Combinatorial Discrete Choice with Deep Reinforcement Learning

Adrian Kulesza

Harvard University

November 2024

Abstract

I utilize new computational methods to study how economic agents solve combinatorial optimization (CO) problems, where an optimal solution is selected from a large, discrete set. CO problems are ubiquitous in structural trade and spatial models, from firms deciding where to source inputs and open plants to social planners determining where to allocate infrastructure and enact policies. I use a machine learning model to approximate policy functions that learn to solve CO problems through repeated interaction with a simulated economic environment. I benchmark this approach to existing algorithms for several CO problems from trade, often yielding either optimal or superior policies with competitive computational times. I then demonstrate how this method can be applied to models with rich interdependencies, for which current methods do not work. I estimate a model of export market entry with complementarity in fixed costs and substitutability through increasing marginal costs.

I thank Pol Antràs and Marc Melitz for detailed feedback and guidance. I benefited from discussion with Costas Arkolakis, Adrien Bilal, Jake Carlson, Juanma Castro-Vincenzi, Melissa Dell, Elhanan Helpman, Mahdi Kahou, Andrew Kao, Gabriel Kreindler, and Elie Tamer at various stages of this project. This research was supported by the Molly and Dominic Ferrante Economics Research Fund.

1 Introduction

Combinatorial Optimization (CO) problems involve making an optimal combination of discrete choices. In trade and spatial settings, CO problems are pervasive from firms choosing where to source inputs, to multinationals selecting plant locations, to social planners choosing how to allocate infrastructure.[1] Solving such problems has relied on using either heuristics or specialized algorithms enabled by model assumptions and precise parameterizations. This project uses advancements from machine learning to learn how to solve otherwise non-tractable CO problems. I use these methods to identify optimal solutions in a wide variety of CO problems, oftentimes outperforming existing techniques both in terms of solution quality and computational time. I then show how to estimate models of combinatorial choice with no restrictions on the complementarity or substitutability between choices, a crux of previous work.

CO problems present themselves when an agent makes a combination of discrete choices to maximize an objective, and any one choice affects the payoffs from the other choices. To use Jia (2008) as an example, when Walmart evaluates the set of markets to enter it will not consider just the unilateral payoff from opening each store. Since nearby stores can share the same distribution infrastructure or cannibalize sales from one another, the payoff from entering one market depends on the set of other markets serviced. With N possible markets, this interdependence turns an optimization problem over N binary choices into one over 2^N subsets of choices. This quickly becomes an intractable problem as N grows, but the economics literature has continued down a handful of routes.

A first strand of the literature proceeds by considering problems where choices are always either complements or substitutes. This enables the use of algorithms that bound the set of possible choices to a set smaller than 2^N, oftentimes uniquely identifying a solution. Jia (2008), Antràs et al. (2017), Alfaro-Urena et al. (2023), and Arkolakis et al. (2023) develop and use such algorithms to tackle problems from input sourcing to export market entry.[2] A second strand relaxes the discrete nature of the problem, instead allowing firms to make continuous decisions. When combined with necessary theoretical restrictions, this opens the door for calculus-based optimization techniques. Along these lines, Castro-Vincenzi (2022) considers a multinational's problem of allocating continuous capacity across a set of countries and Oberfield et al. (2024) considers a firm's choice over a density of plants across space.[3] Third, there are sampling-based algorithms that converge to an optimum by repeatedly perturbing the solution and probabilistically accepting/rejecting changes. Kreindler et al. (2023) and Houde et al. (2023) apply these methods to optimizing transportation and logistics networks. Fourth, some CO problems can be restated as integer linear programs when combined

[1] See Antràs et al. (2017), Tintelnot (2017), and Kreindler et al. (2023) as respective examples.

[2] In cases where N is small, computing an optimum by evaluating every possibility is feasible. Tintelnot (2017) uses this approach to study plant location choices for German multinationals, limiting the choice set to 12 European and North American countries.

[3] The former paper requires certain convexity and concavity properties on the objective function and the latter studies a limiting case where plants have local spillovers only.

with appropriate constraints - enabling the use of specialized commercial optimization software (e.g. Gurobi Optimization, 2024). Head et al. (2024) and de Gortari (2020) use linear formulations of CO problems to tackle multi-stage production problems.

The goal of this paper is to place as few restrictions on the structure of the problem and produce optimal or near-optimal solutions. Additionally, since I am interested in estimation where the CO problems of many agents are repeatedly solved over varying parameters values, I require a solution method which is fast. These criteria generally eliminate the first, second, and fourth approaches as they rely on specific problem features not always available. Choices may not always be strictly complements or substitutes, may not be accurately approximated by a continuous choice, and will not always return linear payoffs. This also eliminates the third approach as it relies on converging to an optimal choice set after sampling and evaluating a larger number of possibilities. The time required to compute new solutions with this approach precludes estimation.

As a solution, this paper proposes a policy function that learns optimal decisions by interacting with a simulated economic environment. The policy function is a flexible, parametric rule that maps payoff-relevant variables to optimal choices. The parameters of this rule are optimized through reinforcement learning, where the rule is trained on simulated problem instances. In instances when the rule performs well, its parameters are updated to encourage similar decisions. By simulating thousands of problems and repeatedly updating parameters, the policy function converges to one that reliably predicts high-quality solutions to the CO problem at hand. The policy function can be used both to solve problems where parameters are fixed, and to estimate parameters where optimal choices are matched to those observed in data.

Machine Learning Literature I build on a set of papers in the machine learning literature that use flexible models to approximate value and policy functions for canonical CO problems from operations research. My main point of reference is Kool et al. (2019), but the general strategy of approximating policy function through reinforcement learning begins with Sutton et al. (1999). Even more broadly, the application of machine learning models to combinatorial optimization problems dates back to Hopfield and Tank (1985). More recently, researchers have developed competitive solvers by combining deep neural networks (i.e. large-scale machine learning models) and reinforcement learning - Bello et al. (2017) and Dai et al. (2017) are the earliest works here. For a more exhaustive review of the literature see Cappart et al. (2021) and Mazyavkina et al. (2020).

The literature I draw on is largely focused on problems like the Traveling Salesman Problem or the Capacitated Vehicle Routing Problem, namely problems featuring linear objective functions with decades of commercial development into algorithms that can solve them to global optimality. The objective of the referenced papers is largely to compete against these solvers, on which there has been mixed success. However, in economics, we often have models with rich nonlinearities that vary from paper to paper in such a way that make it difficult to benchmark algorithms and collectively develop tools. I additionally have the challenge of estimation; how can one use observed

combinatorial choices to back out economically meaningful structure. I contribute to this literature by first developing new model components that improve performance, extending the tools to non-linear models with richer notions of complementarity and substitutability, and demonstrating how these models can be used for the purpose of estimation.

More broadly, the application of machine learning models to approximate rich functions in economics is increasingly common. For two examples,Fernández-Villaverde et al. (2023) use neural networks to approximate the law of motion of non-linear state variables in a model of household savings. Hodgson and Lewis (2023) use neural networks to approximate a value function for consumer search with the goal of extrapolating to unseen states. In terms of CO problems in economics , a contemporaneous paper, Huang and Yu (2024), also experiments with techniques from machine learning. My work differs in several ways. First, the authors rely on restating any static CO problems as dynamic ones where choices are stochastically spread out over time, and they propose approximating the dynamic value function with a machine learning model. I instead approximate a policy function which is directly applicable to the static or dynamic CO problems commonly found in the economic literature. Second, I use different model components specifically designed for CO problems featuring original components that I find necessary for optimal performance on most of my problems. The authors use a standard feed-forward neural network and do not report any systematic optimality gaps, compute times, or learning curves for their model.[4] Finally, I provide multiple estimation strategies and evaluate them on a new, otherwise intractable, model with data on firm-level decisions.

Outline I start with a high-level introduction to the methodology, using a model of export market entry as a running example throughout the paper. Here, firms decide on a set of countries to export to, where each choice generates a revenue and incurs a fixed cost. Exporting to two countries can be either substitutes, due to increasing marginal costs, or complements, due to interdependence in fixed costs. I then posit a parametric policy function that decides what choices to make as a function of each country's characteristics, e.g. revenues potential or distance, and structural parameters. This changes the optimization problem from one over discrete choices, to one over parameters of a policy function that makes those discrete choices for us. The existence of rich function approximations from machine learning and optimization routines from reinforcement learning make this a viable solution strategy.

I then introduce four problems of combinatorial choice from the trade literate. The starting point is the input sourcing and plant location problems of Antràs et al. (2017) and Tintelnot (2017). These share the feature that all choices must be complements or substitutes, never a mix of both. This restriction enables the use of bounding algorithms (developed by Jia 2008 and Arkolakis et al. 2023) which can reliably solve these problems to their global optimum. The second problem

[4]The authors report using 7 layers and 256 or 512 nodes in each layer, this is roughly 3 orders of magnitude smaller than the models I use. I find the model size, architecture, and novel components I contribute necessary for the level of performance I report.

I consider extends the input sourcing model of Antràs et al. (2017), allowing for oligopolistic firms that compete in final goods markets not just with each-other, but with an outside sector. If the elasticity of substitution between firms is greater than that across sectors, then sourcing choices can be complementary at low market shares and substitutable at high market shares.

The third problem is one of global value chains with scale economics. I start with a generalized version of the sequential production model of Antràs and Gortari (2020), where I allow for non-unitary substitution elasticities between stages as well as scale economics. The combinatorial choice is over a sequence of locations corresponding to each stage of production, where a fixed cost needs to be incurred when setting up each stage. I show how such a problem is amenable to a new dynamic programming algorithm. The fourth problem is an extension of the export market entry model used in the introductory example. The model is a static version of Alfaro-Urena et al. (2023), where I break the supermodularity property by introducing a competing force for substitutability.

I continue by formalizing each problem as a Markov decision process with general notation, and stating the generic optimization problem of finding an optimal policy. Next, I introduce the principle of function approximation for the optimal policy. Here I approximate the policy function with a lower-dimensional parametric function. The new objective is then to find the parameters of this function that generate the best policies, as determined by the economic objective function. I conclude the section with discussing the optimization algorithm for fitting the policy function.

What makes function approximation a viable strategy for combinatorial optimization problems is the availability of machine learning models capable of capturing rich interdependence between discrete choices. The two key features of these models is their ability to mix information about various choices and dynamically update what information is considered as a solution is constructed. I walk through an intuitive explanation with the export market entry model as an example.

The optimized policy functions return (near) optimal policies on all problems I consider. For cases where an optimal benchmark is not available, policies are superior to those produced from other algorithms. For the input sourcing and plant location problem solutions are within 0.04% of the optimum for cases with 20 and 50 locations. For the oligopoly input sourcing problem, solutions are within 0.09% for problems with 20 locations. For the 50 and 100 location version I benchmark to a greedy search; solutions are 120% and 46% better in the 50 and 100 location case respectively. For the global value chain problem with fixed costs, I benchmark to a dynamic programming solution which approximates an optimum; solutions are 1% better in the 20, 50, and 100 location cases. For the export market entry problem I later estimate, solutions are within 0.03% of the optimum for the 20 location case and 11% better than a greedy algorithm in the 50 location case.

The next dimension of performance I consider is time; policy functions are quick to optimize and afterwards produce optimal solutions in a relatively short amount of time. Optimization takes on the order of magnitude of minutes, with models normally requiring fewer than 500,000 simulated problems to converge with optimization usually taking less than 11 seconds per 10,000 instances to complete. Producing optimal policies for the export-market entry model I next estimate takes 1

second for 1,000 optimal policies.

The policy function is optimized over a pre-specified distribution of structural parameters. The corresponding estimation problem takes observed combinatorial choices, and asks what structural parameters generate policies consistent with those observed. I sketch a simulated-method of moments procedure as well as a Bayesian and quasi-Bayesian procedure that performs the function approximation and estimation simultaneously. Estimation raises an important question of approximation error. Since the methodology I propose has no guarantees on optimality, the previous section serves to alleviate concerns by demonstrating (near) optimality on a variety of problems.

Finally, I estimate a model of export market entry with substitutability from increasing marginal costs and complementarity from interdependence in fixed costs. The model is a static version of Alfaro-Urena et al. (2023), augmented by decreasing returns to scale in firms' production technology. The inclusion of decreasing returns to scale, consistent with existing evidence on firm exporting (Almunia et al. 2021), produces a better model fit and results in smaller estimated fixed costs. I demonstrate the importance of the new estimates for counterfactuals: a model with only complementarities is not able to generate reallocation to other export markets when negative shocks hit the current export choices. I highlight how this leads to qualitatively different export responses to trade shocks.

I conclude with a discussion of ongoing work and directions for future research. In progress is the estimation of a firm-level characteristics demand system for imports. Firms here have heterogeneous preferences over characteristics which describe each country, and must make both an intensive and extensive margin choice of where to import from. As far as future work goes, the methodology is a general framework for optimization. The function approximation I use and the specific optimization algorithm are just one of many possible options. Future work will be able to draw on new innovations from machine learning and likely achieve better performance. Otherwise, I only consider partial equilibrium static models in this paper. A natural extension of these tools is to general equilibrium or dynamic settings.

2 Approximating and Learning Policy Functions

This section introduces the methodology at a high level with an example to build intuition. Consider a firm choosing a subset of N potential countries to export to. The firm's choice set can be represented by a set of indicator variables $\{\mathbb{1}_i\}_{i \in N}$, which take a value of 1 when exporting to the corresponding country. Exporting generates a fixed revenue from each country, T_i. And, increasing marginal costs make profits a concave function of total revenue, $(\sum_{i=1}^{N} \mathbb{1}_i T_i)^{\beta}$ where $\beta \in (0, 1)$. Exporting to a country also requires a fixed cost payment linear in distance, αd_{hi} where d_{hi} is the distance from the firm's location to the country i. But, complementarities in fixed cost make it cheaper to export to multiple nearby markets. The fixed costs of exporting to i decreases by a factor of $\gamma \sum_{j \neq i}^{N} \mathbb{1}_j d_{ji}^{-1}$, where d_{ji} is the distance between countries j and i. The firm's profit from

a set of choices is given by

$$V\left(\{\mathbb{1}_i\}, s\right) \equiv \left(\sum_{i=1}^{N} \mathbb{1}_i T_i\right)^{\beta} - \sum_{i=1}^{N} \mathbb{1}_i \left(\alpha d_{hi} - \gamma \sum_{j \neq i}^{N} \mathbb{1}_j d_{ji}^{-1}\right) \tag{1}$$

where $s = \{\{T_i, d_{hi}, \{d_{ji}\}\}, \beta, \alpha, \gamma\}$ indexes the problem state.[5]

If $\beta = 1$, then the firm's profit function would only feature complementarity between choices. Conversely, if $\gamma = 0$, then it would only feature substitutability between choices. Jia (2008) and Arkolakis et al. (2023) provide algorithms for solving these cases, but there is no general strategy for solving a combinatorial choice model that features both complements and substitutes.

This project does not directly provide an algorithm for solving any particular combinatorial choice problem. Instead, inspired by work from the intersection of machine learning and operations research, I first write a parametric policy function that decides what choices to make as a function of the problem state. And second, I use a learning algorithm to tune the policy function to maximize a problem-specific profit function like equation (1). This changes the optimization problem from one over discrete choices, to one over parameters of a mapping from states of the problem to indicators.

Suppose, as a simple example, the policy function assigned a probability for exporting to each country:

$$p\left(\mathbb{1}_i = 1 | s; \Theta\right) = \sigma\left(T_i\theta_1 + \beta\theta_2 + d_{hi}\theta_3 + \alpha\theta_4 + \sum_{j \neq i} d_{ji}^{-1}\theta_{5,j} + \gamma\theta_6\right) \tag{2}$$

where s again indexes the problem state, $\Theta = \{\theta_k\}$ collects a set of input weights, and σ is a function that ensures a probability in $(0, 1)$. The goal is to find the weights Θ that maximize the expected profit from applying this policy function to equation (2) over a distribution of problem instances $F(s)$, pre-specified by the researcher.[6] Formally, I am solving

$$\max_{\Theta} \mathbb{E}_{s \sim F(s)} \left[\mathbb{E}_{\{\mathbb{1}_i \sim p(\mathbb{1}_i | s; \Theta)\}} \left[V\left(\{\mathbb{1}_i\}, s\right)\right]\right] \tag{3}$$

where $V\left(\{\mathbb{1}_i\}, s\right)$, given by equation (1), denotes the value of the firm's choice. At an intuitive level, one might expect a positive weight on T_i ($\theta_1 > 0$) and a negative weight on d_{hi} ($\theta_3 < 0$) - choices with a greater revenue net of cost should be more likely to be included in the choice set. Additionally, since as γ gets larger, complementarity effects between choices make each more likely to be included, one could expect a positive associated weight ($\theta_6 > 0$). One could continue reasoning

[5] Antràs et al. (2017) and Tintelnot (2017) are nested in this equation as cases where $\gamma = 0$ and $\beta > 1$ or $\beta < 1$, albeit with different microfoundations and parameterizations of fixed costs. Alfaro-Urena et al. (2023) and Antràs et al. (2023) are similar as well, with richer structure for the interdependencies between fixed costs.

[6] The distribution $F(s)$ is the relevant distributions of structural parameters and payoff relevant variables that one wants the policy function to perform over. The benefit of maximizing unconditional on a particular s, is that I can use the resulting policy function to solve the problems of heterogeneous agents (where $\{T_i, d_{hi}\}$ varies) and to estimate over structural parameters (where $\{\beta, \alpha, \gamma\}$ vary).

for each weight, crafting an algorithm by heuristically assigning a value to each input based on their understanding of equation 1.

Instead, to find the optimal set of weights, I use an optimization paradigm known as reinforcement learning. Imagine I started with $\theta_k = 0 \; \forall k$ so that each choice is equally likely and I evaluated the objective (3). Now suppose I raised θ_1 slightly. This would make countries with higher revenue, T_i, more likely to be selected for exporting, thereby raising the objective value.[7] I can do this repeatedly for each input weight, evaluating perturbations to the weights and adjusting based on the change in the objective value. I may then converge to set of input weights Θ that are in expectation maximizing the objective subject to the functional form assumption of equation (2).[8]

The problem with the approach described above is that equation (2) is likely not flexible or informed enough to make (near) optimal choices. I could go down the route of experimenting with additional input variables, higher-order polynomials of s, interactions between the features of each choice, or other nonlinear functions of s. I opt to instead use a machine learning model known as a *Transformer* for the approximation.[9] Key to its success, the transformer is structured to learn how to optimally mix information streams and understand the context of a problem.[10] For instance, in visual processing tasks one combines pixel-level data to learn higher-order structure about an image, and in language tasks one merges the information embedded in words to uncover latent structure about a sentence. The same principle applies here. Combinatorial problems are characterized by the interdependence between discrete choices, so I naturally use a model that is capable of learning how to best mix information about each choice.

The exact details regarding how a Transformer is able to accomplish this are in Section 5. But one can roughly imagine a policy function as follows:

$$p\left(\mathbb{1}_i = 1 | s; \Theta\right) = \sigma\left(T_i\theta_1\left(s\right) + \beta\theta_2\left(s\right) + d_{hi}\theta_3\left(s\right) + \alpha\theta_4\left(s\right) + \sum_{j \neq i} d_{ji}^{-1}\theta_{5,j}\left(s\right) + \gamma\theta_6\left(s\right)\right),$$

where the weights are dynamically allowed to depend on other features of the problem s. For example, when γ is larger, centrally located countries generate stronger complementarities. One

[7] This example assumes that the features of s are uncorrelated such that choices with high values of T_i are not otherwise systematically different.

[8] There exists global optimum convergence proofs for standard reinforcement learning settings - among the most notable, Watkins and Dayan (1992) proves convergence for a standard Q-learning environment. Unfortunately, I can not make any formal statement in this environment about convergence to a global optimum. The best I can do is reference Sutton et al. (1999), who prove convergence to a local optimum when using a differentiable approximation. A goal of Section 6 is to numerically verify how close I am to optimality or how I compare to alternate approximate/heuristic methods.

[9] Many machine learning models have been used to approximate policy/value function for CO problems - the earliest works Dai et al. (2017) and Bello et al. (2017) use a graph encoder and a recurrent neural network. An earlier version of this project experimented with the Dai et al. (2017) model, on which there was mixed success. The literature has largely pivoted to using variants of Transformers.

[10] Understanding the details of machine learning model performance is an active area of research. Recent work, Tolstikhin et al. (2021) and Yu et al. (2022), has highlighted that information mixing among input data is a key feature of their success.

might want to put more weight on $\sum_{j \neq i} d_{ji}^{-1}$ and allow each d_{ji} to be weighted according to the corresponding revenue, T_j. Or if β was small, large substitution effects imply that choices with high unilateral profit would be desirable. One might then want the weight on T_i and d_{hi} to reflect i's value relative to the other choices. The Transformer is capable of generating such information mixing schemes, and it will learn one appropriate for maximizing equation (1).

To summarize, I am not providing an algorithm tailored to an exact CO problem. Rather, I am proposing a flexible, parametric function for selecting choices and an optimization routine for tuning its parameters to return objective-maximizing solutions over a distribution of problems. No data on optimal actions is required. I use problems from a pre-specified distribution, use policies from a parametric policy function, evaluate the quality of those policies with an objective function, and update the policy function to return increasingly better policies. The detailed optimality results for an export market entry problem are presented in Section 8 and estimation results using firm-level export data are in Section 8. But to preview, the policy function approximation produces objective values that are on average within 0.01% of the optimum.[11]

3 Economic Models of Combinatorial Choice

This section introduces the structural models considered and their corresponding combinatorial optimization problems. I focus on models from the international trade literature, where the multinational environment naturally lends itself to such problems. First, I consider the input sourcing and plant location problems of Antràs et al. (2017) and Tintelnot (2017). The objective functions in these problems are either supermodular or submodular, admitting the use of powerful bounding algorithms that ease optimization. I use these models as a way to benchmark this paper's methodology against known algorithms. Second, I present an oligopoly version of Antràs et al. (2017), where changing demand elasticities create complements at low market shares and substitutes at high market shares. This change prevents the use of existing bounding algorithms. Third, I introduce a global value chain problem with scale economies that involves choosing an optimal sequence of J production stages across N potential locations. Alongside, I develop a dynamic programming approach to compute optima for reference. Fourth, I present a static version of the export market interdependence model of Alfaro-Urena et al. (2023), augmented with increasing marginal costs which generates a force for substitutability. I estimate the model in Section 8 with firm-level data on export decisions from the World Bank Exporter Dynamics Database (Fernandes et al. 2016). Fifth, I preview the subject of followup work; a firm-level characteristics demand system for imports with an extensive and intensive margin.

[11] To evaluate solution methods I use a measure called the optimality gap, which can roughly be interpreted as the average percent difference between the objective values returned by two separate algorithms. See equation (18) for a description of this performance metric.

3.1 Input Sourcing and Plant Location

A subset of N potential choices is selected to solve

$$\max_{\{\mathbb{1}_i\}\in 2^N} \left(\sum_{i=1}^{N} \mathbb{1}_i T_i\right)^{\alpha} - \sum_{i=1}^{N} \mathbb{1}_i f_i, \tag{4}$$

where I refer to $T_i > 0$ as the return to a choice and $f_i > 0$ as the cost of a choice. The parameter $\alpha > 0$ governs the interaction between choices. When $\alpha > 1$, the objective (4) is supermodular: as more choices are included in the solution, the contribution of another choice increases. Conversely, when $\alpha < 1$, the objective (4) is submodular: the contribution of another choice decreases as more choices are added to the solution. Two problems that fit this framework are the input sourcing problem of Antràs et al. (2017) and the plant location problem of Tintelnot (2017). In Antràs et al. (2017) the choice is over what locations to source from and in Tintelnot (2017) the choice is over what locations to open production sites in. In both papers, T_i is associated with some marginal cost reduction (either from combining inputs across countries or outputs across plants) and f_i is the associated fixed cost. In the former case, $\alpha > 1$ which allows the authors to implement the algorithm from Jia (2008) to bound the optimal solution. In the later case, $\alpha < 1$ and the solution method involved limiting N to 12, allowing a direct search for the optimum by evaluating all 2^N possibilities. Recently, Arkolakis et al. (2023) have provided an algorithm for the $\alpha < 1$ that produces bounds for the optimal solution.

The algorithms used to solve either version of the objective (4) proceed by first producing bounds that contain elements which can either never be in the solution or must be in the solution. If these bounds do not uniquely identify a solution, then a search needs to be done among the remaining choices. Arkolakis et al. (2023) additionally develop a branching procedure for speeding up this search. Ultimately, the bounding procedures are fast and reliable, often pinning down a unique solution with no additional computation required. They leverage the strong theoretical assumptions imposed in (4) to their benefit. The methods I am proposing today are geared for richer models, but benchmarking performance to bounding algorithms will serve as a point of reference.

