Chapter 10
Big Data Techniques as a Solution to Theory Problems

Richard W. Evans, Kenneth L. Judd, and Kramer Quist

Abstract This chapter proposes a general approach for solving a broad class of difficult optimization problems using big data techniques. We provide a general description of this approach as well as some examples. This approach is ideally suited for solving nonconvex optimization problems, multiobjective programming problems, models with a large degree of heterogeneity, rich policy structure, potential model uncertainty, and potential policy objective uncertainty. In our applications of this algorithm we use Hierarchical Database Format (HDF5) distributed storage and I/O as well as message passing interface (MPI) for parallel computation of a large number of small optimization problems.

10.1 Introduction

Big data refers to any repository of data that is either large enough or complex enough that distributed and parallel input and output approaches must be used (see [9, p. 3]). Liran and Levin [6] discuss the new opportunities in economics using big data, although they focus primarily on searching for important patterns in existing datasets. Varian [8] describes the tools Google uses to address big data questions and provides a mapping to the open source analogues of those proprietary tools. This paper proposes a very different use of big data techniques, using efficient sampling methods to construct a large data set which can then be used to address theoretical questions as well as econometric ones. More specifically, we sample the parameter space of a parametric model and use the large sample to address a research question.

R.W. Evans (✉)
Department of Economics, Brigham Young University, 167 FOB, Provo, UT 84602, USA
e-mail: revans@byu.edu

K.L. Judd
Hoover Institution, Stanford University, Stanford, CA 94305, USA
e-mail: kennethjudd@mac.com

K. Quist
Department of Economics, Brigham Young University, 151 FOB, Provo, UT 84602, USA
e-mail: kramer.quist@gmail.com

R. Arora (ed.), *Conquering Big Data with High Performance Computing*,
DOI 10.1007/978-3-319-33742-5_10

Furthermore, constructing the data sample is a large but fixed cost which allows one to use high performance computing to cheaply answer many questions.

Our leading example is an optimal tax application from [2], but we also present an econometric example. The approach described in this chapter is ideally suited for solving nonconvex optimization problems,[1] multi-objective programming problems, models with a large degree of heterogeneity, rich policy structure, potential model uncertainty, and potential policy objective uncertainty.

Our implementation of these methods has used the Python programming language with its integration with the Hierarchical Database Format (HDF5) and its distributed storage and parallel I/O. However, these methods are general across platforms. We will also detail a technique that is new to this area, which is using equidistributed sequences both as an approximation-by-simulation technique as well as an adaptive grid refinement technique by equidistributing a sequence on various hypercubic subspaces.[2]

In Sect. 10.2, we give a general description of the big data approach to solving theoretical problems, a computational description, and a description of our use of equidistributed sequences. In Sect. 10.3, we describe an optimal taxation example of this approach. Section 10.4 describes some other applications of this approach, with a more detailed description of an econometric example. Section 10.5 concludes.

10.2 General Formulation of Big Data Solution Method

In this section, we first formulate a general class of problems that are amenable to our big data solution method. We then outline the computational steps of the method and describe some specific techniques that we use in our implementation. We then describe in more depth one of the key tools we use—equidistributed sequences—to efficiently find the set of solutions using our method. This method is scalable to a large number of processors on a supercomputer and to quickly interface with a large size database of individual behavior.

10.2.1 General Formulation of Class of Models and Solution Method

We first describe a general class of models that are very amenable to big data solution techniques. Let a mathematical model be represented by a general system

[1] Nonconvex optimization problems are problems in which the set of possible solutions is a nonconvex set. Androulakis et al. [1] provide a nice introduction and references to general nonconvex optimization problems, as well as common examples. See also [7].

[2] See [5, Chap. 9] on quasi-Monte Carlo methods for an introduction to the uses of equidistributed sequences.

of equations $\boldsymbol{F}(\boldsymbol{x}, \boldsymbol{\theta})$, where $\boldsymbol{x}$ is a vector of endogenous variables, $\boldsymbol{\theta}$ is a vector of model parameters, and $\boldsymbol{F}$ is a vector of functions, each of which operates on some subset of $\boldsymbol{x}$ and $\boldsymbol{\theta}$. Let the solution to that system of equations be a particular vector $\hat{\boldsymbol{x}}(\boldsymbol{\theta})$ such that,

$$\hat{\boldsymbol{x}}(\boldsymbol{\theta}) \equiv \text{argmin}_{\boldsymbol{x}} \boldsymbol{F}(\boldsymbol{x}, \boldsymbol{\theta}). \tag{10.1}$$

In other words, $\hat{\boldsymbol{x}}(\boldsymbol{\theta})$ is a solution to the model given a particular set of model parameters. The specification in (10.1) could also be written as a maximization problem and is general enough to include the solution $\hat{\boldsymbol{x}}$ being the root of the vector of equations $\boldsymbol{F}$.

