter $\rho$ | 1.07 | 1.07 | 1.07 |
| Markup $\mu$ | 0.24 | 0.24 | 0.24 |

## 5.3 Section 3.5 Details

Table 8 presents OLS models of the choice-specific value functions of interest using state variables selected by the corresponding boosted regression models in Table 1 and simulation data generated under the baseline specification of parameters.

Table 8: Choice-Specific Value Function Estimates, OLS Models (Baseline Specification)

| Choice-Specifc Value Function | $\widehat{V}_i(a^g_{ilt}=1)$ | $\widehat{V}_i(a^g_{ilt}=0)$ | $\widehat{V}_i(a^f_{ilt}=1)$ | $\widehat{V}_i(a^f_{ilt}=0)$ | $\widehat{V}_i(a^r_{ilt}=1)$ | $\widehat{V}_i(a^r_{ilt}=0)$ | $\widehat{V}_i(a^{sc}_{ilt}=1)$ | $\widehat{V}_i(a^{sc}_{ilt}=0)$ |
|---|---|---|---|---|---|---|---|---|
| Population | $1.70\times10^1$*** | $1.48\times10^1$*** | $1.99\times10^1$*** | $1.48\times10^1$*** | $1.35\times10^1$*** | $1.48\times10^1$*** | $2.00\times10^1$*** | $1.48\times10^1$*** |
| | $(1.15\times10^0)$ | $(1.76\times10^{-2})$ | $(8.53\times10^{-1})$ | $(1.76\times10^{-2})$ | $(5.99\times10^{-1})$ | $(1.76\times10^{-2})$ | $(5.99\times10^{-1})$ | $(1.76\times10^{-2})$ |
| Own Entry Regstore Allentown, PA | $-1.86\times10^7$ | | $-1.38\times10^7$ | | $-2.43\times10^6$ | | $-7.22\times10^6$ | |
| | $(2.37\times10^7)$ | | $(1.26\times10^7)$ | | $(2.38\times10^6)$ | | $(6.51\times10^6)$ | |
| Own Entry Regstore Boulder, CO | $-7.26\times10^6$ | | $-6.26\times10^6$ | | | | $-3.52\times10^6$ | |
| | $(1.91\times10^7)$ | | $(1.02\times10^7)$ | | | | $(5.26\times10^6)$ | |
| Own Entry Regstore Hartford, CT | $-9.72\times10^6$ | | $-6.68\times10^6$ | | $-1.75\times10^6$ | | $-3.73\times10^6$ | |
| | $(2.08\times10^7)$ | | $(1.08\times10^7)$ | | $(1.59\times10^6)$ | | $(5.56\times10^6)$ | |
| Own Entry Regstore Kansas City, MO | $-1.33\times10^7$ | | $-6.06\times10^6$ | | $-1.89\times10^6$ | | $-3.88\times10^6$ | |
| | $(2.37\times10^7)$ | | $(1.14\times10^7)$ | | $(2.11\times10^6)$ | | $(5.85\times10^6)$ | |
| Own Entry Regstore San Francisco, CA | NA | | | | NA | | NA | |
| | NA | | | | NA | | NA | |
| Own Entry Regstore Augusta, GA | | | $-1.91\times10^7$ | | $-3.10\times10^6$ | | $-9.38\times10^6$ | |
| | | | $(1.51\times10^7)$ | | $(2.77\times10^6)$ | | $(7.76\times10^6)$ | |
| Rival Entry Regstore Albany, GA | | NA | | NA | | NA | | NA |
| | | NA | | NA | | NA | | NA |
| Rival Entry GM Dist Clarksville, TN | | $-1.44\times10^6$*** | | $-1.44\times10^6$*** | | $-1.44\times10^6$*** | | $-1.44\times10^6$*** |
| | | $(1.03\times10^5)$ | | $(1.03\times10^5)$ | | $(1.03\times10^5)$ | | $(1.03\times10^5)$ |
| Rival Entry GM Dist Columbia, MO | | $-5.54\times10^5$*** | | $-5.50\times10^5$*** | | $-5.51\times10^5$*** | | $-5.50\times10^5$*** |
| | | $(1.03\times10^5)$ | | $(1.03\times10^5)$ | | $(1.03\times10^5)$ | | $(1.03\times10^5)$ |
| Rival Entry GM Dist Cumberland, MD | | $-2.35\times10^6$*** | | $-2.35\times10^6$*** | | $-2.35\times10^6$*** | | $-2.35\times10^6$*** |
| | | $(1.03\times10^5)$ | | $(1.03\times10^5)$ | | $(1.03\times10^5)$ | | $(1.03\times10^5)$ |
| Rival Entry GM Dist Dover, DE | | $-1.98\times10^6$*** | | $-1.98\times10^6$*** | | $-1.98\times10^6$*** | | $-1.98\times10^6$*** |
| | | $(1.03\times10^5)$ | | $(1.03\times10^5)$ | | $(1.03\times10^5)$ | | $(1.03\times10^5)$ |
| Rival Entry GM Dist Hickory, NC | | $-1.03\times10^6$*** | | $-1.03\times10^6$*** | | $-1.03\times10^6$*** | | $-1.03\times10^6$*** |
| | | $(1.03\times10^5)$ | | $(1.03\times10^5)$ | | $(1.03\times10^5)$ | | $(1.03\times10^5)$ |
| Constant | $2.77\times10^7$. | $6.36\times10^3$ | $3.25\times10^7$** | $5.08\times10^3$ | $2.27\times10^6$ | $4.31\times10^3$ | $1.48\times10^7$* | $5.14\times10^3$ |
| | $(1.58\times10^7)$ | $(7.43\times10^4)$ | $(1.18\times10^7)$ | $(7.43\times10^4)$ | $(2.22\times10^6)$ | $(7.43\times10^4)$ | $(6.04\times10^6)$ | $(7.43\times10^4)$ |

Note: Each column represents an OLS regression model of the indicated choice-specific value function run only on the state variables selected by the corrresponding boosted regression model from Table 1. Standard errors are shown in parentheses. Significance levels are indicated by: '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1. Results are based on 1000 simulation runs. Some OLS variables return "NA" for variables that do not vary. However, since the boosted model uses no intercept in its component individual univariate regressions, it returns a coefficient under these circumstances. The covariate *Own Entry Regstore City, State* represents own-firm regular store entry in the listed MSA; similarly, *Rival Entry Regstore City, State* represents competitor regular store entry in the listed MSA, and *Rival Entry GM Dist City, State* represents competitor general merchandise distribution center entry in the listed MSA.