3.2 Oligopoly in Input Sourcing and Plant Location

Solving the previous model relies on a theoretical assumption: choices must always be either complements or substitutes. We are beholden to this assumption since without it, we do not have a disciplined way to reduce or search the solution space. To explore one deviation where this assumption does not hold, I will embed the above framework into the market structure of Helpman and Niswonger (2020).[12]

Consider an industry populated by one oligopolistic producer, k, and a continuum of price-taking

[12]The discussion here is applicable to the Atkeson and Burstein (2008) market structure, but I focus on a market structure where only one firm is making a combinatorial choice to simplify later computation.

firms with constant marginal cost. Each firm produces a unique variety. The residual demand curve faced by a producer k is

$$x_k = P^{\delta} p_k^{-\sigma} \tag{5}$$

where $P^{1-\sigma} = \bar{p}^{1-\sigma} + p_k^{1-\sigma}$ and $\delta = \sigma - \varepsilon$. The parameter σ is the elasticity of substitution across varieties in the industry, while ε determines the degree of substitution between the CES aggregate of varieties and the outside economy. The fixed price of other varieties is given by $\bar{p}$. Facing a marginal cost of c_k, the optimal price set by producer k is

$$p_k = \frac{\sigma - \delta s_k}{\sigma - \delta s_k - 1} c_k; \text{ where } s_k = \frac{p_k^{1-\sigma}}{\bar{p}^{1-\sigma} + p_k^{1-\sigma}}. \tag{6}$$

Now assuming that marginal costs are determined by an input sourcing process where $c_k = \left(\sum_{i=1}^{N} \mathbb{1}_i T_i\right)^{-1/\theta}$ and f_i are the associated fixed costs, the optimization problem of firm k is

$$\max_{\{\mathbb{1}_i\} \in 2^N} \left(\frac{\sigma - \delta s_k}{\sigma - \delta s_k - 1} - 1 \right) \left(\sum_{i=1}^{N} \mathbb{1}_i T_i \right)^{-1/\theta} x_k - \sum_{i=1}^{N} \mathbb{1}_i f_i \tag{7}$$

subject to equations (5) and (6). A micro-foundation for this cost-function is provided by Antràs et al. (2017) where T_i governs the cost of goods from location i and θ is the Frechet scale parameter.

If $\sigma - 1 > \theta$ and $\varepsilon - 1 < \theta$, then (7) is neither supermodular nor submodular for all s_k.[13] Intuitively, when the oligopolist's market share is near 0, any marginal change in costs will not affect the aggregate price index P. Cost reductions at this point induce changes in relative demand between the varieties, which is governed by the elasticity of substitution σ. When the firm's market share is near 1, then the aggregate price index is effectively the firm's price. Cost reductions at this point induce changes in the absolute level of demand, which is governed by the elasticity parameter ε. Since $\varepsilon < \sigma$, this results in the (7) turning from a supermodular to a submodular function. The example shows how fragile the interdependence properties of a model can be: small deviations can result in intractable numerical environments.

3.3 Global Value Chains

The previous two sections presented models that require optimizing over a subset of possible choices. The methods proposed today can also be applied to combinatorial optimization problems where the relevant object is a sequence of discrete choices. A familiar case in the trade literate is Global Value Chains (GVCs). I start with a generalization of the GVC model described in Antràs and Gortari (2020) and de Gortari (2020).

Production requires J sequentially completed stages. To produce output at stage j, a stage-

[13] See Appendix Section (D.1) for a formal statement and numerical example.

specific input X^j is combined with the output of the previous stage Y^{j-1} using CES technology:

$$Y^j = \left(\left(\alpha^j\right)^{\frac{1}{\sigma_j}} \left(X^j\right)^{\frac{\sigma_j - 1}{\sigma_j}} + \left(1-\alpha^j\right)^{\frac{1}{\sigma_j}} \left(Y^{j-1}\right)^{\frac{\sigma_j - 1}{\sigma_j}} \right)^{\frac{\sigma_j}{\sigma_j - 1}},$$

where σ_j is the stage-specific elasticity of substitution and α^j governs how intensely each input is used in production. The initial stage is characterized by linear production in the stage-specific input only. These stages can be completed in any of N different locations which each have their own cost, c_n^j, of producing stage-specific goods. Here, superscript j denotes the stage and subscript n the country. If a firm locates two subsequent stages in different countries, they incur a potentially stage-specific iceberg trade cost τ_{nk}^j.

Using the notation of Antràs and Gortari (2020), let $\ell(j)$ denote the location of stage j production and $\boldsymbol{\ell} \equiv \{\ell(1), ..., \ell(J)\}$ the entire sequence of production locations. I can recursively define the cost of production at each stage given the production sequence ℓ as follows:

$$C_{\ell(j)}^j(\ell) = \left(\alpha^j \left(c_{\ell(j)}^j\right)^{1-\sigma_j} + \left(1-\alpha^j\right) \left(C_{\ell(j-1)}^{j-1} \tau_{\ell(j-1)\ell(j)}^j\right)^{1-\sigma_j} \right)^{\frac{1}{1-\sigma_j}}, \tag{8}$$

where for the initial stage costs are $C_{\ell(1)}^1(\ell) = c_{\ell(1)}^1$. A simplified cost equation arises in the limiting case where $\sigma_j \to 1 \forall j$:

$$C_{\ell(j)}^j(\ell) = \delta \prod_{j=1}^{J} \left(c_{\ell(j)}^j\right)^{\alpha^n \beta^n} \prod_{j=1}^{J-1} \left(\tau_{\ell(j)\ell(j+1)}\right)^{\beta_n}$$

where $\beta_j = \prod_{k=j+1}^{J} (1-\alpha_k)$ and δ is scaling parameter consisting of parameters α^j and β^j. This case is the focus of Antràs and Gortari (2020) due to analytical and computationally appealing features - it can be linearized by taking the log. However, restricting the elasticity of substitution across stages to 1 may be unrealistic.[14] The first point of departure is then handling models with non-unitarity elasticities.

The second point of departure is introducing increasing returns to scale (i.e. decreasing average costs) by requiring that fixed costs be paid to locate stages of production across different countries.[15] Namely, setting up production incurs a fixed $f_{\ell(j-1)\ell(j)}^j$ - I allow this fixed cost to depend on the stage j and the locations of stages $\ell(j-1)$ and $\ell(j)$.[16] Taking total quantity Q as exogenous, the

[14] Estimates of across-stage elasticities are lacking, but Atalay (2017) and Brian C. et al. (2024) estimate an elasticity of substitution across intermediates inputs significantly less than 1. If stages of production are internal to a firm and the result of combining separate intermediate inputs in a sequential process, then these results suggest the Cobb-Douglas GVC model may be misspecified.

[15] An increasing returns to scale GVC model is the subject of de Gortari (2020), but it relies on the linearization of the Cobb-Douglas cost equation which permits Integer Linear Programming techniques.

[16] I require that fixed costs depend on nothing beyond the previous stage of production, otherwise the state of the dynamic program will have to include additional prior location choices.

cost-minimization problem is

$$\min_{\ell \in N^J} QC^J_{\ell(J)}(\ell) + \sum_{j=1}^{J} f^j_{\ell(j-1)\ell(j)}, \tag{9}$$

where $C^J_{\ell(J)}(\ell)$ is determined by equation (8). One could alternatively specify a demand system that generates Q as a function of the cost in the final stage.

3.3.1 Dynamic Programming

The GVC problem with scale economies represented by (9) can be solved with dynamic programming through backwards induction. Given the marginal cost of production through stage $J-1$, C^{J-1}, as well as the accumulated fixed costs, F^{J-1}, and the location of production for the previous stage $\ell(J-1)$, the problem faced by the firm in determining the final stage of production can be written as

$$V_J\left(C^{J-1}, F^{J-1}, \ell(J-1)\right) = \min_{\ell(J)\in N} QC^J\left(\ell(J), C^{J-1}, \ell(J-1)\right) + F^J\left(\ell(J), F^{J-1}, \ell(J-1)\right) \tag{10}$$

subject to

$$\begin{aligned} C^J\left(\ell(J), C^{J-1}, \ell(J-1)\right) =& \left(\alpha^J\left(\bar{c}^J_{\ell(J)}\right)^{1-\sigma_J} + \left(1-\alpha^J\right)\left(C^{J-1}\tau^J_{\ell(J-1)\ell(J)}\right)^{1-\sigma_J}\right)^{\frac{1}{1-\sigma_J}}, \\ F^J\left(\ell(J), F^{J-1}, \ell(J-1)\right) =& F^{J-1} + f^J_{\ell(j-1)\ell(j)}. \end{aligned} \tag{11}$$

This setup leverages the fact that stages prior to $J-1$ only affect the payoff at stage J through their contribution to the marginal costs and fixed costs aggregators, C^{J-1} and F^{J-1}. The only specific information one needs about past decisions is the previous stage's location, which determines the current stage's fixed and production costs through $f^J_{\ell(j-1)\ell(j)}$ and $\tau^J_{\ell(J-1)\ell(J)}$.

Continuing this recursion backwards, the problem then faced at stage $J-1$ can be written as

$$\begin{aligned} &V_{J-1}\left(C^{J-2}, F^{J-2}, \ell(J-2)\right) \\ &= \min_{\ell(J-1)\in N} V_J\left(C^{J-1}\left(\ell(J-1), C^{J-2}, \ell(J-2)\right), F^{J-1}\left(\ell(J-1), \ell(J-2)\right), \ell(J-1)\right) \end{aligned}$$

subject to the $J-1$ analog of equation (11). Where note that the payoffs from any decisions at this stage, $V_J(\cdot)$, were determined in the first step. One can continue in this manner until reaching the initial stage.

Searching over all possible sequences requires evaluating N^J possibilities. This dynamic program can reduce the number of computations required to solve the problem. First, note that the continuous states of equation (10), C^j and F^j, need to be discretized into N_C and N_F values. The final part of the state, the location of previous production takes one of N values. There are then $J \times N_C \times N_F \times N^2$ calculations required to find the optimal sequence.[17] For moderately sized value

[17]The state takes one of $N_C \times N_F \times N$ values. Computing the minimum requires N computations - one for each

chains with a realistic number of locations, the dynamic programming approaches requires orders of magnitude less computations.[18] This dynamic program serves as a benchmark for the computational methods I introduce since it is capable of finding a global optimum, up to discretization error.

3.4 Export Market Interdependence

Standard models of extensive-margin export decisions, e.g. Melitz (2003), treat markets as independent: the payoff from exporting to one market is not affected by the decision made for another market. But, recent work, e.g. Alfaro-Urena et al. (2023) and Antràs et al. (2023), suggests interdependence in export decisions operating through fixed cost reductions. The fixed costs of exporting to one market may depend on the set of markets exported to. Intuitively, a firm in Mexico might find it easier to export to Belgium if it already exports to France due to the shared language, proximity, or economic union of the destinations. Simultaneously, there is evidence for substitutability in exporting, e.g. Almunia et al. (2021), arising from decreasing returns to scale. For instance, if the output of a farm in Mexico is limited by the amount of land it owns, then exporting to Canada is a substitute for exporting to the US.

Formally, the problem is

$$\max_{\{\mathbb{1}_i\}\in 2^N} \left(\sum_{i=1}^{N} \mathbb{1}_i T_i \right)^{\beta} - \sum_{i=1}^{N} \mathbb{1}_i \times f_i\left(\{\mathbb{1}_i\}\right), \tag{12}$$

where T_i is a payoff parameter for entering a market i and f_i is the associated fixed cost. The parameter $\beta < 1$ governs the returns to scale and $f_i\left(\{\mathbb{1}_i\}\right)$ denotes that the fixed cost for market i depends on the set of markets chosen. I only consider complementarity in fixed costs, i.e. $f_i\left(\{\mathbb{1}_i\}\right)$ is weakly decreasing in the number of active markets. A model featuring either strict complementarity ($\beta = 1$) through shared fixed costs or strict substitutability ($f_i\left(\{\mathbb{1}_i\}\right) = f_i$) through scale effect can be solved with a bounding algorithm. Alfaro-Urena et al. (2023) do so for the complements-only case, even extending the solution methodology to a dynamic model featuring forward looking firms and sunk costs.[19] In Section (8), I solve, estimate, and produce counterfactuals from a static version of the author's model augmented with a force for substitutability.

candidate location. And one has to do this J times - once for each stage.

[18] If $N = 100$ and $J = 5$ and one discretizes the state with $N_C = N_F = 20$, then brute force requires 10^{10} computations against the $2 * 10^7$ required with the dynamic programming approach.

[19] Work by Liu (2024) extends the authors original algorithm to include both a super-modular static profit function and an endogenous unobserved state variable.

4 Reinforcement Learning

In order to solve the combinatorial problems presented in Section 3, I compute a policy function that maps structural parameters and payoff relevant variables to optimal choices. The functional form of that policy function is a machine learning model. And, the parameters of that functional form are chosen with reinforcement learning to return increasingly better policies. In this section, I introduce the general optimization problem and principle behind reinforcement learning. In the next section I detail the machine learning model.

Reinforcement learning is a broad field concerned with optimizing decision-making by agents. Problems are often formulated as Markov decision processes which enable the use of learning algorithms, dynamic programming ones like value- and policy-function iteration are the most standard. Otherwise, this structure gives a familiar setting for researchers to evaluate problem setups and apply known learning algorithms.[20] This is the same structure used by economists to set up dynamic optimization problems.

Markov decision processes are characterized by four objects: the state space $\mathcal{S}$, the action space $\mathcal{A}$, a transition kernel $T : \mathcal{S} \times \mathcal{A} \to \mathcal{S}$, and a reward function $R : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$. What I am after is a policy function $\pi : \mathcal{S} \to \mathcal{A}$ that maps the state to an optimal action. We can write a recursively-defined value-function to assign a value to each policy:

$$V_\pi(s_t) = R(s_t, \pi(s_t)) + V_\pi(T(s_t, \pi(s_t)))$$

where $s_t \in S$ is the state, $\pi(s_t) \in \mathcal{A}$ is the action prescribed by the policy function, $R(\cdot)$ is the reward associated with taking action $\pi(s_t)$ in state s_t, and $T(\cdot)$ is the new state achieved by taking action $\pi(s_t)$ in state s_t. The subscript t indexes steps in the sequential construction of a solution to a static problem, not time. Hence there is no discounting. Substituting forward for V_π, one can alternatively write

$$V_\pi(s_t) = \sum_{i=0}^{\infty} R(s_{t+i}, \pi(s_{t+i}))$$

subject to $s_{t+i+1} = T(s_{t+i}, \pi(s_{t+i}))$. The goal is to find the policy which maximizes $V_\pi(s_t)$.

As an example, I place the export-market entry problem from Section 2 in this framework. The state s_t includes a vector of indicator variables $\{\mathbb{1}_i\}$ corresponding to each choice, payoff relevant variables $\boldsymbol{x} \equiv \{T_i, d_{hi}, \{d_{ji}\}_j\}$, and structural parameters $\boldsymbol{\vartheta} \equiv \{\beta, \alpha, \gamma\}$: $s_t \equiv \{\{\mathbb{1}_i\}, \boldsymbol{x}, \boldsymbol{\vartheta}\}$. The policy function $\pi(s_t)$ prescribes an action $\pi_t \in N$ - a choice to be added to the solution set. The transition kernel T updates the state by setting the indicator associated with action π_t to a value of 1.

Let $\mathcal{V}(s_t)$ denote the objective value associated with any state as in equation 1.[21] Rewriting

[20] See Vesselinova et al. (2020) and Dai et al. (2017) for a Markov decision process formulation of canonical CO problems, including the traveling salesman problem, for a machine learning application.

[21] The notation is changed from Section 2. The state now includes the set of choices made and is indexed by t - denoting that the set of choices can change as a solution is iteratively constructed.

below,

$$\mathcal{V}(s_t) = \left(\sum_{i=1}^{N} \mathbb{1}_i(s_t) T_i\right)^{\beta} - \sum_{i=1}^{N} \mathbb{1}_i(s_t) \left(\alpha d_{hi} - \gamma \sum_{j \neq i}^{N} \mathbb{1}_j(s_t) d_{ji}^{-1}\right), \tag{13}$$

where $\mathbb{1}_i(s_t)$ makes explicit that the choices are contained within the state. The reward function is then the change in objective value associated with any action-state pair: $R(s_t, \pi_t) = \mathcal{V}(T(s_t, \pi_t)) - \mathcal{V}(s_t)$. The cumulative reward from an initial state with indicators initialized to zero, s_0, under a policy π is then given by

$$V_\pi(s_0) \equiv \sum_{t=0}^{S} R(s_t, \pi_t) = \mathcal{V}(s_S),$$

where s_S is the terminal state consisting of all actions taken $\{\pi_t\}_{t=0}^{S}$. Note, I am modeling a static problem of combinatorial choice as a sequence of discrete decisions. There are a finite number of choices that can be made, and no choice can be undone. There is then some terminal state S at which the state will remain unchanged, and all future rewards will be zero.[22]

Given an initial state s_0, the optimization problem is to find the policy π^* that solves

$$\max_{\pi} V_\pi(s_0).$$

This structure generalizes the problems represented by (4), (7), (9), and (12). The standard solution methods for such problems are iterative grid-based methods which sequentially fill in values of $V_{\pi^*}(s_t)$ starting from terminal or initial states and working backwards or forwards. The bounding algorithms referenced with Section 3.1 and the dynamic programming algorithm presented in Section 3.3.1 exploit properties of the objective function to do this without computing all 2^N or N^J possibilities. But the algorithms crucially hold some of the non-choice variables $\{\boldsymbol{x}, \boldsymbol{\alpha}\}$ fixed.[23] There is no general strategy for efficiently computing $V_{\pi^*}(s_t)$ in combinatorial optimization problems, nor is there an obvious way to include structural parameters in the state without proportionally increasing its size.

4.1 Function Approximation

A tabular policy function $\pi : \mathcal{S} \to A$ has as many "parameters" as states: it stores one action for each state. Given how many combinations of choices and structural parameters there are, this is too large a "parameter" space to optimize over directly. I choose to instead approximate the policy function with a lower-dimensional parametric function.[24] Specifically, I use a stochastic policy

[22] In practice, since choices can be repeated and none can be undone, the terminal state appears after N iterations of the state. This gives an opportunity for each choice to be selected, and no additional iterations are required since nothing can then be added or removed.

[23] Alfaro-Urena et al. (2023) and Arkolakis et al. (2023) provide methods for computing policy functions over some non-choice features of their states, but not structural parameters.

[24] There is an alternative approach that instead approximates an action-value function described in Appendix B.4.

function

$$p\left(\pi|\boldsymbol{s};\Theta\right),$$

which prescribes a probability for each policy $\pi = \{\pi_t\}$ (i.e. sequence of actions) given a state $\boldsymbol{s}$ containing the structural parameters and payoff-relevant variables. The parameters of the function are indexed by Θ. Since the function I fit is an approximation of the true function it will be subject to an approximation error. Generally, there will always exists a tradeoff between computational complexity and approximation error, but Section 6 numerically verifies that this is negligible in my applications.[25]

The new optimization problem is to find the parameters of a parametric function, that solves

$$\max_{\Theta} \mathbb{E}_{\boldsymbol{s}\sim F(\boldsymbol{s})}\Big[\mathbb{E}_{\pi\sim p(\pi|\boldsymbol{s};\Theta)}\left[V_{\pi}\left(\boldsymbol{s}\right) - b\left(\boldsymbol{s}\right)\right]\Big]. \tag{14}$$

The inner expectation is over the distribution of policies implied by the stochastic policy function in a state $\boldsymbol{s}$. The outer expectation is over the distribution of states that I am interested in, $F\left(\boldsymbol{s}\right)$. This distribution of states is pre-specified by the researcher. For example, $F\left(\boldsymbol{s}\right)$ might be a prior for or even estimates of the structural parameters of interest or the distribution of payoff-relevant variables. The benefit of maximizing unconditional on a particular $\boldsymbol{s}$, is that the resulting policy function can be used to evaluate policies for heterogeneous agents, where $\boldsymbol{x}$ varies, or for estimation, where $\boldsymbol{\vartheta}$ varies.

The remaining component of (14) is $b\left(\boldsymbol{s}\right)$, the baseline. Since $\boldsymbol{s}$ contains parameters that affect the scale of $V_{\pi}\left(\boldsymbol{s}\right)$, normalizing with $b\left(\boldsymbol{s}\right)$ ensures that I do not overfit the policy function to states which have high variance in objective values. As a simple example, one could use a running average of past values of $V_{\pi}\left(\boldsymbol{s}\right)$ for related states to construct $b\left(\boldsymbol{s}\right)$. In practice, I attempt and expand on several baselines recommended by the machine learning and combinatorial optimization literature. Appendix Section B.1 provides further details.

A final note on the choice to use a stochastic policy function. Combinatorial optimization problems can be plagued by local minima, interdependencies create lock-in effects where a firm may have to change several decisions at once to get a better outcome. This makes the prospect of a policy function that delivers a global optimum in one shot lofty. The stochastic policy function lets one instead place probability weight on parts of the solution space where a global optimum might exist. By then sampling the policy function and searching these spaces, one can more reliably produce optimal solutions.

[25] See Bottou and Bousquet (2007) for a discussion of the general tradeoffs between computational complexity, time, and approximation error in the context of learning algorithms.

4.2 Optimization Framework

I use the REINFORCE algorithm of Williams (1992) to optimize the objective (14). This is the same algorithm used in my main reference, Kool et al. (2019), to optimize a policy function for routing problems from operations research. Let us start with some iteration of the parameter vector Θ^n. With a completely parametrized policy function, I can do two things. I can both sample policies and evaluate those policies with the economic value function. Therefore, one gradient-based algorithm for updating Θ is

$$\begin{aligned}\Theta^{n+1} &= \Theta^n + \gamma \nabla \mathbb{E}_{\boldsymbol{s}}\Big[\mathbb{E}_\pi\left[V_\pi\left(\boldsymbol{s}\right) - b\left(\boldsymbol{s}\right)\right] \\ &= \Theta^n + \gamma \mathbb{E}_{\boldsymbol{s}}\left[\mathbb{E}_\pi\left[V_\pi\left(\boldsymbol{s}\right) - b\left(\boldsymbol{s}\right)\right] \nabla \log p\left(\pi|\boldsymbol{s};\Theta^n\right)\right],\end{aligned} \tag{15}$$

where γ is a scaling parameter that controls how fast I update parameters.

To understand how this updating works, suppose the policy function for the example from Section 2 was

$$p\left(\pi_t = i;\{\theta_1,\theta_2\}\right) = \sigma\left(\theta_1 T_i + \theta_2 d_{hi}\right),$$

where σ is the sigmoid function. Standard intuition says one should probably have a positive weight on T_i and a negative weight on d_{hi}. Markets with greater revenues and closer in distance should be more likely to be selected. If one started with $\theta_1 = \theta_2 = 0$ and took the gradient as in (15), then one would get a positive partial derivative for θ_1 and a negative partial derivative for θ_2. The gradient suggests that the objective will be on average larger if parameters are updated accordingly - raising θ_1 and lowering θ_2 so that choices with a higher T_i and a lower d_{hi} become sampled more frequently.

There are two challenges with directly applying equation (15): both the gradient and expectations can be difficult to compute. For the gradient issue, the function I use as the approximation will consist only of linear combinations and non-linear transformations of the form $f\left(x\right) = \max\left\{0, x\right\}$. The gradient will therefore be easy to compute. As for the expectations, I use a stochastic version of the gradient-based updating:

$$\Theta^{n+1} = \Theta^n + \gamma \sum_{\boldsymbol{s}}^{S^n} \left[V_\pi\left(\boldsymbol{s}\right) - b\left(\boldsymbol{s}\right)\right] \nabla \log p\left(\pi|\boldsymbol{s};\Theta\right),$$

where in each iteration n I sample a collection of state S^n from $F\left(\boldsymbol{s}\right)$ to approximate the outer expectation and sample one policy from $p\left(\pi|\boldsymbol{s};\Theta^n\right)$ to approximate the inner expectation.

At a high level what I am doing is not too different from grid-based policy function iteration. I start by guessing a policy function, only in this case the parameters of a particular function $p\left(\pi|\boldsymbol{s};\Theta\right)$. I then look at the implied policies and when certain policies return relatively better values of the objective function, I modify the policy function accordingly. I repeatedly evaluate problems and update policies until convergence.

Appendix Section B.2 describes how the optimization algorithm can be augmented with theo-

retical knowledge about the optimal policy function. If we know the optimal policy function must satisfy certain conditions, then a penalty term based on the satisfaction of these condition can be added to optimization objective. I show experimental results using the proposition regarding firm productivity and optimal policies in Antràs et al. (2017). Finally, Appendix Section B.3 describes how a policy function can alternatively be fit to data on optimal policies.

5 Machine Learning Models for Function Approximations

The previous section described how a parametric policy function could be optimized. But, in order for that parametric function to return optimal policies, it must be capable of approximating the true optimal policy function - a function with a high dimensional state space and complex interdependencies between the variables of those states. Fortunately, machine learning has delivered models precisely designed for these sorts of tasks.[26] I employ a model known as a Transformer, first proposed by Vaswani et al. (2017). My specific starting point is the model used by Kool et al. (2019), and precise details about the model architecture can be found in Appendix Section A. This Section overviews the model components and intuitively discusses how they each contribute to approximating the optimal policy function.

The high-level workflow is as follows. First, I use the characteristics of each choice and the features of the problem to create a vector-space representation of each choice. This new representation of the problem summarizes the relevant information for each choice and its interdependence with other choices. Second, I use those vectors to iteratively construct a solution, adding choices based on information collected in their vector representation. This iterative construction additionally lets me change the preferences for choices based on characteristics of those already made - prioritizing complements and avoiding substitutes.

Running Example To illustrate how the approximation works with an example, let us continue with the export-market interdependence problem from Section 2. I assume here that firms have constant returns to scale in production, $\beta = 1$, to simplify exposition. The objective is then

$$\max_{\{\mathbb{1}_i\} \in 2^N} \sum_{i=1}^{N} \mathbb{1}_i T_i - \sum_{i=1}^{N} \mathbb{1}_i \left(\alpha d_{hi} - \gamma \sum_{j \neq i}^{N} \mathbb{1}_j d_{ji}^{-1} \right). \tag{16}$$

5.1 Encoder

The first stage of the model is an Encoder, which generates a vector-space representation of each choice. The goal of this representation is to collect relevant information about each choice and relate

[26] There exists universal approximation theorems, e.g. Hornik et al. (1989), that prove there exists machine learning models that are capable of approximating all functions from certain function spaces. I do not attempt a formal statement here, but there are theoretical foundations for the success of machine learning models in some settings.

that information to the other choices. Let x_i denote the characteristics of choice i and $\boldsymbol{\vartheta}$ denote the structural parameters of the economic model. In the example, (16), this includes location revenues and distances, $x_i \equiv \{T_i, d_{hi}, \{d_{ij}\}\}$, as well as the parameter $\boldsymbol{\vartheta} \equiv \{\alpha, \gamma\}$.