Optimal policy problems often take the form of choosing a subset of the parameter vector $\boldsymbol{\theta}$ to minimize (or maximize) some scalar-valued function W of a vector of scalar valued functions $\boldsymbol{G}$ of the optimized model equations,

$$\hat{\boldsymbol{\theta}} \equiv \text{argmin}_{\boldsymbol{\theta}} W\left(\boldsymbol{G}\left(\boldsymbol{F}\left(\hat{\boldsymbol{x}}(\boldsymbol{\theta}), \boldsymbol{\theta}\right)\right)\right) \tag{10.2}$$

where $\hat{\boldsymbol{x}}(\boldsymbol{\theta})$ is defined in (10.1). If $\boldsymbol{G}$ is vector-valued, then the problem is a multi-objective programming problem. The solution to the policy problem (10.2) is a particular parameterization of the model $\hat{\boldsymbol{\theta}}$ and the model being solved for that particular parameterization $\boldsymbol{F}\left(\hat{\boldsymbol{x}}(\hat{\boldsymbol{\theta}}), \hat{\boldsymbol{\theta}}\right)$.

When both the minimization problem in (10.1) and in (10.2) are convex, the solution $\hat{\boldsymbol{\theta}}$ is straightforward to find with standard computational methods. However, when either (10.1) or (10.2) is a nonconvex optimization problem, finding a solution becomes very difficult and nonstandard computational approaches must be used. Introducing nonconvex structures into the economic model $\boldsymbol{F}$ in (10.1)—such as occasionally binding constraints or nonconvex budget sets—will render the minimization problem in (10.1) nonconvex, thereby making it likely that the minimization problem in (10.2) is not convex. But more subtle model characteristics, such as heterogeneity among the equations in $\boldsymbol{F}$, can maintain the convexity of the minimization problem in (10.1), but break it in (10.2).

10.2.2 Computational Steps to Big Data Solution Method

Our big data approach to solving theory problems, such as the one described in (10.1) and (10.2), is summarized in Table 10.1. The first step is to make a large database of model solutions and objective realizations $\boldsymbol{G}_n\left(\boldsymbol{F}_n\left(\hat{\boldsymbol{x}}(\boldsymbol{\theta}_n), \boldsymbol{\theta}_n\right)\right)$, where $\boldsymbol{\theta}_n$ is the nth realization of the parameter vector $\boldsymbol{\theta}$. The total number of parameter vector realizations N for which the problem is solved is presumably large. One reason why this is a big data solution technique is that the database of model objectives and solutions for each N parameter vectors can be so large that it must be distributed across multiple hard drives. This is step 1 in Table 10.1.

Table 10.1 General summary of big data approach to theory problems

Step description	Output
1. Solve the model for a large number N of parameter vector realizations $\boldsymbol{\theta}_n$.	$\boldsymbol{G}_n\left(\boldsymbol{F}_n(\hat{\boldsymbol{x}}(\boldsymbol{\theta}_n), \boldsymbol{\theta}_n)\right)$
2. Delete all realizations of the objectives vector $\boldsymbol{G}_{n'}$ that are strictly dominated by at least one other realization of the objectives vector $\boldsymbol{G}_n$.	Frontier of $\boldsymbol{G}_n\left(\boldsymbol{F}_n(\hat{\boldsymbol{x}}(\boldsymbol{\theta}_n), \boldsymbol{\theta}_n)\right)$
3. If the frontier from step (2) is not smooth enough, draw P new realizations of the parameter vector $\boldsymbol{\theta}_{n,p}$ in the neighborhood of each remaining point n on the objective vector frontier.	$\boldsymbol{G}_{n,p}\left(\boldsymbol{F}_{n,p}(\hat{\boldsymbol{x}}(\boldsymbol{\theta}_{n,p}), \boldsymbol{\theta}_{n,p})\right)$
4. Delete all realizations of the objectives vector $\boldsymbol{G}_{n',p'}$ that are strictly dominated by at least one other realization of the objectives vector $\boldsymbol{G}_{n,p}$	Frontier of $\boldsymbol{G}_{n,p}\left(\boldsymbol{F}_{n,p}(\hat{\boldsymbol{x}}(\boldsymbol{\theta}_{n,p}), \boldsymbol{\theta}_{n,p})\right)$
5. Repeat refinement steps (3) and (4) until frontier is smooth.	
6. If the objective function W is known, solve for optimal parameter vector $\hat{\boldsymbol{\theta}}$ using Eq. (10.2).	$\hat{\boldsymbol{\theta}} = \text{argmin}_{\boldsymbol{\theta}} \ldots$ $W\left(\boldsymbol{G}_{n,p}\left(\boldsymbol{F}_{n,p}(\hat{\boldsymbol{x}}(\boldsymbol{\theta}_{n,p}), \boldsymbol{\theta}_{n,p})\right)\right)$