The Encoder's first component, an initial embedding function parametrized by Θ_u, takes the inputs for each choice, x_i and $\boldsymbol{\vartheta}$, and generates an initial vector-space representation of that choice

$$u_i^0 = \text{Initial Embedding}\left(x_i, \boldsymbol{\vartheta}; \Theta_u\right).$$

This consists of taking linear combinations of the inputs and applying non-linear transformations to them: e.g. $g\left(\theta_1 T_i + \theta_2 d_{hi}\right)$.[27] This is separately done for each dimension of u_i^0, allowing one to carry multiple representations of each choice. In the example, one could imagine one dimension of u_i^0 is $(T_i - \alpha d_{hi})$ and the others could just pass on the components of x_i. Note, a non-linear transformation of the inputs, like αd_{hi}, can be approximated with various piecewise linear functions.

The second component is an information mixing function parametrized by Θ_μ that interacts the information in each u_i^0 across dimensions and choices to create a new vector-space representation for each choice:

$$\mu_i = \text{Information Mixer}\left(u_i^0, \left\{u_i^0\right\}; \Theta_\mu\right).$$

In the example, one can imagine numerous ways of interaction the initial embeddings to get new representations of each choice. For instance,

$$\sum_{j=1}^{N} \left[\left(T_i - \alpha d_{hi}\right) - \left(T_j - \alpha d_{hj}\right)\right] \quad \text{or} \quad \sum_{j=1}^{N} \left[\gamma d_{ji}^{-1}\left(T_j - \alpha d_{hj}\right)\right]. \tag{17}$$

Both of these seem to capture a feature of the choice relevant for the optimization problem in equation (16). The first measures how unilaterally profitable i is relative to the other choice. The second, how close each choice is to other profitable choices. One would likely want to add choices with high values of these new characteristics. While these representations have intuitive meaning as I have presented them, this is not a requirement of the Encoder - it will learn whatever representation and interaction of the input is relevant for optimization.

5.2 Decoder

The second stage of the model is a Decoder, which generates a sequence of actions based on the choices' embeddings. Let π_t denote the t-th choice selected and $\boldsymbol{\pi}_{1:t}$ the sequence of choices made through t steps. The Decoder consists of an information mixing function parametrized by Θ_π that combines the vector-space representation of each choice to produces a probability distribution over

[27] The function g is known as an activation function. A popular choice is the rectified linear unit: $g(x) = \max\{0, x\}$. These activation functions enable approximation of non-linear function. Relatedly, commercial solvers for integer non-linear programs use linear approximations of non-linear objectives and then standard linear programming techniques to optimize.

actions:

$$\lambda_i = \text{Information Mixer}\left(\mu_i, \{\mu_i\}, \boldsymbol{\pi}_{1:t-1}; \Theta_\pi\right)$$

$$p\left(\pi_t = i | \boldsymbol{\pi}_{1:t-1}, \boldsymbol{\mu}; \Theta_\pi\right) = \frac{\exp\left(\lambda_i\right)}{\sum_k \exp\left(\lambda_k\right)}.$$

The sequential construction of the policy allows one to dynamically adjust preferences for choices based on the complementarity or substitutability generated with the already made choices, $\boldsymbol{\pi}_{1:t-1}$.

Going to the example, one might give a weight to each choice based on the speculated encoder output in (17), but give additional weight to choices that are close to those already selected:

$$\lambda_i = \theta_1 \sum_{j=1}^{N} \left[\left(T_i - \alpha d_{hi}\right) - \left(T_j - \alpha d_{hj}\right)\right] + \theta_2 \sum_{j=1}^{N} \left[\gamma d_{ji}^{-1} \left(T_j - \alpha d_{hj}\right)\right] + \theta_3 \sum_{j \in \boldsymbol{\pi}_{1:t-1} \setminus i} \gamma d_{ji}^{-1}.$$

This way I start by adding choices which have high unilateral profit and are close to other high-profit locations. Then, for the later choices, I give additional consideration to the locations near those already selected.

To complete the notation, note that the probability for a particular policy from initial to terminal choice, $\bar{\boldsymbol{\pi}} = \{\bar{\pi}_1, ..., \bar{\pi}_T\}$, is the product of the relevant sequence of probabilities

$$p\left(\bar{\boldsymbol{\pi}} | \boldsymbol{\mu}; \Theta_\pi\right) = \prod_{t=1}^{T} p\left(\pi_t = \bar{\pi}_t | \bar{\boldsymbol{\pi}}_{1:t-1}, \{\mu_i\}; \Theta_\pi\right).$$

And finally, consider the Encoder and Decoder to be just two components of an overarching policy function: $p\left(\pi | \boldsymbol{s}; \Theta\right)$ where $\Theta = \{\Theta_u, \Theta_\mu, \Theta_\pi\}$ and $\boldsymbol{s} = \{\boldsymbol{x}, \boldsymbol{\vartheta}\}$. It is this policy function that is the subject of optimization in Section 4.2.

5.2.1 Potential Reward Information for Dynamic Decoding

This project contributes one core model feature not found in previous combinatorial optimization and machine learning work. Given a partial policy $\boldsymbol{\pi}_{1:t-1}$, I compute the change in objective function from each potential choice, $\{R\left(s_{t-1}, \pi_j\right); \forall j\}$. This scalar value is added to the Decoder's information mixer at each step. This gives an intermittent signal directly from the objective function on how valuable each choice is given the current state.

Back to the example, the difference between the change in objective function and the unilateral payoff of a choice is the complementarity generated by that choice for the current partial solution:

$$R\left(s_{t-1}, \pi_i\right) - \left(T_i - f_i\right) = \gamma \sum_{j \in \boldsymbol{\pi}_{1:t-1} \setminus i}^{N} d_{ji}^{-1}.$$

The potential reward information is helpful for then distinguishing what choices are valuable because they specifically affect the current policy, through complementarity of substitution effects.

Additional details are in Appendix Section A.2.1 and I quantify the numerical contribution of this component in Section 6.4.

6 Numerical Experiments - Evaluating Optimality

I evaluate performance in two dimensions: the difference in objective value and compute time.[28] I refer to $V_{\pi^x}(\boldsymbol{s})$ as the cost of policy x in problem instance $\boldsymbol{s}$. To compare policies I use the following measure

$$\text{Optimality Gap}\,(x, y|\boldsymbol{s}) \equiv \frac{V_{\pi^x}(\boldsymbol{s}) - V_{\pi^y}(\boldsymbol{s})}{\frac{1}{S}\sum_s V_{\pi^y}(\boldsymbol{s})} \cdot 100 \tag{18}$$

where x and y are distinct policies and $\boldsymbol{s}$ is an instance of the problem. Effectively, I am evaluating the difference in the objective function and normalizing by the mean objective value across problem instances and scaling to get a "percentage" deviation interpretation.[29]

Producing Solutions from Policy Functions There are several ways to use the stochastic policy function to deliver solutions. First, there is greedy decoding which takes the policy with the highest probability. Second, there is sample-many decoding which samples many times from the policy distribution, evaluating each policy, and proceeding with the best one.[30] Third, there is ensemble decoding which trains many policy functions, evaluating each using greedy or sample-many decoding, and taking the best policy across all these evaluations. The intuition behind this third approach is that models are each initialized to random parameter values and exposed to a random sample of problem instances. This may cause certain models to be adept at a subset of problems, so by pooling policy functions one can maximize performance across a wider distribution of problems.

6.1 Input Sourcing & Plant Location

For the Input Sourcing and Plant Location problem from Section 3.1, policies generated from the trained policy function are within 0.01% and 0.04% of the best alternative solution for the 20 and 50 locations cases. Table 1 reports the mean optimality gap and compute time for several algorithms.[31] Performance is evaluated over a sample of problems, each with randomly generated payoff terms $\{T_i, f_i\}$ and value of α drawn from the interval $[0.5, 1.5]$. The submodular, $\alpha < 1$, and supermodular, $\alpha > 1$, cases are separately evaluated.

[28] All model calculations are performed on a single NVIDIA GeForce RTX 4090. This is a consumer grade graphics processing unit (GPU) that can be integrated in most desktop workstations. All computation that uses CPUs was done on a 32-Core AMD CPU.

[29] I normalize by the mean objective value since in cases where the policy prescribes no action, the objective has a value of 0 and the simple percentage deviation then becomes undefined. Therefore, using percent notation with this measure is technically incorrect, but it helps with the intuition of what is being measured.

[30] Whenever I employ sample-many decoding, I also run greedy decoding and take the best policy.

[31] Appendix Figures (21)-(24) shows the distribution of optimality gaps across all problem instances.

There are several algorithms I use as benchmarks. The first is brute-force optimization. This involves evaluating all 2^N possible combinations and taking the best option. I only do this in the $N = 20$ case as it would be impossible to do for one of the larger problem instances. The second is the Gurobi Mixed Integer Non-Linear Program optimizer (Gurobi Optimization, 2024), a commercial software specifically designed for this general class of problems. Gurobi is well known for providing tools for solving linear integer programs, where the solutions are proven to be globally optimal. It has recently provided tools for solving non-linear problems which linearly approximate the supplied non-linear objective and constraints. But, the solution to the approximate problem is not guaranteed to be optimal for the original non-linear problem.

The third algorithm I consider is a bounding algorithm, either Jia (2008) in the supermodular case or Arkolakis et al. (2023) in the submodular case. The fourth is a combination of the bounding algorithm and simulated annealing. Simulated annealing involves probabilistically accepting random perturbations to the policy based on changes in the objective function. Appendix Section B.6 provides a detailed description. The final set of algorithms, prefixed by "Model", are those proposed by this paper. This includes greedy decoding, sample-many decoding, and ensemble decoding of the stochastic policy function trained on the same distribution of problem instances.[32]

The best model-generated solutions are within 0.04% of the best found solution.[33] The bounding algorithms always return the global optimum in the supermodular case, and I closely replicate these policies with mean optimality gaps of 0.01% and 0.04% for the 20 and 50 locations cases.[34] In the sub-modular case this is however not true, a large number of instances remain where the bounds are not unique. Simulated annealing is able to close the optimality gap further but solutions remain far from the optimum.[35] The policy function is able to overcome this problem, producing optimal solutions with a gap of 0.00%.

6.2 Oligopoly in Input Sourcing and Plant Location

For the Oligopoly Input Sourcing and Plant Location problem from Section 3.2, policies generated from the trained policy function are within 0.09% of the global optimum in the 20 location case, and 120% or 46% better than a greedy search in the 50 and 100 location cases. Table 2 reports the mean optimality gap and compute time for several algorithms.[36] Performance is evaluated over a

[32] One feature of the transformer model I use is its invariance to input size. You could use the same model for $N = 20$ as you use for $N = 50$, but in practice training separate models for various sizes works best.

[33] Brute Force is the best solution in $N = 20$ and represents global optimum. Gurobi is the best solution in $N = 50$, but since it uses a linear approximation of the objective function it can not guarantee that the found solution is the global optimum.

[34] This is consistent with Appendix Table A.4 of Antràs et al. (2017) where less than 0.001% of problem instances in their baseline calibration had bounds that did not uniquely identify an optimum.

[35] Arkolakis et al. (2023) additionally provides a branching procedure for exploring the remaining possibilities between bounds.

[36] Appendix Figures (25)-(27) shows the distribution of optimality gaps across all problem instances.

Table 1: Input Sourcing and Plant Location Benchmarks

Method	$N = 20$ (Sub)		$N = 20$ (Sup)			$N = 50$ (Sub)		$N = 50$ (Sup)		
	Cost	Gap	Cost	Gap	Time	Cost	Gap	Cost	Gap	Time
Brute Force	-10.65	0.00%	-10.39	0.00%	(8m)					
Gurobi	-10.65	0.00%	-10.39	0.00%	(31s)	-3.38	0.00%	-42.71	0.00%	(47s)
Bounds	-10.20	22.48%	-9.92	0.00%	(85ms)	-2.12	37.28%	-42.71	0.00%	(292ms)
Bounds + SA	-10.39	12.88%	-10.12	0.00%	(25s)	-2.63	22.37%	-42.71	0.00%	(51s)
Model - Greedy	-10.65	0.00%	-10.39	0.03%	(3s)	-3.32	1.84%	-42.57	0.34%	(17s)
Model - Sample Many	-10.65	0.00%	-10.39	0.01%	(1m)	-3.33	1.41%	-42.69	0.04%	(7m)
Model - Ensemble	-10.65	0.00%	-10.39	0.02%	(18s)	-3.38	0.00%	-42.69	0.04%	(2m)

Note. Time to complete 10,000 vectorized instances simultaneously. The optimality gap is between the listed method and Brute Force for $N = 20$ and Gurobi for $N = 50$. The Cost and Gap are the mean values across all instances. Sample Many uses 25 samples. Ensemble uses 7 separately trained models.

sample of problems, with a fixed set of parameters $\{\theta, \sigma, \varepsilon, \bar{p}\}$ and randomly generated payoff terms $\{T_i, f_i\}$.[37] Since the model does not feature strict substitutability or complementarity, bounding algorithms will not work. I consider several alternatives.

First, I can still use a brute force search for the 20 location case, where all 2^N possibilities are evaluated. Second, I again use Gurobi, but display the results here to highlight its limits for non-linear problems. This oligopoly problem introduces an additional non-linearity with the determination of market share. Gurobi's linear approximation performs poorly in this setting, leading to poor solution quality. Third, I use a greedy search that iteratively adds the choice which generates the greatest increase in the objective function, stopping when all remaining choices generate negative returns. Fourth, I run a simulated annealing algorithm which stochastically searches for a better solution starting from the greedy solution.

Policies are within 0.09% of the global optimum for simulations with 20 locations. For the 50 and 100 location version I benchmark to a greedy search; policies are 120% and 46% better in the 50 and 100 location case. Unlike the previous problem, this is a setting for which we have no theoretically motivated algorithms. The non-linearities hinder standard optimizers like Gurobi, the presence of complements and substitutes prevent bounding procedures from working, and the problem size makes a heuristic algorithm like simulated annealing perform poorly given limited time. On the other hand, this methodology outperforms all referenced algorithms and features compute times amenable to estimation.

[37] I also fix the markup to $\sigma/(\sigma - 1)$ to avoid the additional fixed point computation. The insights about a change from complementarity to substitutability still remain.

Table 2: Oligopoly Input Sourcing and Plant Location Benchmarks

Method	$N = 20$			$N = 50$			$N = 100$		
	Cost	Gap	Time	Cost	Gap	Time	Cost	Gap	Time
Brute Force	-1.83	0.00%	(10m)						
Gurobi	-0.05	97.24%	(53m)	0.21	113.81%	(47m)	0.34	112.27%	(23m)
Greedy	-0.32	82.51%	(242ms)	-1.49	0.00%	(7s)	-2.81	0.00%	(53s)
Greedy + SA	-1.45	20.99%	(13s)	-1.67	-11.78%	(24s)	-2.89	-2.95%	(1m)
Model - Greedy	-1.82	0.70%	(4s)	-3.19	-113.61%	(34s)	-0.63	77.47%	(5s)
Model - Sample Many	-1.83	0.09%	(1m)	-3.29	-120.71%	(14m)	-4.10	-46.09%	(2m)
Model - Ensemble	-1.83	0.16%	(22s)	-3.20	-114.20%	(3m)	-1.07	61.96%	(20s)

Note. Time to complete 10,000 vectorized instances simultaneously. The optimality gap is between the listed method and Brute Force for $N = 20$ and Greedy for $N = 50$ and $N = 100$. Parameters are fixed to $\theta = 1.5$, $\sigma = 8.0$, $\varepsilon = 1.5$, and $\bar{p} = 0.1$. Sample Many uses 25 samples. Ensemble uses 7 separately trained models for $N = 20$ and $N = 50$, 4 models are used for $N = 100$.

6.3 Global Value Chains

For the Global Value Chain problem from Section 3.3, policies generated from the trained policy function are roughly 1% better than the approximate global optimum returned by the dynamic programming solution for the 20, 50, and 100 location cases. Table 3 reports the mean optimality gap and compute time for several algorithms.[38] Performance is evaluated over a sample of 1,000 problems, with a fixed set of parameters $\{\sigma_j, \alpha_j\}$ and randomly generated payoff terms $\{\{C_n^j\}, \{\tau_{nm}, f_{nm}\}\}$.[39]

The first benchmark algorithm I consider is the dynamic programming solution described in Section 3.3. This method returns the global optimum up to the discretization error introduced by turning a continuous state space into a grid. I next introduce a set of heuristic methods called greedy M-level Search. This involves starting from the empty sequence and considering all possible extensions of length $M < J$. The best found extension becomes the location of the next stage. This process is repeated until the entire chain is constructed. Intuitively, this approximates an optimization problem over N^J sequences with one of several shorter segments with length N^M.[40] I also consider a random and simulated annealing solution.

The trained policy function outperforms all the heuristic algorithms as well as the dynamic programming algorithm, which returns global optima up to discretization error. It does so with a fraction of the computational time. The greedy solutions take 0.2 seconds, roughly 18,000 times faster than the dynamic programming solution.

I additionally report compute times by device in the last four rows of Table (3). Machine

[38] Appendix Figures (28)-(30) shows the distribution of optimality gaps across all problem instances.

[39] In practice, I randomly generate 2-D coordinates and make fixed and trade costs a function of the bilateral distances between locations.

[40] I consider a value chain of length $J = 10$. The 3-level greedy search requires $(J-2) \times N^3 + N^2 + N$ evaluations. I look 3 steps ahead at each segment of the value chain with the exception of the 9th and 10th stages which require looking 2 and 1 step ahead respectively.

learning models are specialized for performance on hardware known as GPUs, but can run on any more standard CPUs as well. Most machine learning workflows involve doing model training on GPUs and model inference to produce solutions on CPUs. The GPUs are particularly adept at the kind of parallel computation required for updating parameters with gradient information. On the other hand, generating model output requires much less resources since I do not require gradient computation. And, since CPUs are widely available they can be used at scale and in parallel for this task. Since the largest computational burden in our applications is often estimation, which conditions on a trained model, one can apply CPUs there.

Table 3: Global Value Chain Benchmarks

Method	$N = 20$			$N = 50$			$N = 100$		
	Cost	Gap	Time	Cost	Gap	Time	Cost	Gap	Time
Dynamic Programming	60.41	0.00%	(2m)	57.34	0.00%	(15m)	56.33	0.00%	(60m)
Random	71.77	18.81%	(8s)	71.20	24.17%	(16s)	71.28	26.54%	(41s)
Simulated Annealing	69.21	14.58%	(1m)	68.73	19.86%	(2m)	68.87	22.27%	(6m)
Greedy - 1 Level	71.72	18.73%	(13ms)	67.07	16.97%	(24ms)	63.83	13.31%	(47ms)
Greedy - 2 Level	66.83	10.64%	(1s)	63.54	10.82%	(2s)	61.15	8.56%	(7s)
Greedy - 3 Level	63.46	5.04%	(10s)	60.38	5.31%	(1m)	58.95	4.65%	(9m)
Model (GPU) - Greedy	59.98	-0.70%	(33ms)	57.01	-0.57%	(76ms)	55.80	-0.93%	(202ms)
Model (CPU) - Greedy	...	...	(278ms)	...	...	(1s)	...	...	(2s)
Model (GPU) - Sample Many	59.54	-1.44%	(2s)	56.79	-0.96%	(4s)	55.63	-1.24%	(6s)
Model (CPU) - Sample Many	...	...	(24s)	...	...	(1m)	...	...	(3m)

Note. Time to complete 1,000 vectorized instances simultaneously. The optimality gap is between the listed method and Dynamic Programming for all instances. Parameters are fixed to $\{\sigma_j\} \equiv \{3, 4, 5, 3, 4, 5, 3, 4, 5, 5\}$ and $\{\alpha\} = \{0.3, 0.35, 0.4, 0.45, 0.5, 0.55, 0.6, 0.65, 0.70, 0.75\}$. Sample Many uses 100 samples.

6.4 Model Training

Training the policy functions takes on the order of magnitude of minutes. The training procedure was described in Section 4, but to summarize it involves evaluating the policy function on many problem instances and updating its parameters to place more probability weight on better policies. Table 4 provides information on how long it takes each model to process 10,000 unique problems and Figures 1 and 2 as well as those in Appendix Section D.4 plot the optimality gap of a model throughout training. In general, I require between 50,000 and 1,000,000 problem instance to converge the policy functions - extrapolating from the times in Table 4 this means I generally require on the order of magnitude of minutes to train.[41]

[41] The exception here is ISPL - Oligopoly with 100 locations which uses a sample-many baseline which is particularly slow to construct. The use of this baseline is why Table (2) reports the best results for sample many decoding when $N = 100$. Improving the training of this model is a work in progress.

The second column of Table 4 reports the number of parameters in each model. In terms of machine learning models, the models used here are relatively small - they can be optimized on a personal workstation. The reason for the variability in model size is due to a meta-layer of model training known as hyper-parameter tuning. Hyper-parameters are parameters that govern features of the model and learning but are not directly used in producing model output. Appendix Section B.7 describes hyper-parameter selection in more detail. A work in progress is fine tuning the hyper-parameters to both settle on a universal policy function structure and stabilize training.

The third column of Table 4 highlights how crucial the dynamic decoding component developed in Section 5.2.1 is to producing optimal solutions. This column displays the difference in mean optimality gap between the same model trained both with and without the dynamic decoding component. On the majority of problems I consider, the policy function approximation would not be competitive without this new feature. The cases where this is not true is the Oligopoly Input Sourcing and Plant Location model for 100 locations as well as the GVC models. The Oligopoly model is able to achieve good performance since it uses an intensive training baseline, resulting in long training times.[42] The GVC model may be doing well because the machine learning model used is tailored for sequential prediction problems, and the GVC problem alone requires a sequence of decisions.

Table 4: Model Training

Model	Time per 10000	Parameters (1000)	Dynamic Decoding Difference
ISPL (20)	460ms	479	-66.34%
ISPL (50)	3s	677	-88.86%
ISPL - Oligopoly (20)	1s	2700	-12.75%
ISPL - Oligopoly (50)	2s	121	-123.32%
ISPL - Oligopoly (100)	3m	976	-1.01%
GVC (20)	6s	678	0.13%
GVC (50)	8s	678	0.08%
GVC (100)	11s	270	0.04%
Export (20)	2s	546	-42.63%
Export (50)	6s	477	-51.38%

Note. ISPL refers to Input Sourcing and Plant Location. The number of locations, N, is represented in parentheses next to each problem name. A negative value of Dynamic Decoding Difference suggests that the model trained with the dynamic decoding component is performing better than the model trained without it.

[42] Appendix Section B.1 discusses why the baseline used for $N = 100$ slows training - it requires sampling from the policy function many times.

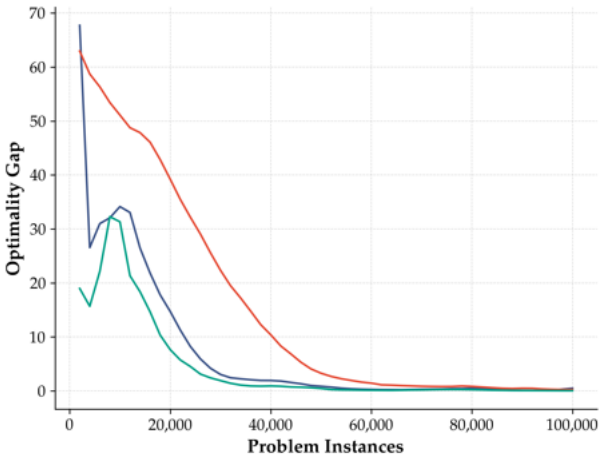


Figure 1: ISPL ($N = 20$)

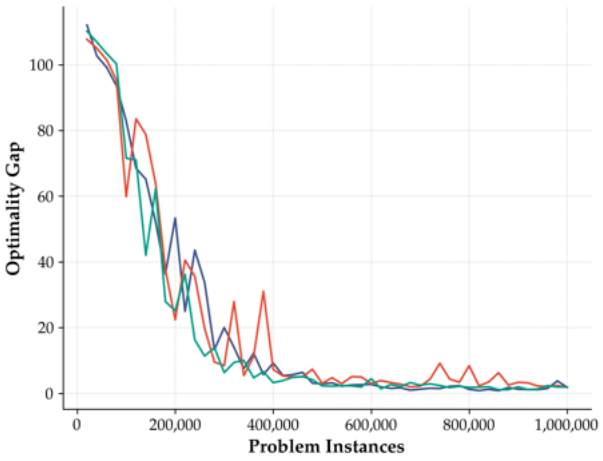


Figure 2: ISPL - Oligopoly ($N = 20$)

Note. Each line corresponds to a separate model trained under a new initialization of parameters and sample of training problems. ISPL stands for Input Sourcing and Plant Location.

7 Estimation

The output of the methodology is a policy function which maps the state of a problem to an optimal policy. The corresponding estimation problem is then given observed policies and environments, how do I find the structural parameters that rationalize those observations. Suppose we have data on policies $\left\{\pi_i^D\right\}$ and the environment that those policies were taken in $\left\{\boldsymbol{x}_f^D\right\}$ - f here indexes the decision-making agent here. Working with the model of Section 2, I might observe the export decisions of firms, $\left\{\pi_f^D\right\}$, and observe their revenues and distances between markets, $\left\{\boldsymbol{x}_f^D\right\}$. The parameters I am interested in estimating are those that govern fixed costs and returns to scale $\boldsymbol{\vartheta} \equiv \{\beta, \alpha, \gamma\}$. I estimate $\boldsymbol{\vartheta}$ by solving

$$\min_{\boldsymbol{\vartheta}} \sum_f \mathbb{E}_{p\left(\pi|\left\{\boldsymbol{x}_i^D, \boldsymbol{\vartheta}\right\}; \Theta\right)} \left[\mathcal{D}\left(\pi, \pi_i^D\right)\right]$$

where Θ are fixed from training and $\mathcal{D}$ is some distance measure between two policies. What this amounts to is, conditional on a trained policy function, finding the value of $\boldsymbol{\vartheta}$ that results in model-implied policies that are as close as possible to the policies I observe in the data.[43] Note, at this point I have already trained a policy function that understand how the value of structural parameters alters the optimal policy; solving this estimation problem just requires using standard global and local optimizers over $\boldsymbol{\vartheta}$.

[43]See Appendix Section B.5 for an alternate estimation strategy from Duarte and Fonseca (2023) that estimates an auxiliary model that directly predicts the relationship between parameters and moments.

Approximation Error I am using an approximate method which is not guaranteed to return the global optima to the objective. This is a necessary tradeoff as the brute-force optimization problem is otherwise numerically intractable. However, this introduces an estimation issue if the approximation error is systematically related to the structural parameters I am estimating, and I believe that firms are finding the global optimum. By approximation error, I refer to the fact that $\mathcal{D}\left(\pi, \pi^D\right) \neq \mathcal{D}\left(\pi^{Opt}, \pi^D\right)$ where π^{Opt} is the globally optimal policy, π is the policy prescribe by the policy function, and π^D is the policy observed in the data. I am only able to calculate $\mathcal{D}\left(\pi, \pi^D\right)$, but minimizing $\mathcal{D}\left(\pi^{Opt}, \pi^D\right)$ is the true objective of estimation. It is hard to directly say anything about this issue, but part of the purpose of the experiments in Section 6 is to show that these methods are capable of regularly returning the global optima and hence $\mathcal{D}\left(\pi, \pi^D\right) = \mathcal{D}\left(\pi^{Opt}, \pi^D\right)$.

7.1 Bayesian Estimation - Complexity and Optimality Tradeoff

The policy functional approximation may worsen as I increase the range of structural parameters considered. For example, consider the input sourcing and plant location model from Section 3.1. Two separate algorithms are required to bound the supermodular ($\alpha > 1$) and submodular ($\alpha < 1$) cases. Likewise, it might be difficult for a single policy function to prescribe optimal policies over a range of parameter values that includes $\alpha > 1$ and $\alpha < 1$. In practice the models are quick to train, so an alternate estimation procedure where I re-optimize a policy function over a varying distribution of parameters is feasible. To formalize this idea, I use a Bayesian estimation procedure, where the prior and posterior of the estimates inform the range of parameters I should be training over.

I start with a prior over the parameters of interest $\boldsymbol{\vartheta}$, $F\left(\boldsymbol{\vartheta}\right)$. Suppose that the agent has idiosyncratic preference for certain policies and maximizes

$$\max_{\pi} V\left(\pi | \left\{\boldsymbol{x}^D, \boldsymbol{\vartheta}\right\}\right) + \varepsilon_{\pi}.$$

I observe a policy choice π^D in the data as well as the payoff relevant variables $\boldsymbol{x}^D$. I also have a model prescribed a policy π^* in state $\boldsymbol{x} = \left\{\boldsymbol{x}^D, \boldsymbol{\vartheta}\right\}$. The likelihood of observing this choice is then

$$p\left(\varepsilon_{\pi} | \boldsymbol{\vartheta}, \boldsymbol{x}^D, \pi^D\right) \geq V\left(\pi^* | \boldsymbol{x}\right) - V\left(\pi^D | \boldsymbol{x}\right).$$

I can then form a posterior for the parameter of interest:

$$p\left(\boldsymbol{\vartheta} | \boldsymbol{x}^D, \pi^D\right) = \frac{p\left(\varepsilon_{\pi} | \boldsymbol{\vartheta}, \boldsymbol{x}^D, \pi^D\right) p\left(\boldsymbol{\vartheta}\right)}{p\left(\varepsilon_{\pi} | \boldsymbol{x}^D, \pi^D\right)}.$$

This motivates an estimation procedure where I start by training a model over a sample of problem instances with parameter values sampled from a prior $p\left(\boldsymbol{\vartheta}\right)$. Once the policy function converges, I use the observed policies to form a posterior over parameter values $p\left(\boldsymbol{\vartheta} | \boldsymbol{x}^D, \pi^D\right)$. I then retrain the

model over this new distribution of parameters, repeating until I converge to a posterior over $\boldsymbol{\vartheta}$.

7.2 Quasi-Bayesian Estimation for Moment-Based Estimators

When estimating models of combinatorial choice, we often do not directly match the policies of a specific agent. This is because agents are subject to idiosyncratic shocks that affect their payoff from various choices. But, we have no way of knowing what the specific realization of those shocks were for an agent in the data. That is, there are elements of $\boldsymbol{x}$ which I do not observe. Instead, we try to replicate moments of the distribution of observed policies. This involves simulating agents with a distribution of idiosyncratic shocks, and matching moments of their optimal policies to moments observed in the data. To be concrete, imagine firms are characterized by an idiosyncratic productivity which affects their payoff from exporting. Although each firm's productivity is unobserved, we might believe the distribution of productivity in the sample in Pareto. Therefore, we simulate firms from a Pareto distribution and compute moments of their optimal policies (e.g. export probabilities) to be used in estimation.

Let us now consider estimation based on moments of the policy distribution. Let $m(\{\pi_f\})$ be some moment or vector of moments of the distribution of agent-specific policies $\{\pi_f\}$. The goal is to minimize the distance, $\mathcal{D}\left(m(\{\pi_f(\boldsymbol{\vartheta}, \boldsymbol{x}^D)\}), m(\{\pi_f^D\})\right)$, between model generated and observed policy moments where $\pi_f\left(\boldsymbol{\vartheta}, \boldsymbol{x}^D\right)$ just specifies that model generate policies are a function of parameters and other payoff relevant variables. Following Chernozhukov and Hong (2003), let us then define a quasi-posterior:

$$p\left(\boldsymbol{\vartheta}|\boldsymbol{x}^D, \pi^D\right) = \frac{\exp\left(-\mathcal{D}\left(m\left(\left\{\pi_i\left(\boldsymbol{\vartheta}, \boldsymbol{x}^D\right)\right\}\right), m\left(\left\{\pi_i^D\right\}\right)\right)\right) p\left(\boldsymbol{\vartheta}\right)}{\int \exp\left(-\mathcal{D}\left(m\left(\left\{\pi_i\left(\tilde{\boldsymbol{\vartheta}}, \boldsymbol{x}^D\right)\right\}\right), m\left(\{\pi_i^D\}\right)\right)\right) p\left(\tilde{\boldsymbol{\vartheta}}\right)}.$$

This is a quasi-posterior in the sense that I am forming a likelihood for an objective value, not some structural shock to any agent's payoffs or preferences.

The quasi-posterior is a useful object for estimation. First, it is still capable of guiding the policy function optimization to a distribution of parameters for which the estimation objective is relatively smaller. With a flat prior, the posterior is maximized precisely at the parameter vector that minimizes $\mathcal{D}(\cdot)$. Second, it turns a problem of non-linear optimization over $\boldsymbol{\vartheta}$ into one integration. For this I can use a suite of Markov Chain Monte Carlo tools to get the posterior distribution. This is relevant in my setting since the objection function $\mathcal{D}$ is discontinuous: it is made up of finite agents making discrete decisions so for some small parameter changes it will not change. This makes gradient-based optimization over $\boldsymbol{\vartheta}$ challenging.

8 Empirical Application

The results in the previous section demonstrated how these tools can be used to solve arbitrary combinatorial optimization problems to (near) optimality in a relatively short amount of time. This aids in structural estimation by both allowing us to consider a richer set of models with less restrictions on the interactions between choices, and speeding up optimization. As a proof of concept, I estimate a static version of the export interdependence model from Alfaro-Urena et al. (2023), where I introduce a mechanism for substitutability through scale effects in the exporting firm. The extended model produces better model fit relative to a model with only complementarity and generates qualitatively different substitution patterns in response to trade shocks.

Data The World Bank Exporter Dynamics Database contains time series export data for firms in 70 countries. The publicly available aggregate dataset is constructed from firm-level customs data, see Fernandes et al. (2016) for details. For 11 countries, the underlying micro data is available to researchers which contains a time series of firm-product level export quantities and prices. I focus the analysis on the 2006 panel of exporting firms from Mexico and only the top 20 export destinations.[44] There are 36,086 firms with an average total export revenue of nearly $7 million. The average number of export destinations per firm is 1.65. Since I do not observe revenues for all firm-destination pairs, all potential export values are estimated using firm and country fixed effects. Appendix Section 10 describes the procedure in more detail and visualizes estimates.

8.1 Model

Consider a firm f located in a home market h, Mexico. It must decide on a subset of N countries to export to, maximizing profits according to

$$\max_{\{\mathbb{1}_i\}\in 2^N} \pi_f\left(\{\mathbb{1}_i\}\right) - \sum_{i=1}^{N} \mathbb{1}_i f_i\left(\{\mathbb{1}_i\}\right).$$

The indicator $\mathbb{1}_i$ takes a value of 1 if the firm chooses to export to country i. The first interdependence I introduce, modeled following Alfaro-Urena et al. (2023), is a complementarity arising from exporting to multiple proximate locations:

$$f_i\left(\{\mathbb{1}_i\}\right) = \eta + \alpha d_{hi} - \gamma\left(1 + \phi d_{hi}\right) \sum_{k\neq i}^{N} \mathbb{1}_k \exp\left(-\beta d_{ki}\right).$$

[44] These are the 20 locations for which I observe the most number of firms exporting. They account for 77% of export events and 95% of export revenues.

Here d_{hi} is the distance between location h and i.[45] Fixed costs are directly increasing in distance, αd_{hi}. If the firm exports to additional markets nearby destination i, it enjoys a fixed cost reduction. That fixed cost reduction depends on how far the destination is from the firm, ϕd_{hi}. It also depends on how close those additional markets in the choice set are to the destination, $\exp(-\beta d_{ki})$.

The second interdependence I introduce is a concavity in the profit function that captures decreasing returns to scale a firm faces when exporting. As one possible mechanism, evidence for increasing marginal costs, particularly in an exporting context, has been documented by Almunia et al. (2021). I therefore posit the following functional form on profits:

$$\pi_f\left(\{\mathbb{1}_i\}\right) = \left(\sum_{i=1}^{N} \mathbb{1}_i \delta_i \left(r_{fi} + \nu_{fi} + \omega_{fi}\right)\right)^{1-\theta}, \tag{19}$$

which can be derived from an increasing marginal cost curve (see Appendix Section C.1). Here r_{fi} is the export potential of a destination i, ν_{fi} is an idiosyncratic revenue shock, ω_{fi} is blocking shock, and δ_i is a revenue shifter for countries belonging to NAFTA (USA and Canada). The revenue shock is drawn from a multivariate normal distribution:

$$\begin{aligned}\nu_{fi} \sim & \mathcal{N}\left(0, \sigma^2\right) \\ \rho_{ki} = & \rho_0 \exp\left(-\rho_d d_{ki}\right),\end{aligned}$$

where ρ_{ki} is the correlation in revenue shocks across locations. The blocking shock takes one of two values, 0 or ∞, with probabilities p and $(1-p)$ respectively.

Equation (19) introduces a force for substitutability between export markets: additional production pushes a firm up its marginal cost curve and reduces the profitability of further exports. If the model just featured one of the two interdependencies, then one could use a bounding algorithm, exploiting the respective supermodularity or submodularity of the cases. But, when I include both interdependencies, then I no longer have theoretical properties to leverage for optimization. Hence, I use the policy function approximation proposed by this paper to find (near) optimal solutions over a distribution of relevant parameter values of payoff relevant variables.

Indirect Evidence of Substitutability Empirically validating increasing marginal costs typically relies on data on total production/sales (Almunia et al., 2021) or prices and utilization rates (Boehm and Pandalai-Nayar, 2022). Since the EDD contains only export data, I present below a pattern of firm export behavior that could be interpreted as underlying substitutability between choices. If one believes that firms have increasing production costs as in equation (19), then there exists an optimal scale for each firm at which point marginal profits intersect marginal revenue. If a firm experiences a negative trade shock in one country that motivates exit from that market, then

[45] I use bilateral distance measures from Mayer and Zignago (2011). Alfaro-Urena et al. (2023) also projects costs onto language distances and regulatory distances, but the authors find that physical distances capture most of the fixed costs and interdependencies.

it should substitute to another market where it receives similar demand.[46]

I test if the set of markets which a firm is entering and exiting are more similar in terms of export revenue, when compared to the full set of entered markets for that firm. Specifically, for each firm-year pair I calculate the sample standard deviation in observed export revenues in two groups of countries. The first group, All Active Markets, includes export revenues in each active market for that firm-year. The second group, Added & Dropped Markets, includes export revenues in any newly added export market (i.e. this market was not serviced in the previous year) and export revenues from the previous period in any newly dropped export market (i.e. this market was serviced in the previous period but not the current one). Figure 3 plots the distribution of standard deviations across firm-year pairs for each group.[47] Note that there is lower variability in export revenues for the set of countries that is being added and dropped, indicating these markets are more similar than the set of countries typically being serviced by a firm.

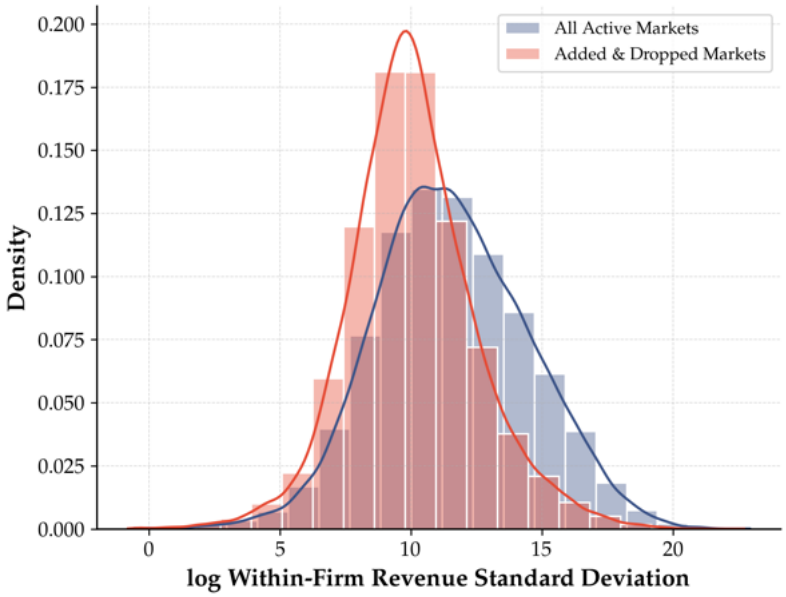


Figure 3: Evidence for Substitutability

Note. For each firm-year pair the standard deviation in revenues is calculated for two sets of export revenues. All Active Markets includes revenues for all markets active for that firm-year. Added & Dropped Markets includes only revenues from last period for dropped markets and revenues from the current period for newly added markets. Only All Market standard deviations are reported for firm-year observations where Added & Dropped Markets are available.

[46] This may not always be the case depending on the structure of fixed costs. A firm might have previously been exporting to a country with high fixed costs and high revenue potential. Now suppose a blocking shock prohibits exporting to this country. If the country with the closest revenue potential has even higher fixed costs that make participation unprofitable, the firm may choose to enter a country with lower fixed costs and entirely different revenue potential.

[47] Appendix Figure 11 plots the distribution of median mean-squared error.

8.2 Policy Function Training

Training a policy function to near-optimal performance requires only a small number of problem instances, generally less than 50,000. The training time per 10,000 problems is 2 seconds.[48] The policy functions are trained over a pre-specified distribution of structural parameters and revenue potentials, but the distances are fixed to only be those observed in the data. The policy function takes as input the revenue for each choice, distances, and a set of parameters $\{\theta, \alpha, \gamma, \phi, \beta, \eta\}$. The parameters that determine revenues, $\{\delta, \sigma, \rho_0, \rho_d\}$, do not directly enter the policy function, but the resulting revenues do.

The policy function generates policies within 0.01% of the global optimum.[49] In estimation, I use the greedy decoding with an optimality gap of 0.03% - marginally worse policies than other decoding strategies but faster compute time. The heuristic algorithms also perform well on this problem - a greedy solution has a 0.67% optimality gap. To understand why, note that the median firm in my setting exports to one location - typically the US. Therefore, interdependence between choices has limited scope to affect policies - most firms choose the location which on its own generates the greatest profits. A greedy solution will generally only fail when complementarities make a set of individually worse choices significantly better when made together. For example, despite the fact that the US might have high unilateral profit for a firm, it could make a higher profit by simultaneously exporting to several South American countries (e.g. Brazil, Venezuela, and Columbia) and enjoying the lower fixed costs generated by the model's complementarities. In practice, this situation is rarely encountered.

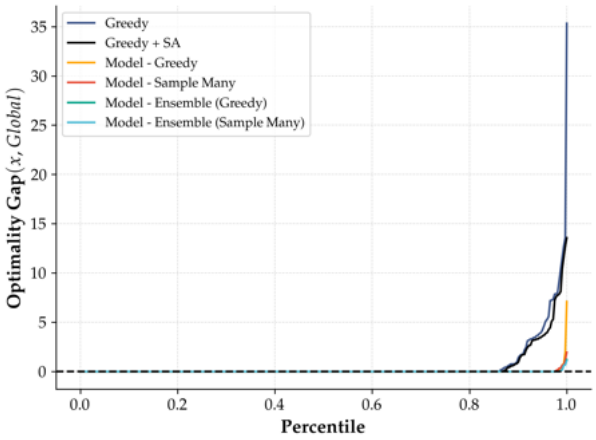
[48] See Section D.4 for examples of training curves and the range of parameters used.

[49] In Appendix Section D.3.4 I present optimality results for the $N = 50$ case. A global optimum is not available, but the Gap with a greedy algorithm is -11.19%.

Table 5: Performance - Summary

Method	$N = 20$		
	Cost	Gap	Time
Brute Force	-493.70	0.00%	(15m)
Greedy	-490.40	0.67%	(1s)
Greedy + SA	-491.16	0.51%	(10s)
Model - Greedy	-493.53	0.03%	(1s)
Model - Sample Many	-493.62	0.02%	(34s)
Model - Ensemble	-493.66	0.01%	(16s)

Figure 4: Performance - Distribution



Note. Time to complete 1,000 vectorized instances simultaneously. The optimality gap is between the listed method and Brute Force for all instances. Sample Many uses 25 samples. Ensemble uses 11 separately trained models. For the distribution of parameters used to evaluate optimality see Appendix Section D.4.4.

8.3 Estimation

I use both a Simulated Methods of Moments and Quasi-Bayesian estimator as described in Section 7. The moments used in estimation are largely drawn from Alfaro-Urena et al. (2023) with some modifications made since I am using a static model and have an additional parameter in θ. Moments are weighted based on first-step estimates. The details are in Appendix Section C, and the referenced paper has further details on the identification arguments behind this choice of moments. I estimate two versions of the model: one unrestricted which I refer to as the baseline model, and a second where I restrict θ to be 0. This second model features only positive complementarities between choices and could be solved using a bounding algorithm.

Table 6 reports estimates for both models and both estimators. The introduction of substitutability in choices alters the parameter estimates to make fixed costs, a, smaller and blocking shocks, p, more frequent. In a model with only complementarity, exporting begets more exporting. High fixed costs and frequent blocking shocks are necessary to keep firms from exporting to too many markets. By introducing increasing marginal costs, firms have an alternate limiting factor for exporting - their limited scale of production.

In Appendix Section C.6 I assess how well identified the parameters are by varying each parameter individually and plotting the change in the objective function. Generally, none of the complementarity or correlation parameters, $\{\beta, \gamma, \phi, \rho_0, \rho_d\}$, are precisely identified - featuring flat objective function variation over a wide range of parameter values. This exercise assures us that complementarities can not be that strong, if they were the number of export events and their geographical correlation would be much stronger. Otherwise, the scope for complementarities to affect

decisions is limited in the model. Firms export to only a few locations, and for changes in the strength of complementarity to induce changes in export behavior rely on very specific conditions. A firm has to be at the margin to enter a set of export markets which can only be profitable when jointly exported to.

Table 6: Estimates

Estimate	β	α	γ	θ	ϕ	σ	p	η	ρ_0	ρ_d	δ
Simulated Method of Moments											
Baseline	4.25	7.99	3.34	0.25	2.50	30.00	0.48	8.72	0.49	0.51	2.51
	(1.11)	(0.25)	(0.05)	(0.00)	(0.60)	(0.50)	(0.01)	(0.93)	(0.21)	(0.24)	(0.07)
$\theta = 0$	4.16	14.43	3.27	0.00	2.45	18.24	0.66	9.41	0.16	0.82	2.49
	(0.93)	(1.00)	(1.46)	(0.00)	(1.11)	(12.93)	(0.08)	(1.34)	(0.08)	(0.16)	(0.65)
Quasi-Bayesian											
Baseline	2.96	8.78	5.59	0.28	4.19	30.46	0.52	6.75	0.52	0.49	3.41
	(0.90)	(3.55)	(3.41)	(0.07)	(2.27)	(9.38)	(0.08)	(2.07)	(0.26)	(0.27)	(0.93)
$\theta = 0$	3.00	12.56	6.02	0.00	4.53	15.20	0.67	7.25	0.38	0.54	3.53
	(0.89)	(2.02)	(3.54)	(0.00)	(2.35)	(3.47)	(0.02)	(1.99)	(0.25)	(0.26)	(0.91)

Note. Baseline model estimates allow $\theta \in [0, 0.5]$ while $\theta = 0$ fixes θ to be 0. SMM standard errors are from 50 re-estimates of the model with bootstrapped simulation shocks. Quasi-Bayes standard errors are the standard deviation of the posterior distribution. See Appendix Section (C) for details.

Figure 6 plots the function $\gamma\left(1 + \phi d_{hj}\right) \exp\left(-\beta d_{kj}\right)$ where j is the country in the legend and d_{kj} is the distance represented by the horizontal axis.[50] The $\theta = 0$ model produces quantitatively similar results to Alfaro-Urena et al. (2023). To compare, note that Mexico and Costa Rica, the subject of the others' data, are 12.4 and 14.0 thousand kilometers from China. A firm which exports to China gets a 8 thousand USD savings from exporting to another country 500 km away from China. By the authors estimates, the savings for a firm in Costa Rica in the same situation appear to be roughly 14 thousand.[51]

Figure 5 plots the model-generated and observed export probabilities. These moments are not targeted in estimation, and therefore serve as a validation exercise. The mean-squared error between observed and model-generated export probabilities of the baseline model is 0.033 while that of the $\theta = 0$ model is 0.178. The baseline model struggles the most to generate sufficient exporting to distant markets in Europe and Asia. In Appendix Figure 9 I plot the revenue potential of a country against its distance to Mexico. In the data firms make very little revenue from participating

[50] The complementarity is represented in terms of 100s of USD, so it needs to be scaled down by a factor of 10 to get Figure 6.

[51] The curves I plot are quantitatively much different from the authors at distance less than 500km, but in the data the average country's nearest neighbor is about 500km so this is not a particularly relevant range of estimates.

in these distant markets - making it hard to generate the necessary patterns. There is likely some unaccounted for factor affecting exporting, e.g. foreign multinationals with plants in Mexico importing back to their home country.

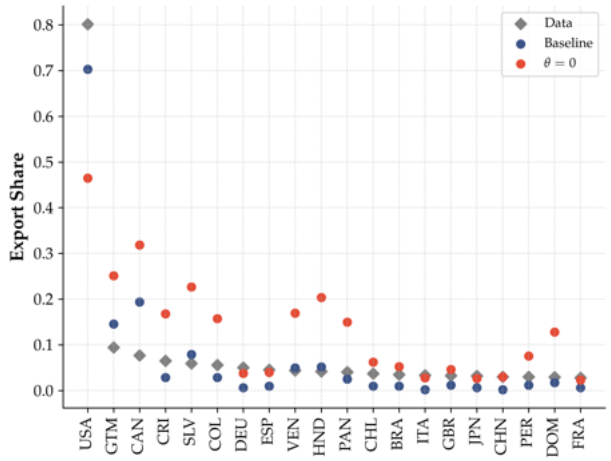


Figure 5: Model Fit

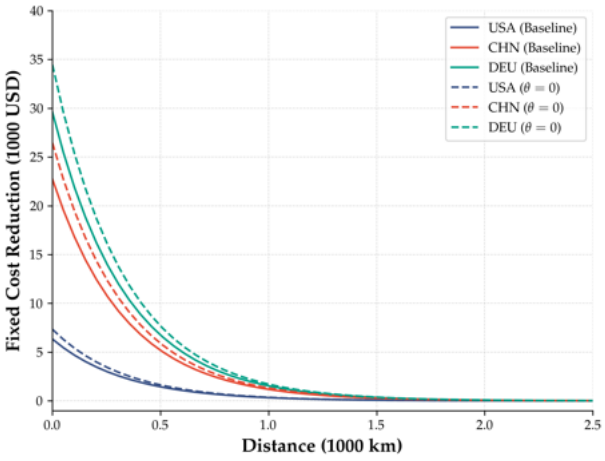


Figure 6: Complementarity Estimate

Note. The Export Share is the percent of exporters who have a destination in their export set. For example, 81% of exporting firms have the USA as an export destination and hence it has a value of 0.81.