In generating our database of individual responses $\hat{\boldsymbol{x}}(\boldsymbol{\theta}_n)$ and the corresponding vector of objectives $\boldsymbol{G}_n$ for realization of the parameter vector $\boldsymbol{\theta}_n$, we used the Python programming language. We also use Python's MPI (message passing interface) library `mpi4py` for simultaneously running computations on multiple processors. We ran this code on 12 nodes with a total of 96 processors on the supercomputer at the Fulton Supercomputing Lab at Brigham Young University.[3] In addition to parallel computation, we also exploited Python's library `h5py`, which enables the HDF5 suite of data and file formats and parallel I/O tools.[4] The Python code in Fig. 10.1 shows some of the key operations that generate our database of responses in the Sales Tax example of [2] described in Sect. 10.3. The code in Fig. 10.1 is taken from multiple Python scripts that work together to solve many model solutions simultaneously solve many sets of model equations for carefully load-balanced sections of policy parameter space.

Lines 2 to 4 in Fig. 10.1 import Python's implementation of MPI, define an MPI communicator, and define a barrier that helps in load balancing. The number of processors to be used as well as the wall time allowed for the computation are

[3]See https://marylou.byu.edu/ for information about the Fulton Supercomputing Lab at Brigham Young University.

[4]See https://www.hdfgroup.org/HDF5/ for a general description of the HDF5 set of tools, and see http://www.h5py.org/ for a description of the Python library which enables the HDF5 tools.

```
1        ...
2        from mpi4py import MPI
3        comm = MPI.COMM_WORLD
4        comm.Barrier()
5        start = MPI.Wtime()
6        ...
7        import h5py
8        ...
9        def init_database(comm, N, sequence_type, n_policies,
             type_space, policy_space, tax_rates_same, filename,
             verbose=True):
10       ...
11       with h5py.File(filename, 'w') as f:
12       ...
```

Fig. 10.1 Python code importing `mpi4py` and `h5py` modules

part of a supercomputer-specific job script which tells the supercomputer to start running the job. Once MPI has been enabled for use on multiple processors, we import the HDF5 set of tools with the `import h5py` call. Line 9 of Fig. 10.1 shows one function `init_database()` that is run in parallel for as many Python instances as we have told the supercomputer to create. Each separate instance is designated by the `comm` object in the function. This function computes the solutions to the model $\hat{\boldsymbol{x}}(\boldsymbol{\theta}_n)$ and $\boldsymbol{G}_n$ and then saves those solutions using the HDF5 parallel I/O functionality of store commands after the command in line 11 of code `with h5py.File(filename, 'w') as f:`.

An arguably more important reason for using big data techniques has to do with the manipulation of the database of model objectives and solutions after its creation. Once the database of $\boldsymbol{G}_n\left(\boldsymbol{F}_n(\hat{\boldsymbol{x}}(\boldsymbol{\theta}_n), \boldsymbol{\theta}_n)\right)$ exists, note that we still have not solved for the optimal parameter vector $\hat{\boldsymbol{\theta}}$. In terms of the database, we want to know what is the $\boldsymbol{\theta}_n$ that minimizes some function of the objectives W from (10.2). HDF5 parallel input-output techniques are very efficient at operations across distributed memory such as weighted averages, nonlinear functions, minima, and maxima.