8.4 Counterfactual

To quantify the importance of having increasing marginal costs in the model, I consider one counterfactual - I apply a negative revenue shock and blocking shock for exports to the US. Figure 7 plots the change in export probability by country under this counterfactual for the baseline and $\theta = 0$ model. First, note that in a model with only complementarities, there are two reasons to expect a change in exporting to non-US markets. A market is either dropped because its export shock is correlated with the US, or because exporting was previously only profitable due to a complementarity generated from simultaneously exporting to the US. There is no reason why a firm should start exporting to a location which it was previously not exporting to. By contrast, in the baseline mode with $\theta > 0$, when a firm drops the US from its export basket it can substitute its export capacity to another market. This leads to starkly different predictions, markets which see net exit under $\theta = 0$ now see large entry.

The presence of substitutability also generates heterogeneity in export response. Figure 8 plots the number of new entry and exit events by market in both models. To reiterate, a model with only complements can not generate new export events from a negative shock to another location. The same is not sure of the baseline model. There we see two broad types of changes in exporting. There are firms which were previously exporting to markets close to the US (e.g. Canada and Guatemala) because they enjoyed a complementarity from selling to all markets. When the US is

blocked, these markets are dropped - hence the large exit responses for Canada and Guatemala. On the other hand, there are firms which exported to the US, but did not have the capacity to export to additional markets. For these firms, dropping the US leads to entry of new markets - hence the large entry responses.

For a concrete example, let us consider a food processing facility in Mexico that sells agricultural products to the US market. The size of this firm is limited by its access to land, the cost of capital, local labor markets, and numerous other factors. It chooses to sell its limited production capacity to the US, the most proximate and largest market. If I block this export link, then the firm will sell its output to alternate markets like Canada or Guatemala. A model with only complementarities does not permit this kind of substitution.

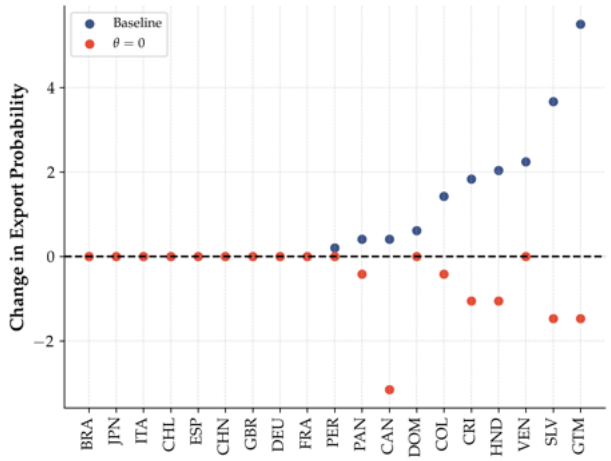


Figure 7: Change in Export Probability

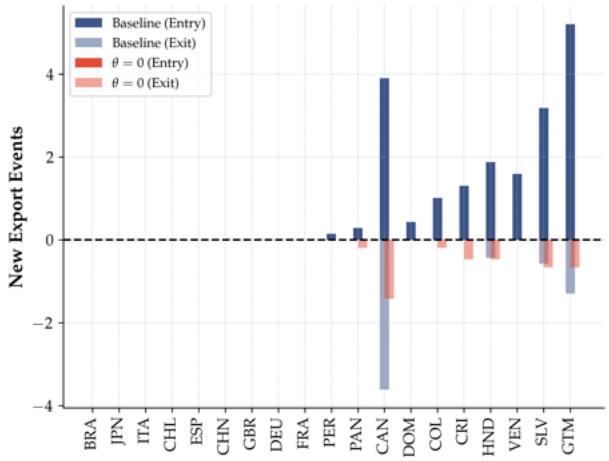


Figure 8: Change in Export Events

Note. The Change in Export Probability is calculated as the percentage point change in the same set of exporting firms. New Export Events are normalized so that a value of 1 is equivalent to 1% of total export events across all firms and markets.

9 Conclusion

Summary This project contributes a computational method for solving and estimating models of combinatorial choices. This method is competitive with existing solution techniques on problems in the trade literature, but is applicable to a wider class of models since it relies on no a-priori theoretical assumptions. This opens the door for richer quantitative trade modeling that can more realistically represent the decision making process of firms and provide new counterfactual insights, as I demonstrated in an example of export market entry. It also lowers the barrier of entry for developing new models on combinatorial choice, as they no longer have to be accompanied by a novel algorithm but can instead just employ this solver. The numerical results presented here also provide a useful benchmark for future research that attempts to improve on these methods or

explore other algorithms for solving combinatorial optimization problems in trade.

Ongoing Work Industrial Organization has developed progressively richer demand systems for discrete choice models. Such models generate more realistic substitution patterns, crucial for generating more accurate counterfactuals. Yet, in trade we often stick with variants of CES demand systems which have numerous theoretical advantages, but constrains the type of substitution patterns that our models can generate.[52] Since trade counterfactuals involve tariff changes, where substitution patterns are critical, IO-style demand systems likely have something to offer.

The model I propose is one of firms' demand for imports of various origins based on the characteristics of the trading parties. For example, consider a clothing distributor in the US selling inexpensive clothing. It imports from Bangladesh and India. One can view this transaction as the results of preference from a characteristic demand system with heterogeneous firms. In this case a firm from a particular sector, clothing, has a preference for a characteristic, low-skilled labor. Two countries abundant in this characteristic are Bangladesh and India.

Directly using tools from IO poses some challenges, demand estimation is largely developed to address discrete choices problems. The first challenge, firms demand a combination of products from a combination of countries - and the decision to import from any one country is likely related to the others. Imports may be complements or substitutes depending on characteristics of the countries and the importing firms. The second challenge, importing involves an extensive margin choice that endogenously generates zero input shares. This biases any estimates that just use intensive margin input shares to recover preferences. What this paper provides is a method that is able to reconcile the combinatorial problem faced at the extensive margin, particularly when faced with flexible substitution patterns or complementarities at the intensive margin. See Appendix Section E for a longer discussion that includes a stylized model, formalization of the estimation challenges, and sketch of an empirical model.

Directions for Future Research Most models of combinatorial choice in the trade literate are partial equilibrium - we model the decision making process of firms in one country with fixed factor prices. This setup is motivated partially by data - we have limited information on export and import policies of firms in many countries. But it is also a decision motivated by computational constraints - it is difficult to solve for a fixed point equilibrium that requires solving combinatorial optimization problems for many firms in many countries. These methods could open the door to richer general equilibrium models, by allowing us to compute equilibria in reasonable time.

All the models I have presented are static, but these methods can likewise be applied to dynamic choices. However, dynamic models like Alfaro-Urena et al. (2023) require one to make a sequence of a set of decisions, which increases the number of choices the model needs to make by a multiple

[52] Adao et al. (2017) estimates a random-coefficients demand system for imports where they use GDP-per capita as a dimension along which preferences vary. They do so using aggregated trade flows, which obscures the underlying extensive-margin decisions.

of the number of time periods. To avoid having to predict $N \times T$ decisions, I can model a policy function which makes N choices as a function of a sequence of future variables and roll out this policy T times - once for each period. I then optimize the policy function taking the cumulative discounted reward of these T rollouts.

Lastly, the machine learning literature evolves rapidly and there are thousands of new innovations to model architecture and training schemes that come out every year. I have tested one of many possible models and training algorithms, there are likely improvements to be made. This paper provides a starting point, but future implementations of these methods will have access to better models. For instance, there is recent work (e.g. Luo et al. 2023 and Sun and Yang 2023) on scaling to CO problems with 100s or 1000s of choices. In spatial settings where firms are making choices over a granular geography like cities or countries, these models can be applied. Relatedly, there is work to be done on leveraging more computational resources to enable larger problem instances and faster estimation. This paper's results are from one desktop workstation, but model training and estimation can all be parallelized across multiple GPUs - available either on university clusters or for rent from cloud service providers.

References

Adao, R., A. Costinot, and D. Donaldson (2017). Nonparametric Counterfactual Predictions in Neoclassical Models of International Trade. *The American Economic Review 107*(3), 633–689.

Alfaro-Urena, A., J. Castro-Vincenzi, S. Fanelli, and E. Morales (2023). Firm Export Dynamics in Interdependent Markets.

Almunia, M., P. Antràs, D. Lopez-Rodriguez, and E. Morales (2021, November). Venting Out: Exports during a Domestic Slump. *American Economic Review 111*(11), 3611–3662.

Antràs, P., E. Fadeev, T. C. Fort, and F. Tintelnot (2023). Exporting, Global Sourcing, and Multinational Activity: Theory and Evidence from the United States.

Antràs, P., T. C. Fort, and F. Tintelnot (2017, September). The Margins of Global Sourcing: Theory and Evidence from US Firms. *American Economic Review 107*(9), 2514–2564.

Antràs, P. and A. Gortari (2020). On the Geography of Global Value Chains. *Econometrica 88*(4), 1553–1598.

Arkolakis, C., F. Eckert, and R. Shi (2023). Combinatorial Discrete Choice: A Quantitative Model of Multinational Location Decisions.

Atalay, E. (2017, October). How Important Are Sectoral Shocks? *American Economic Journal: Macroeconomics 9*(4), 254–280.

Atkeson, A. and A. Burstein (2008, December). Pricing-to-Market, Trade Costs, and International Relative Prices. *American Economic Review 98*(5), 1998–2031.

Bello, I., H. Pham, Q. V. Le, M. Norouzi, and S. Bengio (2017, January). Neural Combinatorial Optimization with Reinforcement Learning.

Berry, S., J. Levinsohn, and A. Pakes (1995, July). Automobile Prices in Market Equilibrium. *Econometrica 63*(4), 841.

Boehm, C. E. and N. Pandalai-Nayar (2022, December). Convex Supply Curves. *American Economic Review 112*(12), 3941–3969.

Bottou, L. and O. Bousquet (2007). The Tradeoffs of Large Scale Learning. In *Advances in Neural Information Processing Systems*, Volume 20. Curran Associates, Inc.

Brian C., F., D. Ghose, and G. Khanna (2024, May). *Production Networks and Firm-Level Elasticities of Substitution*. Policy Research Working Papers. The World Bank.

Cappart, Q., E. Khalil, A. Lodi, C. Morris, and P. Veličković (2021). Combinatorial optimization and reasoning with graph neural networks.

Castro-Vincenzi, J. (2022). Climate Hazards and Resilience in the Global Car Industry.

Chernozhukov, V. and H. Hong (2003, August). An MCMC Approach to Classical Estimation. *Journal of Econometrics 115*(2), 293–346.

Dai, H., E. B. Khalil, Y. Zhang, B. Dilkina, and L. Song (2017). Learning Combinatorial Optimization Algorithms over Graphs.

de Gortari, A. (2020). Global Value Chains and Increasing Returns.

Dingel, J. I. and F. Tintelnot (2020, May). Spatial Economics for Granular Settings.

Duarte, V. and J. Fonseca (2023). Global Identification with Gradient-Based Structural Estimation.

Fernandes, A. M., C. Freund, and M. D. Pierola (2016, March). Exporter behavior, country size and stage of development: Evidence from the exporter dynamics database. *Journal of Development Economics 119*, 121–137.

Fernández-Villaverde, J., S. Hurtado, and G. Nuño (2023). Financial Frictions and the Wealth Distribution. *Econometrica 91*(3), 869–901.

Gurobi Optimization, LLC. (2024). Gurobi Optimizer Reference Manual. Technical report.

Head, K., T. Mayer, M. Melitz, and C. Yang (2024). Industrial policies for multi-stage production: The battle for battery-powered vehicles.

Helpman, E. and B. C. Niswonger (2020). Dynamics of Markups, Concentration and Product Span.

Hodgson, C. and G. Lewis (2023, September). You Can Lead a Horse to Water: Spatial Learning and Path Dependence in Consumer Search.

Hopfield, J. J. and D. W. Tank (1985, July). "Neural" computation of decisions in optimization problems. *Biological Cybernetics 52*(3), 141–152.

Hornik, K., M. Stinchcombe, and H. White (1989, January). Multilayer feedforward networks are universal approximators. *Neural Networks 2*(5), 359–366.

Houde, J.-F., P. Newberry, and K. Seim (2023). Nexus Tax Laws and Economies of Density in E-Commerce: A Study of Amazon's Fulfillment Center Network. *Econometrica 91*(1), 147–190.

Huang, J. and J. Yu (2024, February). Applications of Deep Learning-Based Probabilistic Approach to.

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

Kool, W., H. van Hoof, and M. Welling (2019). Attention, Learn to Solve Routing Problems! *ICLR*.

Kreindler, G., A. Gaduh, T. Graff, R. Hanna, and B. A. Olken (2023, June). Optimal Public Transportation Networks: Evidence from the World's Largest Bus Rapid Transit System in Jakarta.

Kwon, Y.-D., J. Choo, B. Kim, I. Yoon, Y. Gwon, and S. Min (2021, July). POMO: Policy Optimization with Multiple Optima for Reinforcement Learning.

Liu, J. (2024). Multinational Production and Innovation in Tandem.

Luo, F., X. Lin, F. Liu, Q. Zhang, and Z. Wang (2023). Neural Combinatorial Optimization with Heavy Decoder: Toward Large Scale Generalization.

Mayer, T. and S. Zignago (2011). Notes on CEPII's Distances Measures: The GeoDist Database. *SSRN Electronic Journal*.

Mazyavkina, N., S. Sviridov, S. Ivanov, and E. Burnaev (2020). Reinforcement Learning for Combinatorial Optimization: A Survey.

Melitz, M. J. (2003). The Impact of Trade on Intra-Industry Reallocations and Aggregate Industry Productivity. *Econometrica 71*(6), 1695–1725.

Oberfield, E., E. Rossi-Hansberg, P.-D. Sarte, and N. Trachter (2024, March). Plants in Space. *Journal of Political Economy 132*(3), 867–909.

Sun, Z. and Y. Yang (2023). DIFUSCO: Graph-based Diffusion Solvers for Combinatorial Optimization.

Sutton, R. S., D. McAllester, S. Singh, and Y. Mansour (1999). Policy Gradient Methods for Reinforcement Learning with Function Approximation. In *Advances in Neural Information Processing Systems*, Volume 12. MIT Press.

Tintelnot, F. (2017). Global Production with Export Platforms. *The Quarterly Journal of Economics 132*(1), 157–209.

Tolstikhin, I., N. Houlsby, A. Kolesnikov, L. Beyer, X. Zhai, T. Unterthiner, J. Yung, A. Steiner, D. Keysers, J. Uszkoreit, M. Lucic, and A. Dosovitskiy (2021, June). MLP-Mixer: An all-MLP Architecture for Vision.

Vaswani, A., N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin (2017). Attention Is All You Need.

Vesselinova, N., R. Steinert, D. F. Perez-Ramirez, and M. Boman (2020). Learning Combinatorial Optimization on Graphs: A Survey with Applications to Networking. *IEEE Access 8*, 120388–120416.

Vinyals, O., M. Fortunato, and N. Jaitly (2015). Pointer Networks. In *Advances in Neural Information Processing Systems*, Volume 28. Curran Associates, Inc.

Watkins, C. J. C. H. and P. Dayan (1992, May). Q-learning. *Machine Learning 8*(3), 279–292.

Williams, R. J. (1992, May). Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning 8*(3), 229–256.

Yu, W., M. Luo, P. Zhou, C. Si, Y. Zhou, X. Wang, J. Feng, and S. Yan (2022). MetaFormer Is Actually What You Need for Vision.

A Machine Learning Model Details

The model architecture I employ is largely drawn from Kool et al. (2019). This section outlines the precise model structure.

A.1 Encoder

Let $\ell \in \{1, ..., L\}$ denote the layers of the network and $i \in \{1, ..., N\}$ a choice in the optimization problem. There is an initial embedding

$$\boldsymbol{h}_i^{(0)} = W^x x_i + b^x \tag{A.20}$$

where if x_i has d_x dimensions, then W^x has dimensionality $d_x \times d_e$ where d_e is the embedding size. There is then several attention layers which each contains the following operations:

$$\begin{aligned} \hat{\boldsymbol{h}}_i =& BN^\ell \left(\boldsymbol{h}_i^{(\ell-1)} + MHA_i^\ell \left(\boldsymbol{h}_1^{(\ell-1)}, ..., \boldsymbol{h}_n^{(\ell-1)} \right) \right) \\ \boldsymbol{h}_i^{(\ell)} =& BN^\ell \left(\hat{\boldsymbol{h}}_i + FF^\ell \left(\hat{\boldsymbol{h}}_i \right) \right) \end{aligned} \tag{A.21}$$

The attention layers contains three components. First, there is the multi-head attention $MHA\,()$, second there is the batch normalization $BN\,()$, and finally there is the feed forward network $FF\,()$. The multi-head attention starts by comping a query, key, and value which are the product of the embedding with a matrix of size $d_e \times d_e$.

$$\boldsymbol{q}_i = W^Q \boldsymbol{h}_i, \quad \boldsymbol{k}_j = W^K \boldsymbol{h}_j, \quad \boldsymbol{v}_j = W^V \boldsymbol{h}_j \tag{A.22}$$

I split the query, key, and value into various heads by subsetting the dimensions. So if $d_e = 128$ and there are 8 heads, then the query, key, and value for the mth head have size $d_m = 16$ are are the $(m-1) \times 16 + 1$ through $m \times 16$ dimensions of the full query, key, and value. For each head I compute new embedding with the following sequence of operations:

$$u_{ij}^m = \frac{\boldsymbol{q}_i^m \boldsymbol{k}_j^m}{\sqrt{d_k}} \tag{A.23}$$

$$a_{ij}^m = \frac{e^{u_{ij}^m}}{\sum_{j'} e^{u_{ij'}^m}} \tag{A.24}$$

$$\boldsymbol{h}_i^m = \sum_j a_{ij}^m \boldsymbol{v}_j^m \tag{A.25}$$

I then collect the output of each head back into an embedding of size d_e, multiply it by another matrix of size $d_e \times d_e$ to conclude the MHA operation

$$MHA_i\left(\boldsymbol{h}_1, ..., \boldsymbol{h}_N\right) \equiv W^H \left\{\boldsymbol{h}_i^1, ..., \boldsymbol{h}_i^M\right\} + b^H. \tag{A.26}$$

The batch normalization, $BN\left(\right)$, normalizes the embedding values by subtracting the mean value across dimensions and dividing with the variance. The resulting values are then rescaled and transformed with learned parameters. Finally, the feed-forward consists of

$$FF\left(h_i\right) = W^2 \text{ReLu}\left(W^1 h_i\right)$$

where W^1 has dimensionality $d_e \times d_f$ and W^2 has $d_f \times d_e$. The function ReLu is known as an activation function which just takes $\max\left(0, x\right)$ along each dimension of the transformed embedding.

A.1.1 Edge Information

Suppose there was bilateral information I had in the form of a tensor y_{ij}. I can likewise embed this information:

$$\boldsymbol{\tau}_{ij} = W^y y_{ij}$$

where W^y has dimensionality $d_y \times d_e$ and d_y is the number of edge-level variables. I can either use this in the attention mechanism

$$u_{ij}^m = \frac{\boldsymbol{q}_i^m \boldsymbol{k}_j^m \boldsymbol{\tau}_{ij}^m}{\sqrt{d_k}},$$

or I can have separate edge embeddings that I iteratively update alongside the node embeddings.

A.2 Decoder

I define the aggregate node embedding as the mean embeddings from the final layers of the encoder:

$$\bar{\boldsymbol{h}}^L = \frac{1}{n} \sum_{i=1}^{N} \boldsymbol{h}_i^{(L)}. \tag{A.27}$$

Let $h_{\pi_{t-1}}$ and h_{π_1} denote the encoder embeddings for the previous and initial policy choices respectively. Additionally, ket D_t denote additional dynamic components of the context and let d_{jt} denote dynamic components of node j, The context vector is then given by concatenating these vectors

$$\bar{\boldsymbol{h}}_c = \begin{cases} \left\{\bar{\boldsymbol{h}}^L, h_{\pi_{t-1}}, h_{\pi_1}, D_t\right\} & \text{if } t > 1 \\ \left\{\bar{\boldsymbol{h}}^L, \boldsymbol{v}^a, \boldsymbol{v}^b, D_t\right\} & \text{if } t = 1 \end{cases} \tag{A.28}$$

with $\boldsymbol{v}^a, \boldsymbol{v}^b$ as learnable vectors for the initial choice.

A.2.1 Dynamic Potential Decoding

To compute the choice probabilities I again use a final multi-headed attention layer. Here I add a component to the decoder I refer to as dynamic potential. Given the current state s_{t-1} I compute the change in the objective under each possible addition to the policy. That is, I compute

$$d_{j,t} = R\left(s_{t-1}, \pi_j\right)$$

where $R\left(s_{t-1}, \pi_j\right)$ is the reward function described in Section 4. I then compute the query, key, and value combining the embedding with this dynamic component:

$$\boldsymbol{q}_c = W^Q \bar{\boldsymbol{h}}_c, \quad \boldsymbol{k}_{j,t} = W^K \boldsymbol{h}_j + W^{Kd} d_{j,t}, \quad \boldsymbol{v}_{j,t} = W^V \boldsymbol{h}_j + W^{Kd} d_{j,t} \tag{A.29}$$

Using multi-head attention once again I get an updated query:

$$u^m_{cj,t} = \frac{\boldsymbol{q}^m_c \boldsymbol{k}^m_{j,t}}{\sqrt{d_k}} \tag{A.30}$$

$$a^m_{cj,t} = \frac{e^{u^m_{cj,t}}}{\sum_{j'} e^{u^m_{cj',t}}} \tag{A.31}$$

$$\boldsymbol{h}^m_{c,t} = \sum_j a^m_{cj,t} \boldsymbol{v}^m_j \tag{A.32}$$

$$\bar{\boldsymbol{h}}^F_{c,t} \equiv W^H \left\{\boldsymbol{h}^1_{c,t}, ..., \boldsymbol{h}^M_{c,t}\right\} + b^H \tag{A.33}$$

And finally, I get choice probabilities using this query and the previously computed keys:

$$\boldsymbol{q}_{c,t} = W^Q \bar{\boldsymbol{h}}^F_{c,t} \tag{A.34}$$

$$u_{cj,t} = \frac{\bar{\boldsymbol{h}}^F_{c,t} \boldsymbol{k}_{j,t}}{\sqrt{d_k}} \tag{A.35}$$

$$p\left(\pi_t = j | \boldsymbol{\pi}_{1:t-1}, \{\boldsymbol{h}_i\}\right) = \frac{e^{u_{cj,t}}}{\sum_{j'} e^{u_{cj,t'}}} \tag{A.36}$$

B Additional Computational Details

B.1 Baselines

Recall the objective for policy function training:

$$\max_{\Theta} \mathbb{E}_{\boldsymbol{s}} \Big[\mathbb{E}_{p(\pi|\boldsymbol{s};\Theta)} \left[V_\pi\left(\boldsymbol{s}\right) - b\left(\boldsymbol{s}\right) \right] \Big].$$

The baseline, $b(\boldsymbol{s})$, constructs a separate solution for the optimization problem that I use to benchmark performance for the current iteration of the policy function. This helps in optimization by reducing the variance of the gradient and helping identify when parameters should be increased or decreased. The simplest baseline I experiment with is the exponential baseline, which tracks previous values of $V_\pi(s_0)$ and computes $b(s_0)$ as a weighted average of past realizations of $V_\pi(s_0)$. The second baseline, developed by Kool et al. (2019), is known as the greedy rollout baseline. Recall that the policy function is a stochastic policy function that returns a probability weight over optimal policies. The greedy rollout baseline involves sampling policies from the policy function to determine $V_\pi(\boldsymbol{s})$, and taking the policy with highest probability as $b(\boldsymbol{s})$. A natural extension of this which I develop is a sample many rollout baseline, which samples several policies from the policy function, takes the best one, and uses that as the baseline. This is the baseline I use for ISLP - Oligopoly 100, and hence why I see poor greedy performance in that problem. But note a sample-many baseline requires as many additional iterations of the model as number of samples, this meaningfully reduces training speeds, as seen in Table 4.

Another baseline I use is known as Policy Optimization with Multiple Optima (POMO) from Kwon et al. (2021). This baseline leverages the fact that the model prescribes a sequence of actions, but some CO problems require only a set of actions where the order of actions does not matter. Therefore, I can force the policy function to start from separate initial actions, and if any initial actions are part of the optimal set, then ideally the policy function should still return the same optimal set despite the different starting points. The baseline is then the average value of the objective function across all initializations of the policy function. For a detailed discussion of the advantages of this baseline please see the reference work above.

There are two, related issues with using POMO that I attempt to innovate around. The first, is that you should ideally only include in the baseline the initial actions that will be part of the final optimal solution. This is not an issue for CO problems like the traveling salesman problem where every choice will be part of the optimal solution, but in my case it remains. A secondary problem with having too many initial actions in the baseline is the computational memory cost. For every problem instance I consider I have to keep track of the gradient of the final outcomes with respect to the 100s of thousands of parameters of the model. But since I are now considering multiple evaluations of the same problem, every time I sample a problem to approximate the outer expectation $\mathbb{E}_{\boldsymbol{s}}$ I need up to 100 times the memory in the $N = 100$ case since there is a gradient associated with each initial starting point. This additionally becomes a problem come estimation, since I have to again compute a policy from many starts which raises computation time.

I proceed with a strategy similar to Dynamic Potential Decoding. I first compute the marginal benefit of each choice on its own:

$$d_j = R(s_0, \pi_j).$$

I then assign a probability to each action according to

$$Pr\left(\pi_1 = \pi_j\right) = \frac{\exp\left(\gamma d_j\right)}{\sum_{j'} \exp\left(\gamma d_{j'}\right)}$$

where γ is a scaling parameter. I then sample starting points from this distribution, where the number sampled is less than N. This raises the probability that I have good initial starts and lower the number of starts I have to compute with.

Another strategy I employ is occasionally using a heuristic algorithm as the baseline, particularly during the start of training. I set a probability that the current batch of sampled states uses a heuristic algorithm as baseline and allow this probability to go to 0 as training progresses. This is almost like doing supervised learning at the start, and once I have a moderately informed policy function, allowing it to explore better possible policies.