But now assume that you are not sure what the correct objective aggregating function W is. In our approach, we search our database of responses and delete all entries in the database $\boldsymbol{G}_{n'}$ that are strictly dominated by another entry $\boldsymbol{G}_n$. The first panel in Fig. 10.2 shows the entire database of objective realizations for each $\boldsymbol{\theta}_n$ where there are two objectives (each $\boldsymbol{G}_n$ has two elements). In other words, each dot represents $G_{1,n}$ and $G_{2,n}$ for a given vector of parameters $\boldsymbol{\theta}_n$. The second panel in Fig. 10.2 shows the points on the frontier in terms of $G_{1,n}$ and $G_{2,n}$ for all n after deleting all the strictly dominated points.

The execution in our code of this deletion of strictly dominated points, as shown in the first two panels of Fig. 10.2, is something that we have had success in speeding

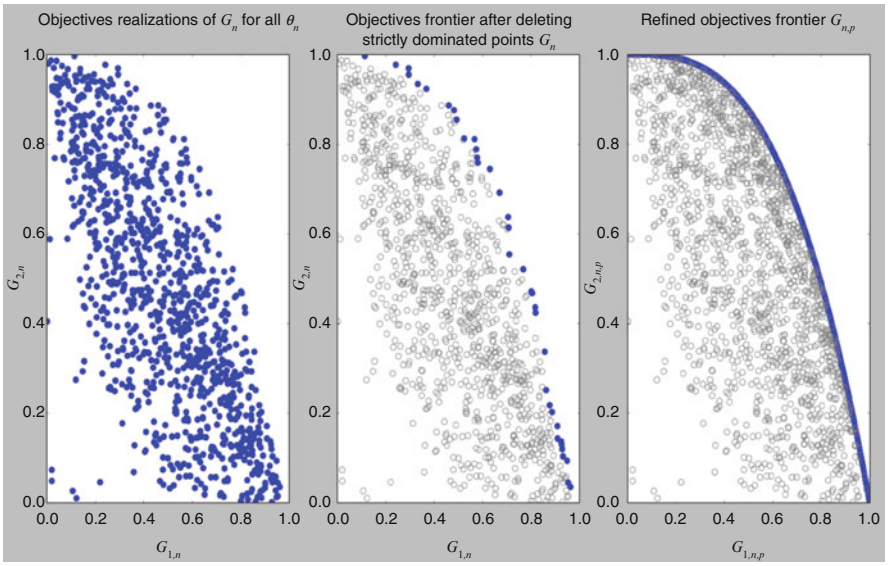

Fig. 10.2 General process of finding the objectives $\boldsymbol{G}_n$ frontier

up and optimizing. We use a parallel quicksort routine in which we sort portions of the points in the first panel along one dimension of objectives $\boldsymbol{G}_{1,n}$ and then delete data points in which the $\boldsymbol{G}_{2,n}$ objective is dominated.

Often times, the frontier traced out in the first deletion of strictly dominated points is too coarse. In that case, we do another step of choosing equidistributed sequences of new realizations of $\boldsymbol{\theta}_n$ in the neighborhood of each point on the frontier. We then delete all the strictly dominated objectives from those new realizations to find the refined frontier shown in the third panel of Fig. 10.2.

10.2.3 Virtues of Equidistributed Sequences

In choosing a grid size N of realizations of the parameter vector $\boldsymbol{\theta}_n$ and in refining around points on the objective vector frontier $\boldsymbol{\theta}_{n,p}$, it is important to have an efficient method to keep track of all the points that are generated and which points get saved in the database. Using equidistributed sequences provides that efficiency, both in terms of spreading N points uniformly throughout a particular space and in keeping track of those points.

Equidistributed sequences are deterministic sequences of real numbers where the proportion of terms falling in any subinterval is proportional to the length of that interval. A sequence $\{x_j\}_{j=1}^{\infty} \subset D \subset \mathbb{R}^n$ is equidistributed over D if and only if

Table 10.2 Equidistributed sequences in $\mathbb{R}^n$

Name of sequence	Formula for $(x_1, x_2, \ldots, x_n)$
Weyl	$(\{np_1^{1/2}\}, \ldots, \{np_n^{1/2}\})$
Haber	$(\{\frac{n(n+1)}{2}p_1^{1/2}\}, \ldots, \{\frac{n(n+1)}{2}p_n^{1/2}\})$
Niederreiter	$(\{n2^{1/(n+1)}\}, \ldots, \{n2^{n/(n+1)}\})$
Baker	$(\{ne^{r_1}\}, \ldots, \{ne^{r_n}\})$, r_j rational and distinct

$$\lim_{n\to\infty} \frac{\mu(D)}{n} \sum_{j=1}^{n} f(x_j) = \int_D f(x)dx \tag{10.3}$$

for all Riemann-integrable $f(x) : \mathbb{R}^n \to \mathbb{R}$, where $\mu(D)$ is the Lebesgue measure of D.

There are a number of equidistributed sequences that possess this property. Let $p_1, p_2, \ldots$ denote the sequence of prime numbers $2, 3, 5, \ldots$, and let $\{x\}$ represent the fractional part of x, that is $\{x\} = x - \lfloor x \rfloor$. Table 10.2 contains examples of a number of equidistributed sequences. Figure 10.3 shows the first 10,000 points for two-dimensional Weyl, Haber, Niederreiter, and Baker sequences.

Baker et al. [2] and Bejarano et al. [3] use a scaled Baker sequence. Quasi-Monte Carlo integration is used to integrate over the type space for each point in policy space given the type space distribution. Quasi-Monte Carlo integration is similar to Monte Carlo integration, but chooses points using equidistributed sequences instead of pseudorandom numbers. This allows for a faster rate of convergence for a large number of points. With N points in s dimensions, quasi-Monte Carlo techniques converge in $O\left(\frac{(\log N)^s}{N}\right)$ as opposed to $O\left(\frac{1}{\sqrt{N}}\right)$ for Monte Carlo techniques.[5]

A key distinction between equidistributed sequences and pseudorandom sequences is that equidistributed sequences do not look like random numbers. As can be seen in Fig. 10.3, they generally display substantial serial correlation. From the outset, equidistributed sequences are chosen so as to perform accurate integration, and are not encumbered by any other requirements of random numbers

Another practical advantages of using equidistributed sequences is that it allows one to represent the entire multi-dimensional space of parameters $\boldsymbol{\theta}$ in the minimization problem (10.2) as a one-dimensional list, which allows for easy partitioning across computing nodes. Additionally, using equidistributed sequences makes for easy expansion of the database. One has merely to append additional points to the end of the list.

[5] See [5, Chap. 9] on quasi-Monte Carlo methods for a more thorough discussion of the advantages of using equidistributed sequences to execute simulation-based methods, Riemann-integrable functions, and Lebesgue measure.

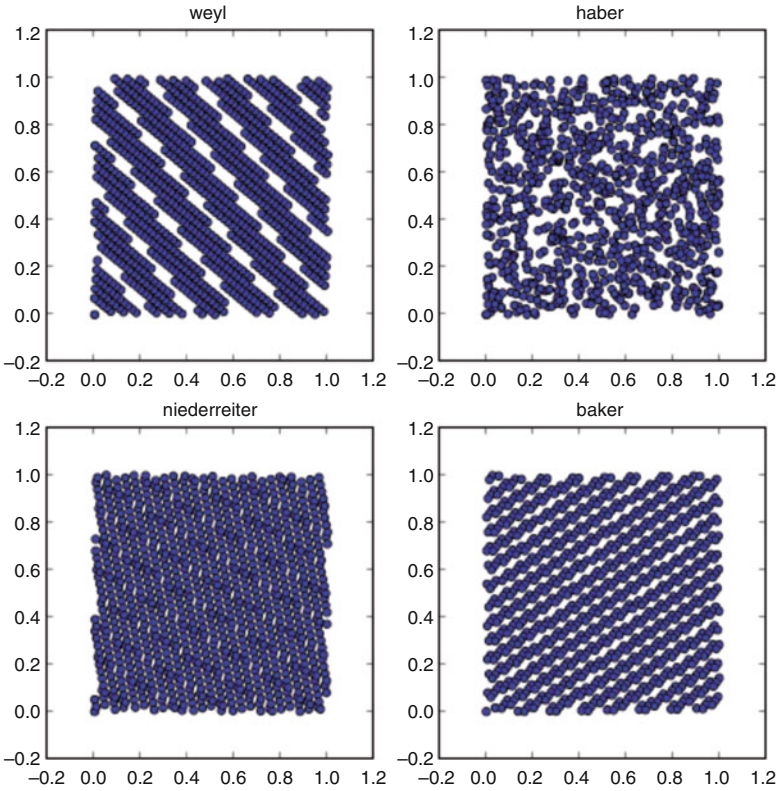

Fig. 10.3 Four two-dimensional equidistributed sequences with $n = 10{,}000$

10.3 Optimal Tax Application

In this section, we highlight an application of the approach presented in Sect. 10.2 to a theory problem related to optimal taxation. This example is described in [2] and solves for an optimal schedule of sales tax rates given a heterogeneous population of consumers. In this sales tax problem, each individual's optimization problem—analogous to the general problem in (10.1)—is convex. However, because of the heterogeneity across individuals, the policy maker's problem in choosing the optimal tax—analogous to the general problem in (10.2)—is not convex.