B.2 Using Theory to Guide Optimization

Even when one can not analytically characterize a solution to a combinatorial optimization problem, theory might imply certain properties that the solution should satisfy. For example, let us introduce a firm-level productivity shifter $\varphi > 0$ to the input sourcing and plant location problem of Section 3.1:

$$\max_{\{\mathbb{1}_i\} \in 2^N} \left(\sum_i \mathbb{1}_i \varphi T_i \right)^{\alpha} - \sum_i \mathbb{1}_i f_i.$$

Antràs et al. (2017) provides the following proposition: when $\alpha > 1$, the sourcing set is increasing in φ - i.e. a more productive firm sources from at least all the locations of a less productive firm. I can then add an additional term to the objective:

$$\text{Theory Loss}\left(\Theta|\boldsymbol{s}\right) = -\mathbb{E}_{p(\pi|\boldsymbol{s};\Theta)} \left[\sum_{i=1}^{N} \left(\pi_i \left(\{\{T_i, f_i\}, \alpha\} \right) - \pi_i \left(\{\{\bar{\varphi} T_i, f_i\}, \alpha\} \right) \right) \mathbb{1}\left\{\alpha > 1\right\} \right]$$

where $\bar{\varphi} > 1$ and π_i takes a value of 1 if choice i is made as part policy π. This loss takes non-zero values when a firm with lower productivity makes choices that a firm with higher productivity would not make, conditional on $\alpha > 1$. Adding this term to the objective ensures I am optimizing the original objective function and satisfying the theoretical predictions of the model. In principle, this loss, by placing additional constraints on the policy function, should guide the optimization of the policy function to a subset of the policy function space where the optimal policy ought to exist. Appendix Section D.2 presents numerical experiments for optimization using theory loss.

B.3 Supervised Learning

The application of machine learning models to approximate rich functions in economics is not new. Most commonly, neural networks are used in regression tasks where I want to predict an outcome given covariates. These models are trained via supervised learning - a training paradigm where I have access to a training dataset of desired output. Formally, if one observed states $\boldsymbol{s}$ and optimal policies for those state $\pi^*(\boldsymbol{s})$, then one would be solving

$$\max_{\Theta} \mathbb{E}_{\boldsymbol{s}\sim F(\boldsymbol{s})}\Big[\mathbb{E}_{\pi\sim p(\pi|\boldsymbol{s};\Theta)}\left[D\left(\pi^*\left(s\right),\pi\right)\right]\Big],$$

where D is some distance measure between two policies. In this project, I do not have access to the optimal policy and instead am using a simulated reward scheme to guide a policy function to an optimal solution.

There is a role for supervised learning in finding faster solutions to problems for which we have alternate, but slower means of generating optima. For instance, Vinyals et al., 2015 trains a policy function for the Traveling Salesman Problem where a specialized algorithm was used to generate a training dataset of optima. As one possible application in economics, Alfaro-Urena et al. (2023) reports a compute time of over 13 minutes for finding the optimal solution to their dynamic exporting problem. A training dataset could be generated by computing solutions using the author's algorithm, and a model could be fit to replicate these solutions given the relevant inputs.

B.4 Value-Based Learning

In value-based learning we parametrize an action-value function known as a Q function.The Q function maps the current state and an action taken to some value consisting of an immediate reward and the continuation value from corresponding actions:

$$Q\left(s_t,a_t\right)=R\left(s_t,a_t\right)+\max_{a_{t+1}}Q\left(s_{t+1},a_{t+1}\right).$$

The value function naturally follows as $V\left(s_t\right)\equiv\max_{a_t}Q\left(s_t,a_t\right)$. We posit a parametric function $Q\left(\cdot;\Theta\right)$ and choose the parameters Θ so to optimize

$$\min_{\Theta}\mathbb{E}_{s,a}\left(Q\left(s,a;\Theta\right)-\left(R\left(s,a\right)+\max_{a'}Q\left(s',a';\Theta\right)\right)\right)^2.$$

If we are able to approximate the true Q, then the optimal policies can be derived by iteratively maximizing Q at each state. This optimization problem ends up being relatively easy since $Q\left(\cdot|\Theta\right)$ is specifically designed to be easily differentiable with respect to Θ.

Early work on CO with ML models started with value-based learning - notably Dai et al. (2017). The literate has since convergence on policy-based learning. First, the policy function is much closer to the object of interest for someone concerned with optimization, so modeling it directly can involve

less overhead than modeling a Q function which needs to understand off-optimal value-action pairs. Second, by modeling a stochastic policy function, I can directly incorporate experimentation in the learning process by sampling from the policy function. It also helps with inference by allowing the model to direct the search down multiple possible solutions when there exists local minima with similar objective values.

B.5 Alternative Estimation Strategies

In practice, we often use a simulated method of moments procedure to estimate structural parameters in CO problems where we match moments of the model generated policy distribution to observed combinatorial choices. A problem with this approach however is that the resulting objective function may not be smooth. Namely, a small change in the parameter of interest may not induce any policy changes, since we only simulate a finite set of actions and each action is discrete, and hence may result in a zero gradient. As a possible workaround, I can use the insights of Duarte and Fonseca (2023) where a second model is fit to predict the relationship between parameters and moments. This requires a training dataset which would be generated by evaluating the policy function on a grid of relevant parameter values. This second model would be easier to optimize over due to this differentiability and faster evaluation time.

B.6 Simulated Annealing

In simulated annealing I stochastically update the policy based on how deviations improve the objective value. I start by setting some initial temperature T_0 and initializing the policy to some sequence of actions: $\{\pi_1, ..., \pi_T\}$. I then randomly sample an element of the policy to change to a different choice. I then evaluate the difference in the objective value under these two policies:

$$\text{Energy Difference} = V(\pi') - V(\pi).$$

I accept the change with probability

$$Pr(\text{Accept}) = \min\left\{\exp\left(\frac{\text{Energy Difference}}{T}\right), 1\right\}$$

If the energy difference is positive, then I always accept otherwise we accept with some probability. I then update $T' = T \times \alpha$ where $\alpha < 1$ to lower the probability that a negative innovation is accepted.

I choose the initial temperature as follows. I first implement a greedy solution to the CO problem at hand. I then calculate the standard deviation of objective values, σ, and set

$$T_0 = \frac{\sigma}{|\log(0.2)|}.$$

This means that a change which decreases the objective value by 1 standard deviation is accepted with a probability of 20%. I then choose $\alpha = 0.9995$ and update T only every 5 iterations so that after 10000 iterations the probability of that same 1 standard deviation change being accepted is $100 * exp(-1/(0.9995^{2000} \times T_0)) \approx 1\%$. One issue with initializing in this way is that I call σ the standard deviation in costs, but when problem instances are drawn with varying structural parameters making a comparison in objective values across problem instances can be somewhat meaningless. To be more precise, one can set different simulated annealing parameters based off of a more relevant distribution of payoffs for the particular problem instance at hand.

A final note on using simulated annealing in combination with bounding algorithms. The lower bound tells you which elements must be in the solution and the upper bound tells you which remaining elements are eligible to be in the solution. I use this information to force lower bound choices to be in the solution, and only allow choices in the upper bound to be considered during optimization.

B.7 Hyper-Parameter Search

In Table (4) I reported the model sizes for which there is considerable heterogeneity. This arises from the fact that I use slightly different model structure in each model. The model structure varies as a function of hyper-parameters which control the number of embedding dimensions, the number of encoder layers, the size of hidden layers in the feed-forward layer, the presence of dynamic decoding components, and the presence of edge attention mechanisms. There are additionally hyper-parameters that shape the speed and variability of training. This includes the batch size, the POMO size, the use of gradient and probability clipping, the learning rate, the decay of the learning rate, entropy regularization, the baseline used, and the probability of using various heuristics as replacement for the baselines. I perform a grid search over these hyper-parameters with the final optimality gap on a validation dataset used to distinguish the best combination.

C Additional Estimation Details

C.1 Functional Form for Profits

Claim. A profit function concave in revenues can be micro-founded with increasing marginal costs.

Proof. Suppose the exporting firm faces a fixed international price which I normalize to 1. Let the marginal production cost be given by $mc(q) = \left(1 - q^{-\theta}\right)$ where $q \geq 1$ and $\theta \in (0, 1)$. Integrating costs from 1 to q, I get $c(q) = \left(1 - q^{1-\theta}\right) / (1 - \theta) + q - 1$. I can write profits as $\pi = q - c(q)$, which yields $\pi = \left(r^{1-\theta} + \theta\right) / (1 - \theta)$.

C.2 Firm and Country Size Distribution

The World Exporter Dynamics Database gives data at the firm-product-country level for export revenues. Since I only observe export revenues in active destinations, I impute revenue potentials by estimating the following model where I aggregate revenues to the firm-country level:

$$\log r_{fj} = \delta_f + \delta_j + \varepsilon_{ij}.$$

Here f indexes the firm and j the destination. Figure 9 plots the country fixed effect δ_j as a function of distance from Mexico, and Figure 10 plots the firm fixed effect δ_f.

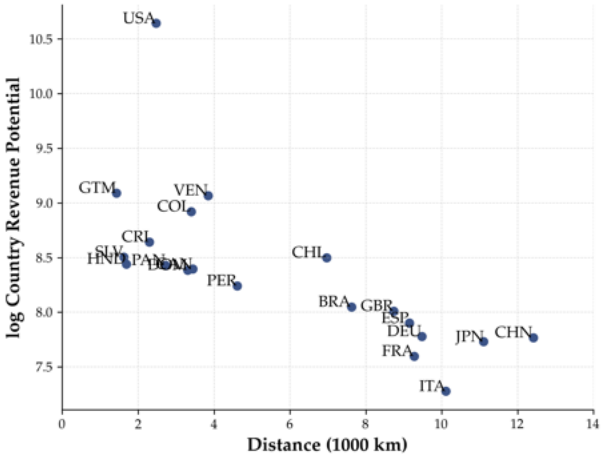


Figure 9: Country Potential & Distances

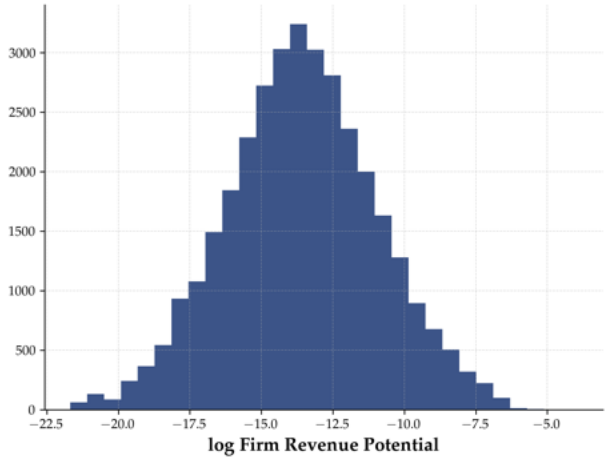


Figure 10: Firm Potential

C.3 Alternative Measurement of Firm Choice Variability

Figure 11 plots the distribution of median mean-squared error for each firm-year in both groups.

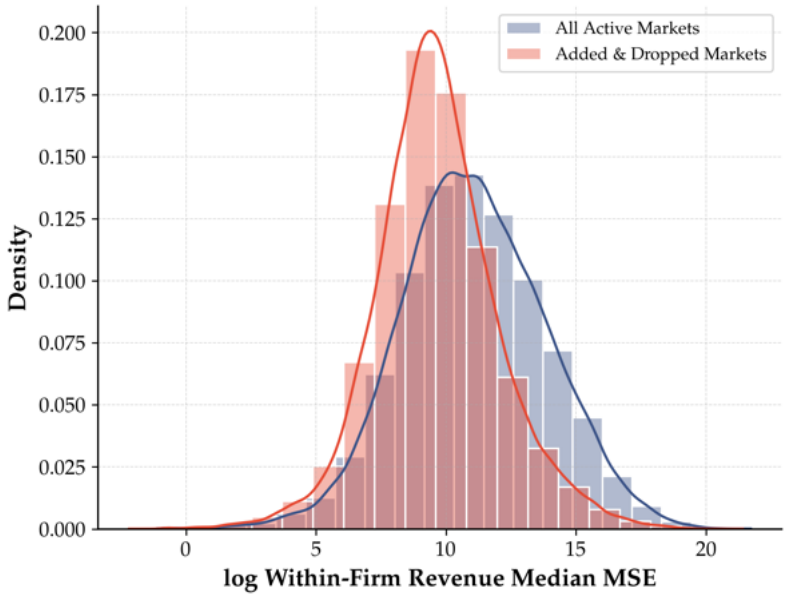


Figure 11: Evidence for Substitutability

C.4 SMM Objective

The moments I target in estimation are largely based off of Alfaro-Urena et al. (2023), with modifications made for the country of study and the fact that I only consider one distance measure. The paper has some identification intuition for each of these moments.

The first set of moments is the average distance of active markets in two categories of destinations

$$m_{fj} = y_{fj} \mathbb{1}\left\{d_{hj} < \bar{d}_{hj}\right\} d_{hj}$$
$$m_{fj} = y_{fj} \mathbb{1}\left\{d_{hj} \geq \bar{d}_{hj}\right\} d_{hj}$$

I use $\bar{d}_{hj} = 8$ which roughly separate counties into North and South America for $d_{hj} < \bar{d}_{hj}$ and then Europe and Asia for $d_{hj} \geq \bar{d}_{hj}$,.

The second set of moments looks at the propensity to export to locations which are close to other high-value markets. For this I define the aggregate export potential of a location as

$$AE_j^{x_2} = \sum_{j' \neq j} \mathbb{1}\left\{\underline{d}_{hj}^{x_2} \leq d_{hj} < \bar{d}_{hj}^{x_2}\right\} E_{j'}$$

where $E_{j'}$ the country fixed effect recovered from regressing export revenues on firm and country

fixed effects. The distance thresholds $\left\{\underline{d}_{hj}^{x_2}, \bar{d}_{hj}^{x_2}\right\}$ are indexed by x_2 and are given by

$$\left\{\underline{d}_{hj}^{x_2}, \bar{d}_{hj}^{x_2}\right\} = \begin{cases} (0, 0.8) & \text{if } x_1 = 1, \\ (0.8, 1.6) & \text{if } x_2 = 2, \\ (1.6, 2.4) & \text{if } x_2 = 3. \end{cases}$$

I then consider the export probability for countries which are nearby and far as a function of their export potential:

$$\begin{aligned} m_{fj} =& y_{fj} \mathbb{1}\left\{d_{hj} < \bar{d}_{hj}\right\} \mathbb{1}\left\{AE_j^{x_2} = 0\right\} \\ m_{fj} =& y_{fj} \mathbb{1}\left\{d_{hj} < \bar{d}_{hj}\right\} \mathbb{1}\left\{0 < AE_j^{x_2} \leq p_{66}\left(AE_j^{x_2}\right)\right\} \\ m_{fj} =& y_{fj} \mathbb{1}\left\{d_{hj} < \bar{d}_{hj}\right\} \mathbb{1}\left\{AE_j^{x_2} > p_{66}\left(AE_j^{x_2}\right)\right\} \\ m_{fj} =& y_{fj} \mathbb{1}\left\{d_{hj} \geq \bar{d}_{hj}\right\} \mathbb{1}\left\{AE_j^{x_2} = 0\right\} \\ m_{fj} =& y_{fj} \mathbb{1}\left\{d_{hj} \geq \bar{d}_{hj}\right\} \mathbb{1}\left\{0 < AE_j^{x_2} \leq p_{66}\left(AE_j^{x_2}\right)\right\} \\ m_{fj} =& y_{fj} \mathbb{1}\left\{d_{hj} \geq \bar{d}_{hj}\right\} \mathbb{1}\left\{AE_j^{x_2} > p_{66}\left(AE_j^{x_2}\right)\right\} \end{aligned}$$

This gives me 6 moments for each x_2 and hence 18 moments in total.

The third set of moments get the correlation in export choices across countries of similar firm export potential

$$m_{fj} = y_{fj} \frac{1}{N} \sum_{f'} y_{f'j} \mathbb{1}\left\{Q\left(R_f\right) = Q\left(R_f\right)\right\}$$

where E_i is the firm fixed effect recovered from regressing export revenues on firm and country fixed effects. This gives me 4 moments, one for each quartile. The fourth set of moments get the correlation in export choices across countries of similar export potential and proximity:

$$m_{fj} = y_{fj} \frac{1}{N} \sum_{j'} y_{fj'} \mathbb{1}\left\{Q\left(E_{fj}\right) = Q\left(E_{fj'}\right)\right\} \mathbb{1}\left\{\underline{d}_{hj}^{x_2} \leq d_{jj'} < \bar{d}_{hj}^{x_2}\right\}.$$

This gives me 3 moments, one for each x_2. I compute the mean value of m_{fj} across all locations and firms. The final set of moments is the mean and variance in the number of destination markets. The estimation objective is then to minimize the distance between model generated and observed moments, weighted appropriately. I use the DIRECT as a global optimizer followed by a local Nelder-Mead optimizer.

Weighting Matrix I perform two-step SMM and later two-step Quasi-Bayesian estimation, where moment conditions are first weighted by the inverse square of the moment value observed in the data. In the second step, I weight by the inverse of the covariance matrix for moments generated by the first-step $\theta = 0$ model. This matrix is used for both the second-step baseline and $\theta = 0$ model. The estimation exercise is meant to address how our estimates would differ if we incorrectly specified our model featuring complementarities only. Therefore, I use a common weighting matrix.

C.5 Sampling and Standard Errors

There are two sources of sampling variability. First, since there are 33,901 firms I will not solve a policy for each firm, but rather a sample of these firms. When I sample a firm, this just means I use their firm-specific distribution of revenue potentials. Second, when I sample one of the firms in the data, I will additionally draw a set of revenue and blocking shocks for each location. In estimation, I draw 8,000 firms from the data and then draw shocks for each of these firms.

To get SMM standard errors, I keep the set of 8,000 firms fixed, but I bootstrap the revenue and blocking shocks and re-estimate the model. For Quasi-Bayes, I keep the 8,000 firms and their shocks fixed. The standard errors are then from the posterior distribution of the estimated parameters. Therefore, the standard errors reported for both estimators quantify separate types of uncertainty.

C.6 Identification

In order to assess if the estimated parameters are properly identified, i.e. uniquely minimizing the estimating objective function, I vary each parameter individually over the domain I restrict the estimator to.

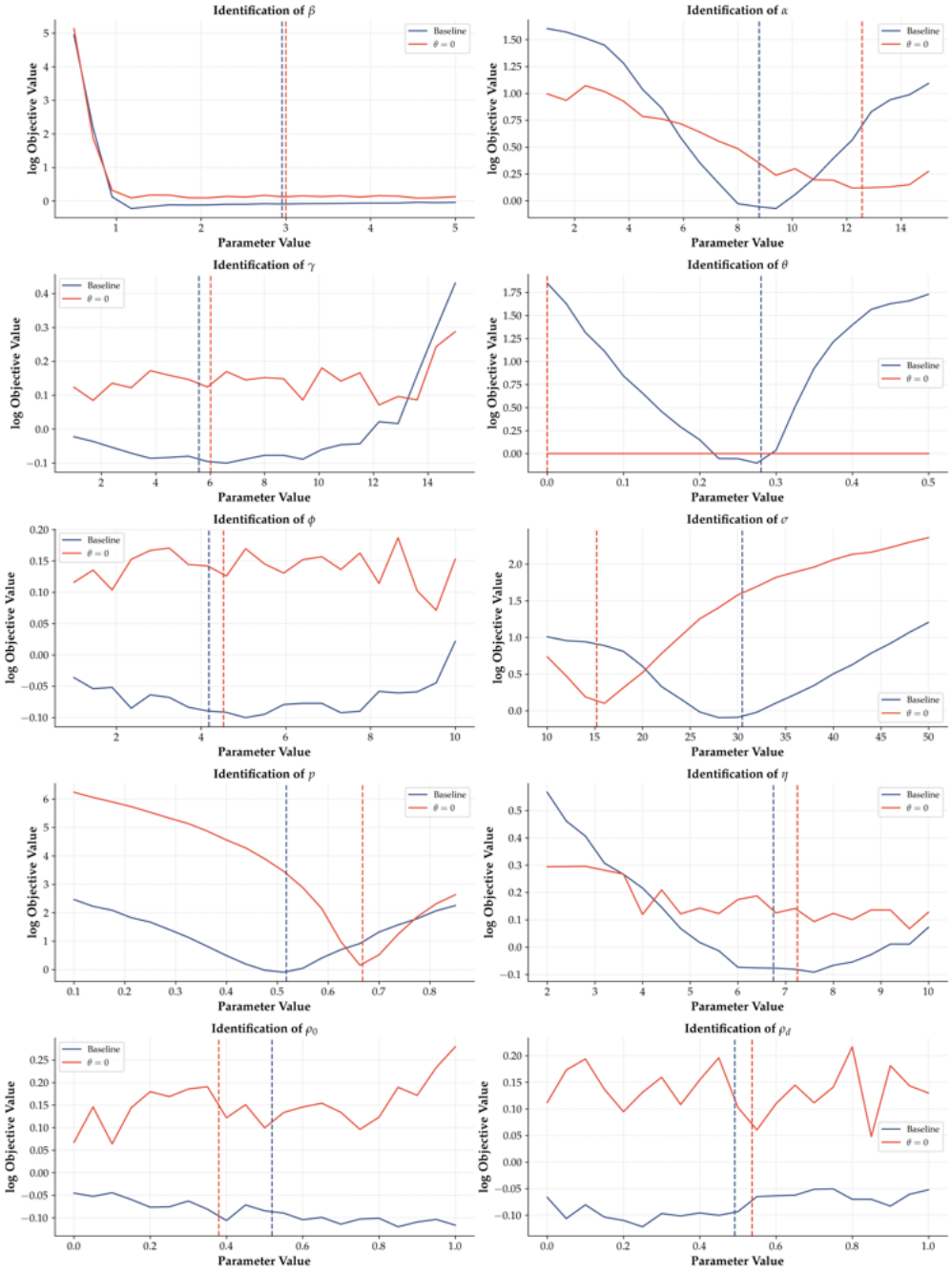


Figure 12: Parameter Identification

Note. The dashed vertical lines indicate the parameter estimates under each model, the solid lines are the log objective function values when varying the variable indicated in the title.

C.7 Quasi-Bayesian Estimation

The quasi-bayes posterior is given by

$$p\left(\boldsymbol{\vartheta}|\boldsymbol{x}^D,\pi^D\right)=\frac{\exp\left(-\lambda\mathcal{D}\left(m\left\{\pi_i\left(\boldsymbol{\vartheta},\boldsymbol{x}^D\right)\right\},m\left(\left\{\pi_i^D\right\}\right)\right)\right)p\left(\boldsymbol{\vartheta}\right)}{\int\exp\left(-\lambda\mathcal{D}\left(m\left(\left\{\pi_i\left(\tilde{\boldsymbol{\vartheta}},\boldsymbol{x}^D\right)\right\}\right),m\left(\left\{\pi_i^D\right\}\right)\right)\right)p\left(\tilde{\boldsymbol{\vartheta}}\right)},$$

where $m\left(\cdot\right)$ is the vector of moments described in Appendix Section C.4, $\boldsymbol{x}^D$ includes country revenue potentials and distances, $\boldsymbol{\vartheta}$ is the vector of estimated parameters, and λ is a scaling parameter. The function $\mathcal{D}\left(\cdot\right)$ is the standard SMM distance formula with a weighting matrix as described in Appendix Section C.4. The scaling parameter helps control the acceptance rate of the MCMC algorithm (Metropolis Hastings). When λ is larger, parameters that result in worse objective values are accepted with lower probability.

Each parameter $\vartheta_k \in \boldsymbol{\vartheta}$ is assumed to belong to a bounded space $\vartheta_k \in \left[b_k^L, b_k^U\right]$. The prior $p\left(\boldsymbol{\vartheta}\right)$ is a multi-variate normal distribution centered around $\left\{\left(b_k^L+b_k^U\right)/2\right\}_k$ with diagonal covariance elements of $\left\{\left(\left(b_k^L-b_k^U\right)/4\right)^2\right\}_k$ and zero off-diagonal element. The initial proposal distribution is a multi-variate normal distribution with zero mean and diagonal covariance terms $\left\{\left(b_k^L-b_k^U\right)\times 0.025\right\}_k$. After 2000 iterations, the proposal distribution is updated to be proportional to the covariance of the generated Markov chain. It continues to be updated every 200 iterations. Any proposals outside of the allowed bounds for each parameter are discarded.

Figure 13 is a trace plot for the posterior posterior distribution. Figure 14 is the cumulative acceptance rate for the proposal distribution. Figures 15 and 16 include trace plots and the posterior distribution for the baseline model and the $\theta = 0$ model. Each of the below figures has a burnin of 2000 iterations.