Baker et al. [2] set up an economic environment in which a policy maker must choose a schedule of sales tax rates on the different types of goods consumed by households and in which the population of households differs in terms of their wage and their elasticity of substitution among eight different types of consumption goods. Total consumption by a given household C is a constant elasticity of substitution (CES) function of all the individual types of consumption c_i the household can choose,

$$C \equiv \left(\sum_{i=1}^{8} \alpha_i (c_i - \bar{c}_i)^{\frac{\eta-1}{\eta}} \right)^{\frac{\eta}{\eta-1}} \quad \forall \eta \geq 1 \tag{10.4}$$

where $\eta \geq 1$ is the elasticity of substitution among all of the consumption goods, $\alpha_i \in [0, 1]$ is the weight on the consumption of each type of good with $\sum_i \alpha_i = 1$, and $\bar{c}_i \geq 0$ is a minimum level of consumption for each type of good.

Household's face a budget constraint in which their total expenditure on consumption goods must be less-than-or-equal-to their income, which in this case is simply their wage.

$$\sum_{i=1}^{8} (1 + \tau_i) c_i \leq w \tag{10.5}$$

The household's objective function is a Constant Relative Risk Aversion (CRRA) utility function defined over total consumption from (10.4),

$$u(C) = \frac{C^{1-\gamma} - 1}{1 - \gamma} \tag{10.6}$$

where $\gamma \geq 1$ is the coefficient of relative risk aversion. The household's optimization problem is to choose a vector of consumptions $\boldsymbol{c} = \{c_1, c_2, \ldots c_8\}$ that maximizes total utility (10.6) subject to the budget constraint (10.5).

Let a household's type be defined by its wage w and its elasticity of substitution η. The household's problem is a convex optimization problem where the solution is a vector of consumption functions $\boldsymbol{c}(w, \eta; \boldsymbol{\tau})$ that are functions of a household's type (w, η) and the vector of sales tax rates $\boldsymbol{\tau} = \{\tau_1, \tau_2, \ldots \tau_8\}$ as well as a utility function $u\Big(\boldsymbol{c}(w, \eta; \boldsymbol{\tau})\Big)$ that is also a function of household type (w, η) and the vector of tax rates $\boldsymbol{\tau}$. This problem is analgous to the problem in (10.1) in Sect. 10.2.

For any given sales tax regime $\boldsymbol{\tau}$, there are as many different household utility levels $u\Big(\boldsymbol{c}(w, \eta; \boldsymbol{\tau})\Big)$ and optimal consumption vectors $\boldsymbol{c}(w, \eta; \boldsymbol{\tau})$ as there are different types of individuals. Baker et al. [2] then choose 57,786 different sales tax policy vectors $\boldsymbol{\tau}$ with 5,100 different types of individuals resulting in solving nearly 300 million individual optimization problems.[6] They assume that the policy maker chooses the optimal sale tax schedule $\boldsymbol{\tau}$ to maximize some combination of the total utility in the economy (the sum of individual utilities u) and total tax revenue. Each point along the solid curve in Fig. 10.4 represents a sales tax policy $\boldsymbol{\tau}$ that is on the frontier in terms of both total utility (x-axis) and total revenue (y-axis).

Figure 10.5 shows how the optimal sales tax policies $\boldsymbol{\tau}$ change across the frontier in Fig. 10.4. That is, sales tax rates at the left-hand-side of Fig. 10.5 correspond to

[6]Table 3 in [2] details that this computation took 29.5 h of wall time using 96 processors. In serial, this would have taken nearly 3,000 h (about 120 days), which is not feasible.