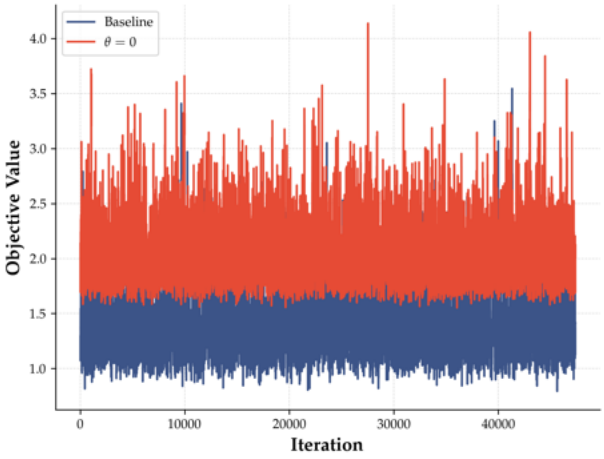


Figure 13: Posterior Trace

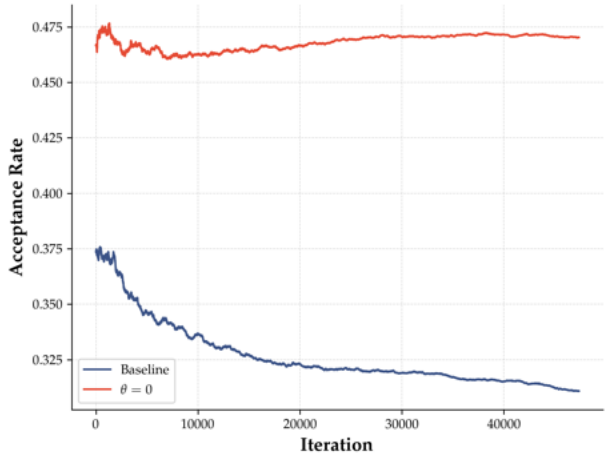


Figure 14: Acceptance Rate

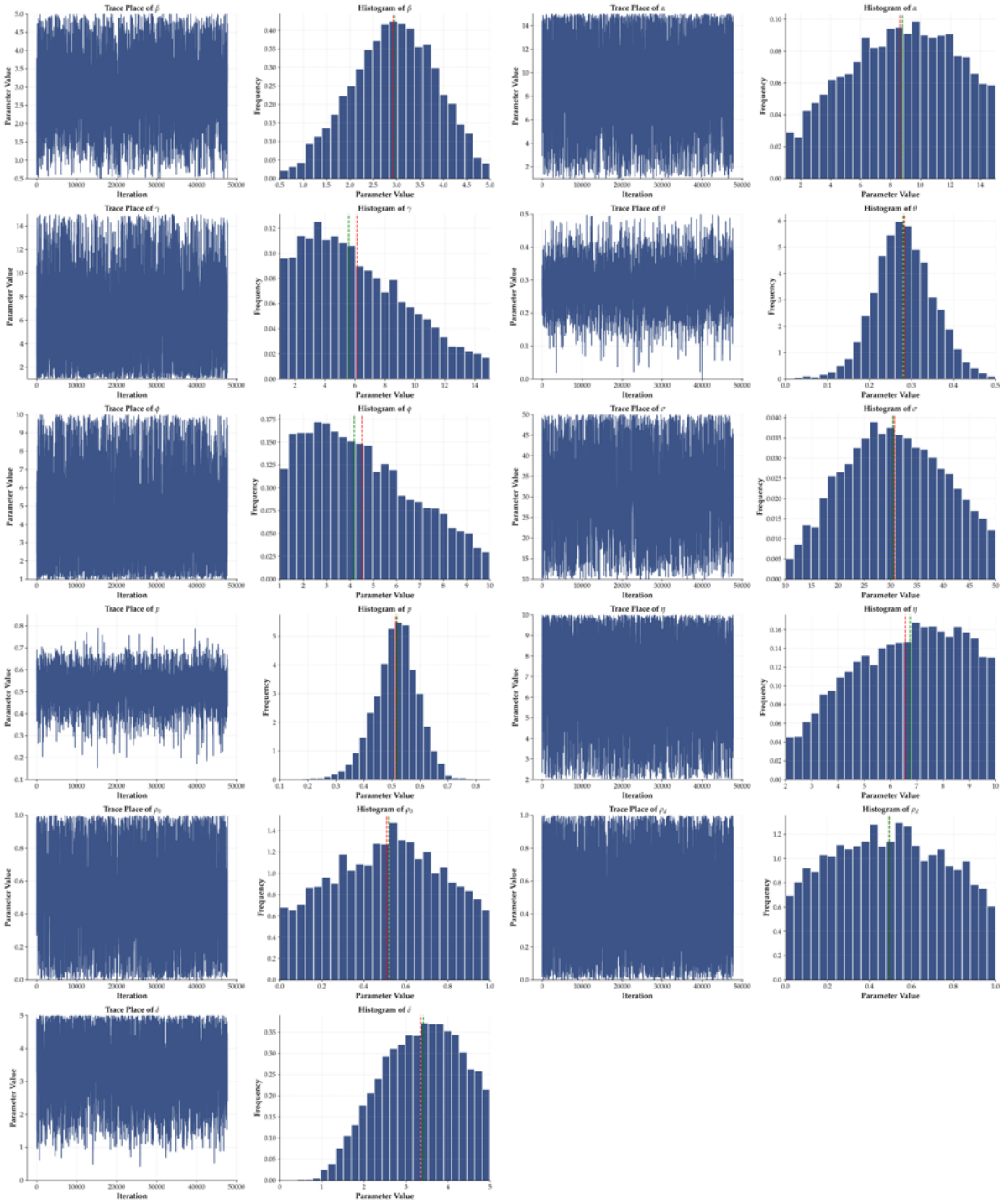


Figure 15: Baseline Trace and Posterior Plots

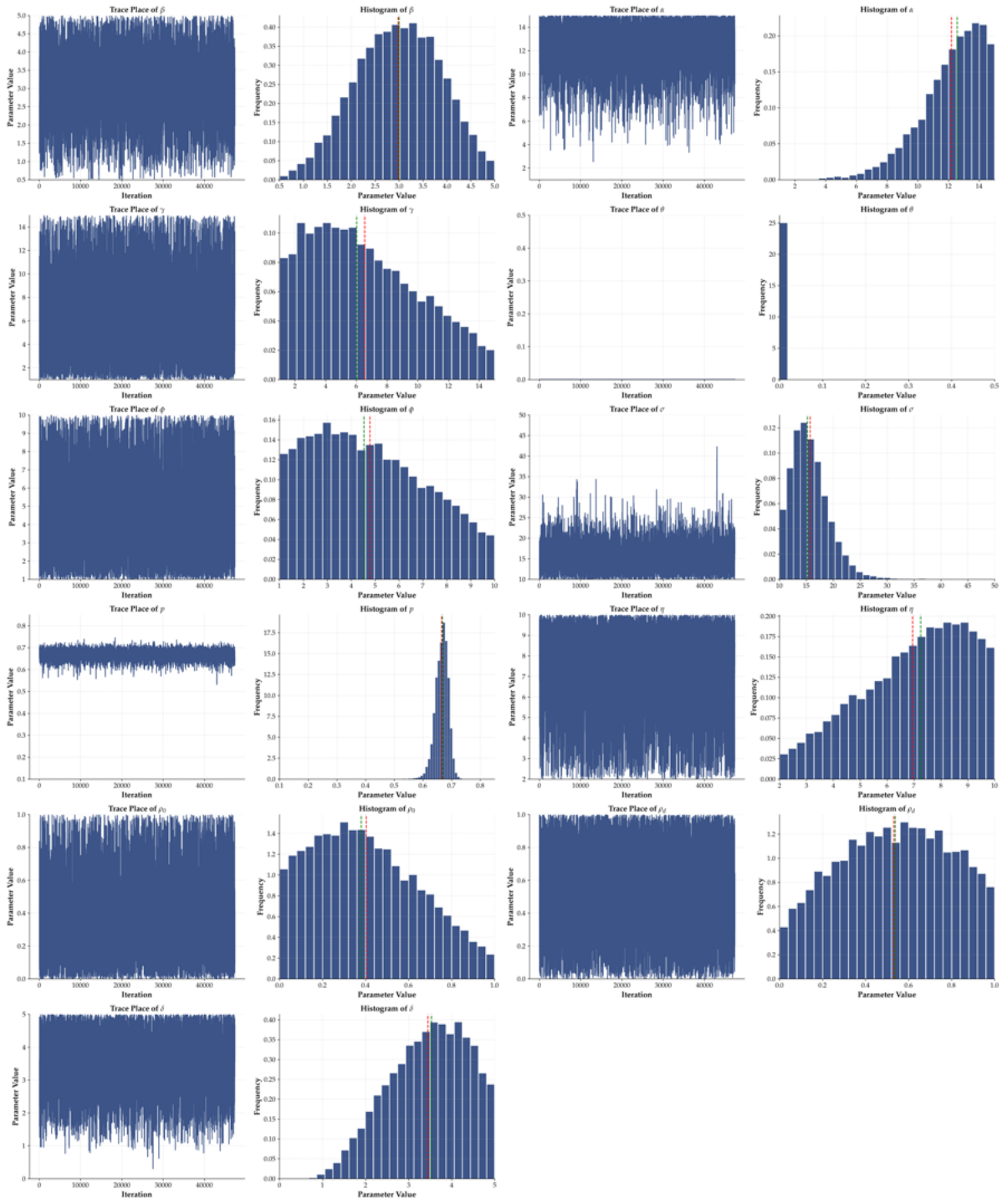


Figure 16: $\theta = 0$ Trace and Posterior Plots

D Additional Numerical Results

D.1 ISPL Oligopoly Computation

Claim. If $\sigma - 1 > \theta$ and $\varepsilon - 1 < \theta$, then (7) is neither supermodular nor submodular.

Proof. Suppose $c_k \to \infty$, then $P = \bar{p}$. The price elasticity of revenue is then $1 - \sigma$. If instead, $c_k \to 0$, then $P = p_k$ and the price elasticity of revenue is $1 - \varepsilon$. The sign of the derivative of the profit function with respect to $\sum_{i=1}^{N} \mathbb{1}_i T_i$ depends on $(\sigma - 1)/\theta$ in the former case and $(\varepsilon - 1)/\theta$ in the latter case. The statement of the proposition follows.

Numerically, Figure (17) displays how (7) can be neither supermodular nor submodular.

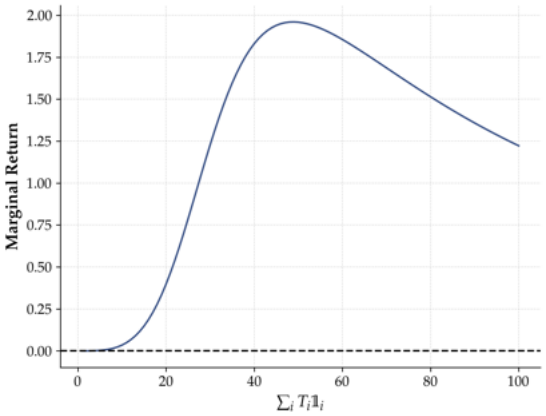


Figure 17: Marginal Return from $\sum_i \mathbb{1}_i T_i$

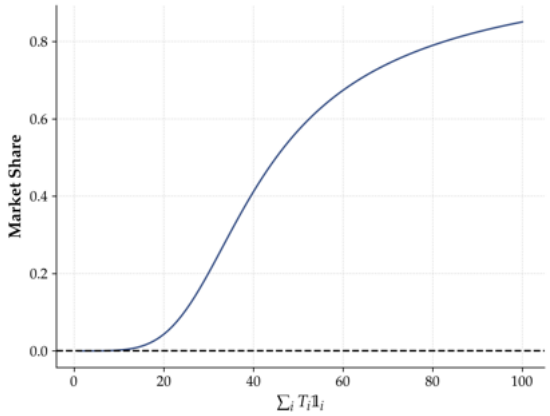


Figure 18: Market Share

Note. Simulations parameters: $\sigma = 4$, $\theta = 1.5$, $\varepsilon = 1.5$, and $\bar{p} = 0.2$.

D.2 Theory Loss

In this section I train an ISPL model with an additional term in the objective function motivated by properties of the true policy function. Consider a value $\alpha > 1$, it must be the case that any choice made at α would also be made when I increase the parameter to some $\alpha + \varepsilon$ with $\varepsilon > 0$. Therefore, I can place an additional constraint on the objective function: it must maximize equation 4 and any choice made for a value of α greater than 1, must continue to be made for a larger value of α. Formally, I add the following term as an additional loss

$$\mathbb{1}\left\{\left(\pi_i\left(\left\{\left\{T_i, f_i\right\}, \alpha + \varepsilon\right\}\right) - \pi_i\left(\left\{\left\{T_i, f_i\right\}, \alpha\right\}\right)\right) < 0\right\} \mathbb{1}\left\{\alpha > 1\right\}.$$

I include this Loss term for the first several batches of training before removing it. The idea here is that we are looking for a policy function that maximizes equation 4 among the entire set of policy functions that the model can approximate. That policy function is also part of the subset of policy functions that feature this property regarding increasing α, which is a smaller set than the full set of admissible policy functions. If I first move the training to this subset of policy functions, then I can more quickly converge to the objective maximizing policy function.

In Figure 19 I plot examples of learning curves with the same initialization of parameters and training data, but some are trained with this additional loss. The additional loss results in faster convergence. However, this is not always the case. The average difference in log Optimality Gap

across 250 trained models between models trained with and without the Theory Loss is plotted in Figure 20.

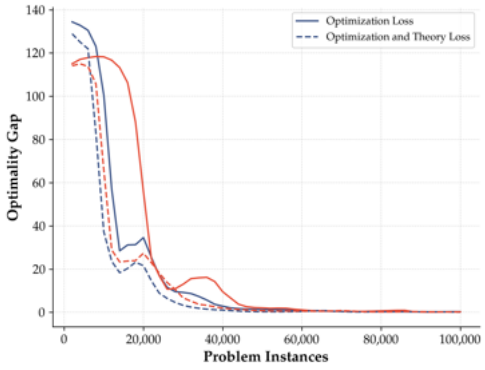


Figure 19: Sample Learning Curves

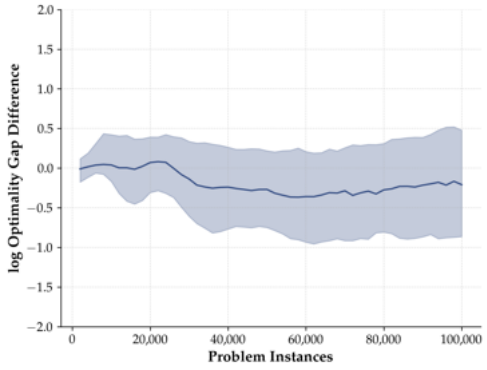


Figure 20: Median Learning Curve

D.3 Optimality Evaluation

The optimality gap is defined by Equation (18). A positive value indicates the policy given by the legend performed worse than the reference policy, listed on the y-axis. The opposite is true for a negative value. And, a value of zero indicates both methods return the same policy. The optimality gap is evaluated in a distribution of problem instances and the values are sorted from best-case to worst-case.

D.3.1 Input Sourcing & Plant Location

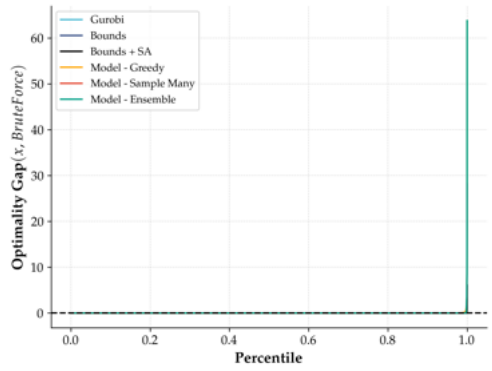


Figure 21: Supermodular ($N = 20$)

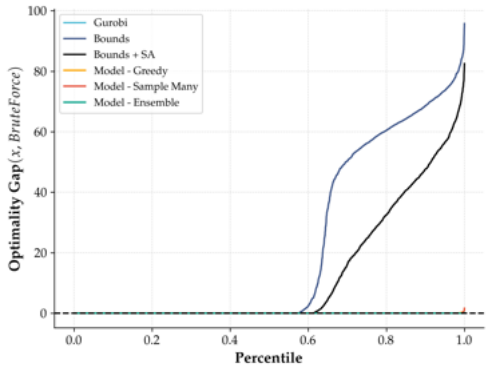


Figure 22: Submodular ($N = 20$)

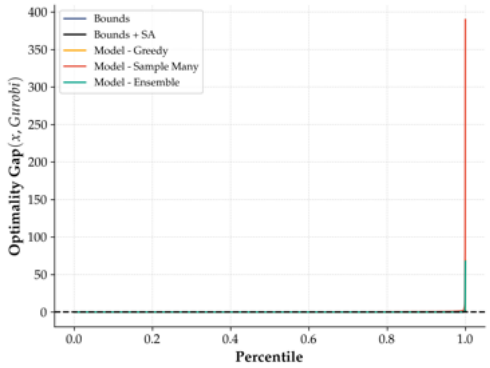


Figure 23: Supermodular ($N = 50$)

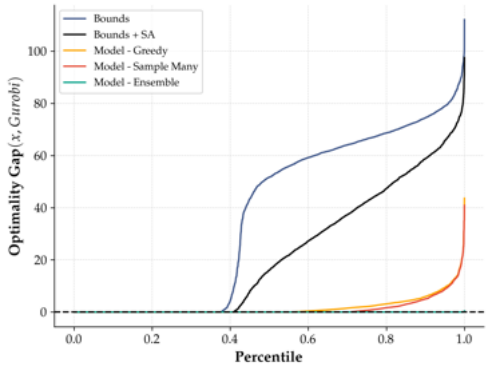


Figure 24: Submodular ($N = 50$)

D.3.2 Input Sourcing & Plant Location - Oligopoly

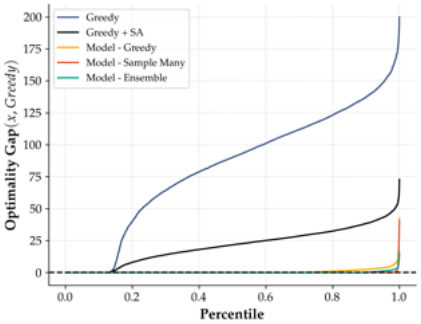


Figure 25: $N = 20$

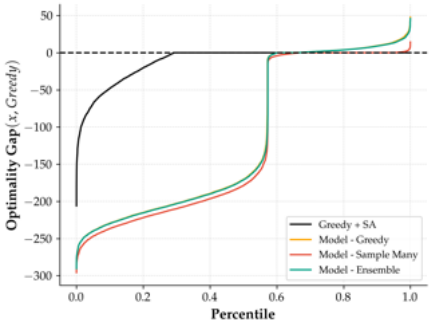


Figure 26: $N = 50$

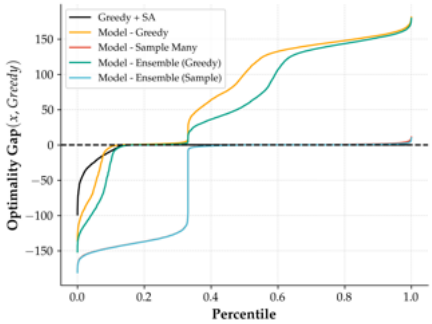


Figure 27: $N = 100$

D.3.3 Global Value Chains

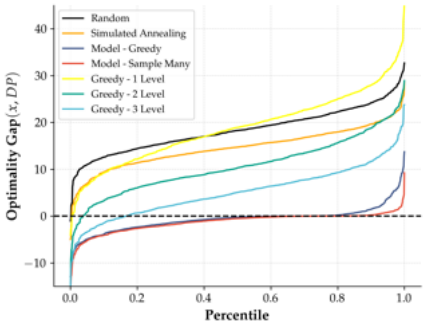


Figure 28: $N = 20$

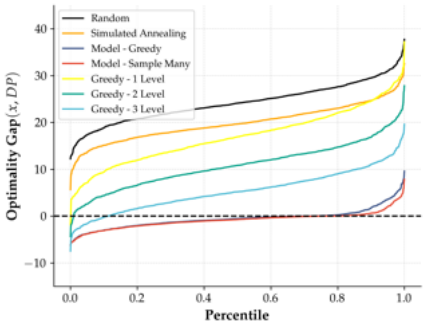


Figure 29: $N = 50$

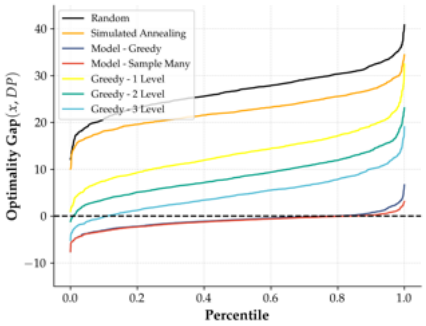


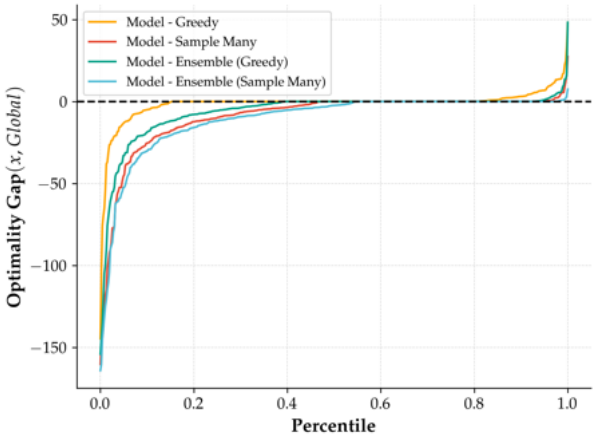
Figure 30: $N = 100$

D.3.4 Export Market Interdependence

Table 7: Algorithm Performance

Method	$N = 50$		
	Cost	Gap	Time
Greedy	-2.43	0.00%	(34s)
Model - Greedy	-2.55	-1.35%	(8s)
Model - Sample Many	-2.74	-9.33%	(8m)
Model - Ensemble (Greedy)	-2.68	-6.40%	(1m)
Model - Ensemble (Sample)	-2.80	-11.19%	(44m)

Figure 31: Algorithm Performance



D.4 Model Training

D.4.1 Input Sourcing & Plant Location

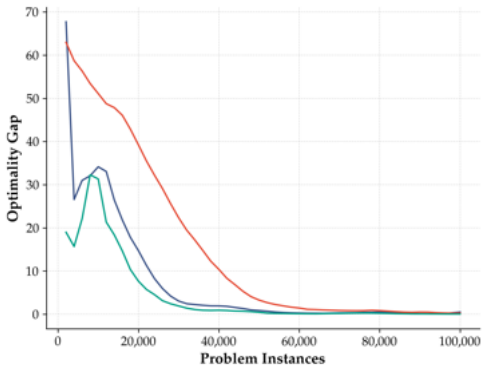


Figure 32: $N = 20$

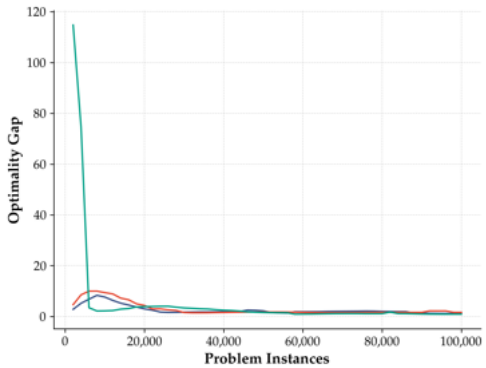


Figure 33: $N = 50$

D.4.2 Input Sourcing & Plant Location - Oligopoly

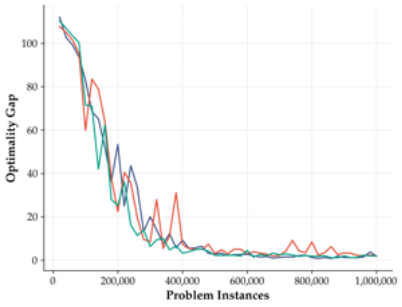


Figure 34: $N = 20$

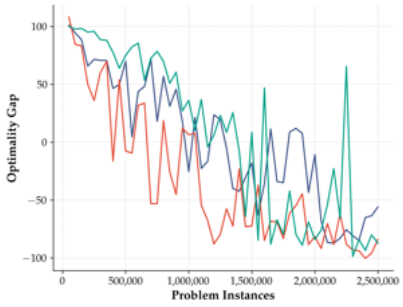


Figure 35: $N = 50$

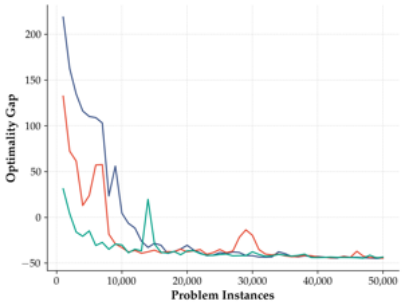


Figure 36: $N = 100$

D.4.3 Global Value Chains

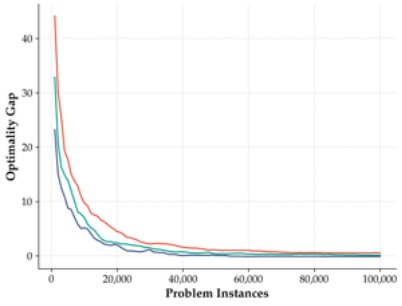


Figure 37: $N = 20$

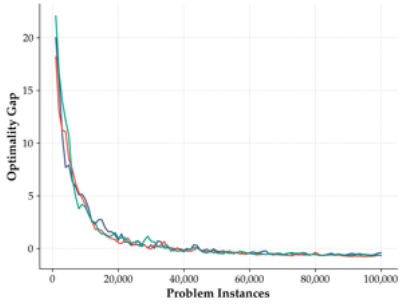


Figure 38: $N = 50$

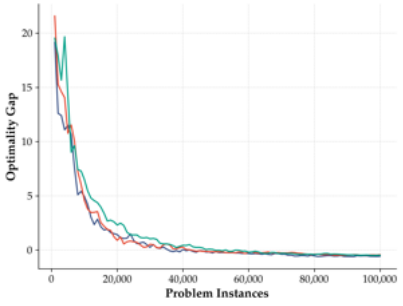


Figure 39: $N = 100$

D.4.4 Export Market Interdependence

Table 8 displays the range of parameter values I train over. Table 9 displays the range of parameter values I use to generate country revenues. Note, the model does not directly take as input parameters that generate the revenues potentials: $\{\sigma, p, \rho_0, \rho_d\}$. The model takes as input the distribution of revenue potentials, so as long as these are generated in a way that is consistent with the method used for estimation there should not be an issue with optimality.