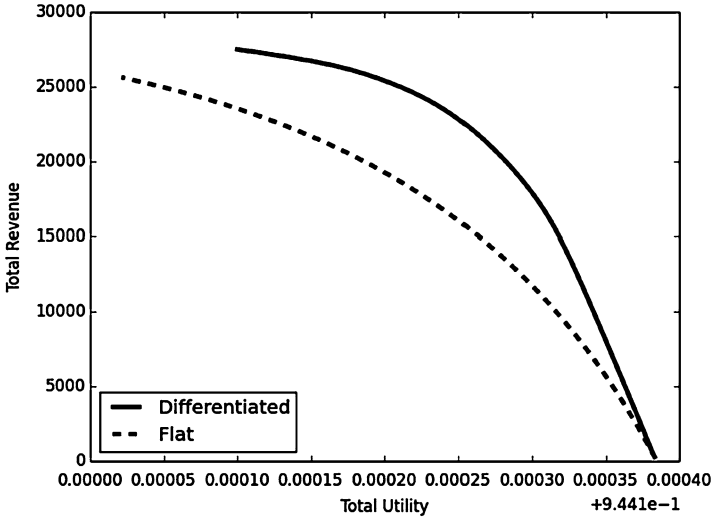

Fig. 10.4 Total utility-revenue frontiers for optimal differentiated tax versus optimal flat tax

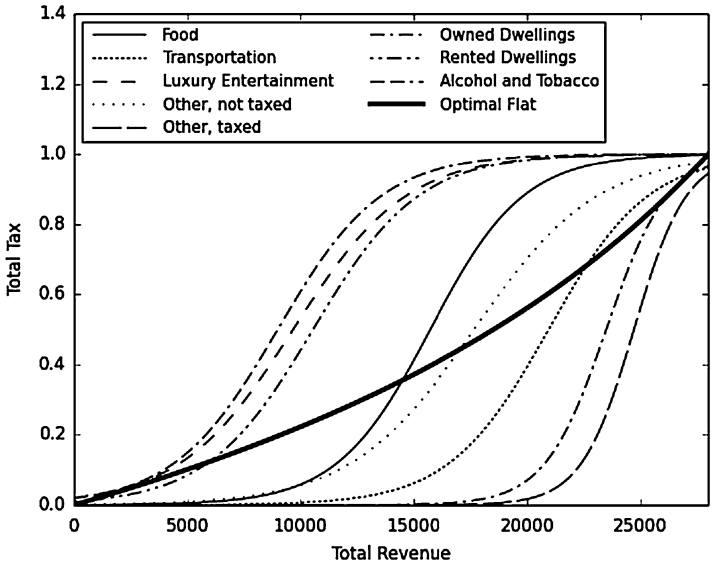

Fig. 10.5 Optimal tax rates for good *i* for different levels of total revenue

points on the frontier in the lower-right side of Fig. 10.4. Figure 10.5 shows which taxes get increased first in the most optimal sales tax schedule as more total revenue is required. Figure 10.5 represents the entire set of optimal sales tax schedules $\hat{\boldsymbol{\tau}}$ for many different possible policy maker objective functions W over the two objectives of total utility and total revenue.

Baker et al. [2] use this framework to test the welfare and total revenue effects of one optimal sales tax rate on all consumption goods instead of a set of optimal tax rates on each consumption good. They also estimate the loss in revenue from exempting a class of consumptions goods, such as some services are in the U.S. economy. They find that there is only a small loss in total revenue from exempting services from sales taxation. However, that loss is small only because other taxes are higher in order to make up for the exempted category. Further they find a 30 % loss in total tax revenue from a sales tax regime with only one optimally chosen tax rate versus one in which multiple sales tax rates on goods are chosen optimally.

10.4 Other Applications

Multi-objective, nonconvex optimal policy problems abound of the form described in Sect. 10.2. Optimal insurance contracts, political institutions, problems with occasionally binding constraints, auction and mechanism design, and maximum likelihood estimation are a few examples. But the big data approach in this paper is also well-suited for models in which the individual problem from Eq. (10.1) takes a long time to run for any given parameter vector $\boldsymbol{\theta}_n$. The strength of this big data approach to these problems is that the model can be solved independently and in parallel for a grid of points in parameter space $\boldsymbol{\theta}_n$. These solutions can be stored and used by later researchers. These later researchers can either get their solution by interpolating between the stored solutions, or by adding to the database in regions in parameter space $\boldsymbol{\theta}_n$ that are too sparse. To describe this class of problems, we define a slightly different version of the model from Sect. 10.2.