Figures 40 and 41 plot learning curves for the 20-location and 50-location export market entry model. There is considerable variability in training outcomes, a work in progress is tuning the hyper-parameters of model training to stabilize training.

Table 8: Training Parameter Space

Parameter	β	α	γ	θ	ϕ	η
Bounds	(0.0,5.0)	(0.0,20.0)	(0.0,20.0)	(0.0,0.5)	(0.0,10.0)	(0.0,50.0)

Table 9: Revenue Sampling Parameter Space

Parameter	σ	p	ρ_0	ρ_d
Bounds	(0.0,50.0)	(0.0,1.0)	(0.0,1.0)	(0.0,1.0)

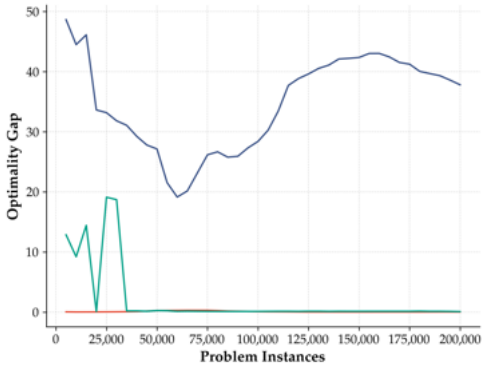


Figure 40: $N = 20$

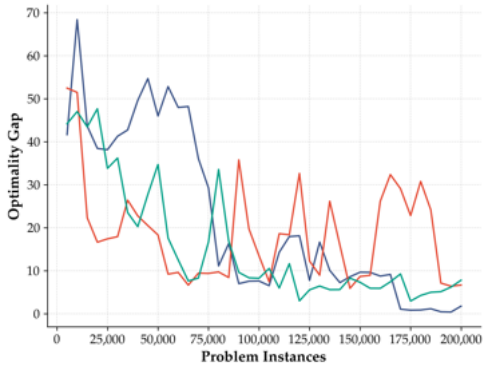


Figure 41: $N = 50$

E Firm-Level Import Demand System

E.1 Stylized Model

Each country is described by a vector of characteristics $\left\{x_j^c\right\}$ where $j \in N$ indexes the country and $c \in C$ the characteristic. For example, countries could be described by a vector of factor endowments or the number of firms in various industries.[53] A firm has a nested CES production structure over the characteristics and countries:

$$Y = \left(\sum_c^C \left(\sum_j^N \left(y_j^c x_j^c \right)^{\frac{\sigma_c - 1}{\sigma_c}} \right)^{\frac{\sigma_c}{\sigma_c - 1} \frac{\sigma - 1}{\sigma}} \right)^{\frac{\sigma}{\sigma - 1}}$$

where σ_c is the elasticity of substitution across country-specific value of characteristic c and σ is the elasticity of substitution across characteristics. Suppose firms face an inverse demand function $p = AY^{-1/\rho}$ and that sourcing from a country imposes a fixed costs f_j. The firm's optimization problem is given by

$$\max_{\{y_j, \mathbb{1}_j\}} A \left(\sum_c^C \left(\sum_j^N \mathbb{1}_j \left(y_j^c x_j^c \right)^{\frac{\sigma_c - 1}{\sigma_c}} \right)^{\frac{\sigma_c}{\sigma_c - 1} \frac{\sigma - 1}{\sigma}} \right)^{\frac{\sigma}{\sigma - 1} \frac{\rho - 1}{\rho}} - \sum_{j,c} p_j^c y_j^c - \sum_j f_j \mathbb{1}_j.$$

[53] Appendix Section E.4 provides one possible micro foundation based on the number of firms in various sectors.

Claim. The degree of complementarity between countries depends on their vector of characteristics. Two countries can be complements if they share the same characteristics and those characteristics have low substitutability, or if they specialize in different characteristics.

Proof. See Appendix Section E.5.

For example, let us consider a steel plant located in the US. This firm requires two types of inputs: raw materials in the form of iron ore and a suite of capital equipment needed to melt, move, shape, and process that iron ore into steel production. Suppose that in the data I observed this firm importing goods from two ISIC industries: C (Mining and Quarrying) and 29 (Machinery). The iron ore input falls under ISIC C and is a highly substitutable input. However, the capital equipment is imported under ISIC 29 and highly specialized to specific tasks of the firm and hence not very substitutable. If there are fixed costs from importing, the firm might be unwilling to source iron ore from say Brazil and Australia since they are effectively offering the same product. But if Germany makes the best foundries and Italy makes the best steel presses, then as non-substitutable ISIC 29 inputs the firm may be willing to import these products from their respective countries. Regardless of the sourcing decision of capital equipment, the plant will need to source both iron ore and capital equipment, these are highly non-substitutable - making sourcing from Germany/Italy complementary with sourcing from Brazil/Australia.

E.2 Identification Challenges

In order to take the model closer to data, I introduce some heterogeneity to the preferences. There are a few ways to do this, but right now I consider heterogeneous tastes for each characteristic-country:

$$Y_i = \left(\sum_c^C \left(\nu_{i,j}^c \sum_j^N \left(y_{i,j}^c x_{i,j}^c \right)^{\frac{\sigma_c - 1}{\sigma_c}} \right)^{\frac{\sigma_c}{\sigma_c - 1} \frac{\sigma - 1}{\sigma}} \right)^{\frac{\sigma}{\sigma - 1}}$$

where i indexes the firm and $\nu_{i,j}^c$ is a idiosyncratic preference for characteristic c in country j. To highlight the identification challenge that arises when using only intensive margin export data, let us suppose that a firm has $\nu_{i,j}^c = 0 \forall c \neq 1$. For example, this might be a clothing distributor which only cares about sourcing garments from countries with cheap low-skill labor, so the only characteristic it is sensitive to is the endowment/price of this factor in a country.

The import shares this firm exhibits are then give by

$$\chi_{ij} = \frac{\nu_{i,j}^1 \mathbb{1}_{ij} \left(x_j^1 \right)^{\frac{\sigma_1 - 1}{\sigma_1} \sigma} p_j^{-\sigma}}{\left(\sum_k \nu_{i,k}^1 \mathbb{1}_{i,k} \left(x_k^1 \right)^{\frac{\sigma_1 - 1}{\sigma_1} \sigma} p_k^{1-\sigma} \right)}.$$

where $\mathbb{1}_{ij}$ are the sourcing decisions the firm. An issue arises since the firm is only going to select into importing from locations with high x_j^c. That is, $\mathbb{E}\left[x_j^c \nu_{i,j}^c | \chi_{ij} > 0 \right] \neq 0$, which will bias the estimates of σ_1. Previous work on import sourcing which tries to estimate this kind of intensive margin

regression has to argue that $\nu_{i,j}^c$ is only realized after the extensive-margin decisions are made, e.g. Antràs et al. (2017). This seems unlikely, especially if firms are making repeated transactions over time - and part of the fixed costs I model might be information acquisition costs - where firms try to reveal $\nu_{i,j}^c$.

Zeros in Trade Data. This approach to demand estimation, where I explicitly model the extensive and intensive margin decisions, connects to a broader issue of zeros in bilateral trade data. Dingel and Tintelnot (2020) provides a useful summary of the general issue in spatial settings. The standard rationale for zeros is that with a finite number of firms and stochastic idiosyncratic preferences, you can end up in situations where no firm draws the necessary shock to source from a location. The sourcing probabilities are still based on economic fundamentals, but it is sampling which principally generates the zeros rather than an endogenous extensive margin choice. The model I use will still feature idiosyncratic preferences that give rise to a stochastic sourcing process, but the additional structure I place on preferences lets more of the sourcing decisions load on characteristics and cross-country dependencies between choices.

E.3 Empirical Model

The CES structure used in this stylized example maps closest to the models of input sourcing in the trade literature. But, a more flexible model would serve us better, especially if I want to capture richer forms of cross-country substitutability. The basic structure of the model follows Berry et al. (1995). Let i index the decision-making agent, j the country, and c a characteristic of the choices. The utility a firm gains from each choice is then given by

$$u_{ij} = \sum_c \beta_i^c x_j^c + \sum_c \sum_{c'} \sum_k \beta_{ik}^{cc'} h_k \left(x_j^c, \left\{ x_{j'}^{c'} \right\}_{j' \neq j} \right) + \xi_j + \varepsilon_{ij}$$

where

$$\begin{aligned} \beta_i^c &= \overline{\beta}^c + \sum_r z_{ir} \alpha_r^c + \nu_i^c \sigma^c \\ \beta_{ik}^{cc'} &= \overline{\beta}_k^{cc'} + \sum_r z_{ir} \alpha_{rk}^{cc'} + \nu_i^{cc'} \sigma_k^{cc'}. \end{aligned} \tag{E.1}$$

This is a random-coefficients demand system where I allow an agent's preference for a characteristic to vary depending on a vector of observed characteristics $\{z_{ir}\}_r$ and a dimension of unobserved heterogeneity ν_i^c. What is non-standard is the inclusion of interactions between choices: $h_k(x_j^c, \{x_{j'}^{c'}\}_{j' \neq j})$. Here I consider a set of functions each indexed by k, h_k, which are various interactions between the choice x_j and the vector of other choices. This is how I generate all the potential complementarity between choices. For example, let $h_k(\cdot) = x_j^c \sum_{j' \neq j} x_{j'}^{c'}$ with $\beta_{ik}^{cc'} > 0$ and $x_j^c > 0 \forall j$. This function captures a pattern whereby when more choices are added to the sourcing set with high values of characteristic c, the utility gained from a choice which itself has high values

of c increases.

For the sake of tractability, I assume that the agent imports a continuum of goods and ε_{ij} is drawn for each individual unit. Let $\nu_i \equiv \{\{\nu_i^c\}_c, \{\nu_{ik}^{cc'}\}_{c,c',k}\}$ collect all the relevant dimensions of unobserved heterogeneity for agent i. The spending share of agent i on goods from country j is then given by

$$\chi_{ij}(\nu_i) = \frac{\mathbb{1}_{ij} \exp\left(\left[\sum_c \beta_i^c x_j^c + \sum_c \sum_{c'} \sum_k \beta_{ik}^{cc'} h_k\left(x_j^c, \{x_{j'}^{c'}\}_{j' \neq j}\right)\right] + \xi_j\right)}{\sum_m \mathbb{1}_{im} \exp\left(\left[\sum_c \beta_m^c x_m^c + \sum_c \sum_{c'} \sum_k \beta_{im}^{cc'} h_k\left(x_m^c, \{x_{m'}^{c'}\}_{m' \neq m}\right)\right] + \xi_m\right)}.$$

The unweighted market shares of each country are then given by

$$s_j = \int_i \chi_{ij}(\nu_i)\, dF(\nu_i).$$

I finally model the firm's extensive margin decisions as

$$\max_{\mathbb{1}_j} \log \sum_j \mathbb{1}_j \exp\left(\left[\sum_c \beta_i^c x_j^c + \sum_c \sum_{c'} \sum_k \beta_{ik}^{cc'} f_k\left(x_j^c, \left\{x_{j'}^{c'}\right\}_{j' \neq j}\right)\right] + \xi_j\right) - \sum_j \mathbb{1}_j f_{ij}$$

where

$$f_{ij} = \sum_c \gamma^c d_j^c + \sum_c \sum_{c'} \sum_k \gamma_k^{cc'} g_k\left(d_j^c, \left\{d_{j'}^{c'}\right\}_{j' \neq j}\right) + \sigma^e \delta_{ij}.$$

Here $\left\{d_j^c\right\}_c$ is a vector of characteristics that are relevant for the extensive margin decisions of a firm and g_k are interactions of those characteristics. The Alfaro-Urena et al. (2023) model of fixed-cost interdependence would fit under this structure if you let d_j^c be the relevant distance, language, and trade agreement measures and $g(\cdot)$ be the interdependence functional forms they impose. I do not impose a random-coefficients structure on the determinants of fixed-costs, but I do allow one dimension of heterogeneity in δ_{ij}.

Estimation. Estimating this model differs from standard Berry et al. (1995) estimation, in that even the intensive-margin parameters become non-linear. This is because the extensive margin can not be solved without taking a stance on the intensive margin parameters.[54] The estimation procedure will need to be simulated method of moments where I attempt to match moments of the intensive and extensive margin. But, given the large number of non-linear parameters this introduces, I will likely rely on the quasi-bayesian procedure introduced in Section 7.1. The core contribution of this approach is that I try to seriously handle the issue of observed and unobserved heterogeneity in the preference for country exports. This will yield more realistic substitution patterns, which can critically evaluate the proportional substitution assumptions of the commonly

[54] In Appendix Section E.6 I describe the estimation challenge in additional detail.

used CES preferences and yield more realistic trade counterfactuals.

E.4 Characteristic Demand System Microfoundation

Let us consider a single country j to start. This country is populated by a continuum of upstream firms in each sector, $N_{j,c}$ - where j indexes the country c the sector. Each firm sells a differentiated input sold at price w_j. These upstream inputs are aggregated by a perfectly competitive intermediary sector with access to the following input aggregation technology:

$$y_{j,c} = \left(\int_0^{N_c} x\left(i\right)^{\beta} di \right)^{\frac{1}{\beta}}.$$

These firms sell at marginal cost:

$$p_{j,c} = w_j N_{j,c}^{-\frac{1}{\beta}}.$$

These sector-specific intermediates and then bundled into single exportable commodity y_j with price $p_j = \sum_c p_{j,c}$. I can alternatively view one unit of this commodity as providing $N_{j,c}^{\frac{1}{\beta}}$ units of each sector's input as a price of Cw_j. The final good producing firm then combines differentiated global varieties, subject to import fixed costs, to produce consumable goods:

$$\max_{\{y_j, \mathbb{1}_j\}} A \left(\sum_c^C \left(\sum_j^N \mathbb{1}_j \left(y_j N_{j,c}^{\frac{1}{\beta}} \right)^{\frac{\sigma_c - 1}{\sigma_c}} \right)^{\frac{\sigma_c}{\sigma_c - 1} \frac{\sigma - 1}{\sigma}} \right)^{\frac{\sigma}{\sigma - 1} \frac{\rho - 1}{\rho}} - \sum_j y_j C w_j - \sum_j f_j \mathbb{1}_j.$$

E.5 Complementarity Claim

Restating Profit Function. I start by restating the firms objective function:

$$\max_{\{y_j, \mathbb{1}_j\}} A \left(\sum_c^C \left(\sum_j^N \mathbb{1}_j \left(y_j x_j^c \right)^{\frac{\sigma_c - 1}{\sigma_c}} \right)^{\frac{\sigma_c}{\sigma_c - 1} \frac{\sigma - 1}{\sigma}} \right)^{\frac{\sigma}{\sigma - 1} \frac{\rho - 1}{\rho}} - \sum_j p_j y_j - \sum_j f_j \mathbb{1}_j.$$

Lets drop the indicators for now, but they will reappear in the end whenever I sum over counties. Taking the first-order condition with respect to y_j yields:

$$\begin{aligned} p_j = & \left(\frac{\sigma}{\sigma - 1} \frac{\rho - 1}{\rho} \right) A \left(\sum_c^C \left(\sum_j^N \left(y_j x_j^c \right)^{\frac{\sigma_c - 1}{\sigma_c}} \right)^{\frac{\sigma_c}{\sigma_c - 1} \frac{\sigma - 1}{\sigma}} \right)^{\frac{\sigma}{\sigma - 1} \frac{\rho - 1}{\rho} - 1} \\ & \times \sum_c^C \left[\left(\frac{\sigma_c}{\sigma_c - 1} \frac{\sigma - 1}{\sigma} \sum_j^N \left(y_j x_j^c \right)^{\frac{\sigma_c - 1}{\sigma_c}} \right)^{\frac{\sigma_c}{\sigma_c - 1} \frac{\sigma - 1}{\sigma} - 1} \frac{\sigma_c - 1}{\sigma_c} \left(x_j^c \right)^{\frac{\sigma_c - 1}{\sigma_c}} y_j^{\frac{\sigma_c - 1}{\sigma_c} - 1} \right] \end{aligned}$$

Taking the ratio of prices for two countries I get :

$$\frac{p_j}{p_k} = \frac{y_j^{\frac{\sigma_c-1}{\sigma_c}-1}\alpha_j}{y_k^{\frac{\sigma_c-1}{\sigma_c}-1}\alpha_k} \Leftrightarrow \frac{p_j^{-\sigma}}{p_k^{-\sigma}} = \frac{y_j\alpha_j^{-\sigma}}{y_k\alpha_k^{-\sigma}} \tag{E.2}$$

where the term α_j is given by

$$\alpha_j = \sum_c^C \left[\left(\sum_j^N \left(y_j x_j^c\right)^{\frac{\sigma_c-1}{\sigma_c}} \right)^{\frac{\sigma_c}{\sigma_c-1}\frac{\sigma-1}{\sigma}-1} \left(x_j^c\right)^{\frac{\sigma_c-1}{\sigma_c}} \right].$$

Substituting equation E.2, I can get an implicit equation for α_j

$$\alpha_j = \sum_c^C \left[\left(\sum_k^N \left(\alpha_{k'}^{\sigma} p_{k'}^{-\sigma} x_k^c\right)^{\frac{\sigma_c-1}{\sigma_c}} \right)^{\frac{\sigma_c}{\sigma_c-1}\frac{\sigma-1}{\sigma}-1} \left(x_j^c\right)^{\frac{\sigma_c-1}{\sigma_c}} \right]. \tag{E.3}$$

I can then write output as

$$Y = \left(\sum_c^C \left(\sum_j^N \left(y_j x_j^c\right)^{\frac{\sigma_c-1}{\sigma_c}} \right)^{\frac{\sigma_c}{\sigma_c-1}\frac{\sigma-1}{\sigma}} \right)^{\frac{\sigma}{\sigma-1}}$$

and substituting equation E.2 I get

$$\begin{aligned} Y &= \left(\sum_c^C \left(\sum_j^N \left(\left(\alpha_j^{\sigma} p_j^{-\sigma} y_k \alpha_k^{-\sigma} p_k^{\sigma}\right) x_j^c \right)^{\frac{\sigma_c-1}{\sigma_c}} \right)^{\frac{\sigma_c}{\sigma_c-1}\frac{\sigma-1}{\sigma}} \right)^{\frac{\sigma}{\sigma-1}} \\ &= y_k \alpha_k^{-\sigma} p_k^{\sigma} \left(\sum_c^C \left(\sum_j^N \left(\left(\alpha_j^{\sigma} p_j^{-\sigma}\right) x_j^c \right)^{\frac{\sigma_c-1}{\sigma_c}} \right)^{\frac{\sigma_c}{\sigma_c-1}\frac{\sigma-1}{\sigma}} \right)^{\frac{\sigma}{\sigma-1}} \end{aligned}$$

Rearranging terms and summing both sides yields

$$Y \sum_k^N \left(\alpha_k^{\sigma} p_k^{-\sigma+1}\right) \left(\sum_c^C \left(\sum_j^N \left(\alpha_j^{\sigma} p_j^{-\sigma} x_j^c\right)^{\frac{\sigma_c-1}{\sigma_c}} \right)^{\frac{\sigma_c}{\sigma_c-1}\frac{\sigma-1}{\sigma}} \right)^{-\frac{\sigma}{\sigma-1}} = \sum y_k p_k,$$

which using $\sum y_k p_k = PY$ returns

$$P = \left(\sum_c^C \left(\sum_j^N \left(\frac{\alpha_j^{\sigma} p_j^{-\sigma}}{\sum_k^N \alpha_k^{\sigma} p_k^{-\sigma+1}} x_j^c \right)^{\frac{\sigma_c-1}{\sigma_c}} \right)^{\frac{\sigma_c}{\sigma_c-1}\frac{\sigma-1}{\sigma}} \right)^{-\frac{\sigma}{\sigma-1}}.$$

To get profits I then write

$$\max_{Y} AY^{\frac{\rho-1}{\rho}} - PY,$$

which yields

$$A\frac{\rho-1}{\rho}Y^{-\frac{1}{\rho}} = P \Leftrightarrow Y = \left(\frac{1}{A}\frac{\rho}{\rho-1}P\right)^{-\rho}.$$

Substituting back into the objective gives

$$\pi = \tilde{A}P^{1-\rho}$$

where $\tilde{A} = A^{\rho}\left(\frac{\rho}{\rho-1}\right)^{-\rho}\left(\frac{1}{\rho-1}\right)$. I can then rewrite the original objective as

$$\max_{\{\mathbb{1}_j\}} \tilde{A}\left(\left(\sum_{c}^{C}\left(\sum_{j}^{N}\left(\frac{\mathbb{1}_j\alpha_j^{\sigma}p_j^{-\sigma}}{\sum_k^N \mathbb{1}_j\alpha_k^{\sigma}p_k^{-\sigma+1}}x_j^c\right)^{\frac{\sigma_c-1}{\sigma_c}}\right)^{\frac{\sigma_c}{\sigma_c-1}\frac{\sigma-1}{\sigma}}\right)^{\frac{\sigma}{\sigma-1}}\right)^{\rho-1} - \sum_j f_j\mathbb{1}_j.$$

Minimal Example - Similar Characteristics. Suppose there are two countries $j \in \{A, B\}$ and two characteristics $c \in \{1, 2\}$. Suppose that prices are equal across counties and there are no trade costs. Normalize that price to 1: $p_A = p_B = 1$. The characteristics are distributed as follows:

$$\begin{aligned} x_A^1 &= \bar{x} \\ x_A^2 &= 0 \\ x_B^1 &= \bar{x} \\ x_B^2 &= 0 \end{aligned}$$

Suppose that the firm already imports from A. I want to know if adding B to the sourcing set will raise the payoff from sourcing from A. Note that with the normalization, I now have

$$\frac{\mathbb{1}_j\alpha_j^{\sigma}p_j^{-\sigma}}{\sum_k^N \mathbb{1}_j\alpha_k^{\sigma}p_k^{-\sigma+1}} = \frac{\mathbb{1}_j\alpha_j^{\sigma}}{\sum_k^N \mathbb{1}_j\alpha_k^{\sigma}} \in [0, 1].$$

Additionally, with the imposed symmetry, I know that when the firm sources from one location $\frac{\mathbb{1}_j\alpha_j^{\sigma}}{\sum_k^N \mathbb{1}_j\alpha_k^{\sigma}} = 1$, and if sources from both them $\frac{\mathbb{1}_j\alpha_j^{\sigma}}{\sum_k^N \mathbb{1}_j\alpha_k^{\sigma}} = 0.5$. So the question reduces to

$$\tilde{A}\left((0.5\bar{x})^{\frac{\sigma_1-1}{\sigma_1}} + (0.5\bar{x})^{\frac{\sigma_1-1}{\sigma_1}}\right)^{\frac{\sigma_1}{\sigma_1-1}(\rho-1)} - \tilde{A}\left(\bar{x}\right)^{(\rho-1)} \geq \tilde{A}\left(\bar{x}\right)^{(\rho-1)}.$$

This is profits from sourcing from both countries $\{A, B\}$ less the profits from just sourcing from B compared to the profits from sourcing from just A. This quantity is positive whenever

$$\left(2\left(0.5\right)^{\frac{\sigma_1-1}{\sigma_1}}\right)^{\frac{\sigma_1}{\sigma_1-1}(\rho-1)} \geq 2.$$

This is true whenever is $\frac{\rho-1}{\sigma_1-1} > 1 \Rightarrow \sigma < \rho$. This is more likely to be true when the characteristic 1 is less substitutable and the price elasticity of demand is high, so any price reductions produce proportionally larger changes in quantities demanded. Conversely, if the characteristic is very substitutable, then it becomes less likely that sourcing from these counties is complementary.

Minimal Example - Different Characteristics. Consider the same setting, but I redistributed characteristics as follows:

$$\begin{aligned} x_A^1 =& \bar{x} \\ x_A^2 =& 0 \\ x_B^1 =& 0 \\ x_B^2 =& \bar{x} \end{aligned}$$

Suppose that the firm already imports from A. I want to know if adding B to the sourcing set will raise the payoff from sourcing from A. I can follow an nearly identical set of steps to get

$$\tilde{A}\left(\left(0.5\bar{x}\right)^{\frac{\sigma-1}{\sigma}} + \left(0.5\bar{x}\right)^{\frac{\sigma-1}{\sigma}}\right)^{\frac{\sigma-1}{\sigma}(\rho-1)} - \tilde{A}\left(\bar{x}\right)^{(\rho-1)} \geq \tilde{A}\left(\bar{x}\right)^{(\rho-1)},$$

which just swaps the cross-country elasticity of substitution for the cross-characteristic one. This is more likely to be true when the characteristics themselves are less substitutable and the price elasticity of demand is high.

E.6 Challenges in Estimation

I first subset parameters into a linear and non-linear set. The linear parameters are $\beta^{\mathcal{L}} = \left\{\overline{\beta}^c, \overline{\beta}_k^{cc'}, \alpha_r^c, \alpha_{rk}^{cc'}\right\}$ and the non-linear parameters are $\beta^{\mathcal{N}} \equiv \left\{\sigma^c, \sigma_k^{cc'}, \sigma^e, \gamma^c, \gamma_k^{cc'}, \right\}$. In standard BLP estimation I start with a guess for the non-linear parameters. I then solve for the $\{\delta_j\}$, such that the model-implied spending shares

$$s = \int \frac{\mathbb{1}_{ij} \exp\left(\delta_j + g\left(\nu_i\right)\right)}{\sum_m \mathbb{1}_{ij} \exp\left(\delta_j + g\left(\nu_i\right)\right)} dF\left(\nu_i\right),$$

equal those observed in the data. The intensive margin parameters are then determined by regressing δ_j on the characteristics of choice j using moment conditions based off of ξ_j being uncorrelated with various instruments. The problem is that $\mathbb{1}_{ij}$ is a function of the intensive margin parameters. So I can not get δ_j without first taking a stance on those parameters.