Let an econometric model $\boldsymbol{F}(\boldsymbol{\beta}, \boldsymbol{\theta})$ be defined over a vector of exogenous parameters $\boldsymbol{\beta}$, whose values are taken from outside the model, and a vector of endogenous parameters $\boldsymbol{\theta}$, whose values are estimated by the model.

$$\hat{\boldsymbol{\theta}} \equiv \text{argmin}_{\boldsymbol{\theta}} \boldsymbol{F}(\boldsymbol{\beta}, \boldsymbol{\theta}) \tag{10.7}$$

The solution to the problem (10.7) is an optimal parameter vector $\hat{\boldsymbol{\theta}}(\boldsymbol{\beta})$ as a function of exogenous parameters and a model solution $\boldsymbol{F}\left(\boldsymbol{\beta}, \hat{\boldsymbol{\theta}}(\boldsymbol{\beta})\right)$ as a function of exogenous parameters.

One immediate econometric application of this big data approach is when the minimization problem (10.7) is nonconvex, given an exogenous parameter vector $\boldsymbol{\beta}$. This is often the case in maximum likelihood estimation. Through a large database of potential endogenous parameter vector $\boldsymbol{\theta}_n$ and refinements around the objective function frontier, confidence that $\hat{\boldsymbol{\theta}}$ is the global optimum increases as the size of the database increases. This is simply a non-derivative optimizer that uses an adaptive grid search method.

However, this method becomes very valuable when each computation of a solution $\hat{\boldsymbol{\theta}}(\boldsymbol{\beta})$ for a given exogenous parameter vector $\boldsymbol{\beta}$ takes a long time. Many

instances of maximum likelihood estimation fit this criterion. Imagine a model that estimates region-r specific parameters $\boldsymbol{\theta}_r$ for a given calibration of other parameters $\boldsymbol{\beta}_r$ for that region. Each estimation for a given region r and its calibrated parameters $\boldsymbol{\beta}_r$ might take a long time. If we have stored a database of precomputed solutions $\hat{\boldsymbol{\theta}}_r(\boldsymbol{\beta}_r)$, then one can simply interpolate the estimation of a new region rather than computing the solution again.

More generally, maximum likelihood estimation of the problem in (10.7) for one particular realization of the exogenous parameters $\boldsymbol{\beta}$ might require a long computational time (sometimes days). If one were to precompute once and store in a database the solutions to the problem $\hat{\boldsymbol{\theta}}(\boldsymbol{\beta})$ for a grid of potential exogenous parameter realizations, solutions on the continuous space of potential exogenous parameter vector realizations $\boldsymbol{\beta}$ could be quickly computed by interpolating between points saved in the database. One study for which this technique is currently being employed is [4], which describes a difficult maximum likelihood estimation of a quantile regression.

10.5 Conclusion

Computing capability is ever becoming more powerful, less costly, and more broadly available. At the same time, the techniques to store and interact with large datasets are also improving. We have described a novel approach to solving complex minimization problems that combines the tools of MPI for parallel processing the tools of parallel I/O for using big data techniques. This approach allows researchers to solve optimal policy problems that are otherwise too complex. A leading example is the optimal income tax problem from Sect. 10.3. This method has myriad other applications ranging from optimal insurance contracts to maximum likelihood estimation.

Acknowledgements We thank the Hoover Institution and the BYU Macroeconomics and Computational Laboratory for research support. We also thank the Mary Lou Fulton Supercomputing Laboratory at Brigham Young University for use of the supercomputer.

References

1. I.P. Androulakis, C.D. Maranas, C.A. Floudas, αBB: a global optimization method for general constrained nonconvex problems. J. Glob. Optim. **7**(4), 337–363 (1995)
2. C. Baker, J. Bejarano, R.W. Evans, K.L. Judd, K.L. Phillips, A big data approach to optimal sales taxation. NBER Working Paper No. 20130, National Bureau of Economic Research (May 2014)
3. J. Bejarano, R.W. Evans, K.L. Judd, K.L. Phillips, K. Quist, A big data approach to optimal income taxation. Mimeo (2015)

4. B.R. Frandsen, Exact nonparametric inference for a binary endogenous regressor. Mimeo (2015)
5. K.L. Judd, *Numerical Methods in Economics* (MIT, Cambridge, 1998)
6. E. Liran, J.D. Levin, The data revolution and economic analysis. NBER Working Paper No. 19035, National Bureau of Economic Research (2013)
7. S.K. Mishra (ed.), *Topics in Nonconvex Optimization: Theory and Applications*. Nonconvex Optimization and Its Applications (Springer, New York, 2011)
8. H.R. Varian, Big data: new tricks for econometrics. J. Econ. Perspect. **28**(2), 3–28 (2014)
9. T. White, *Hadoop: The Definitive Guide* (O'Reilly Media; Sebastopol, California, 2012)