Documentation Code

Ignacio Rios

Stanford GSB

May 10, 2018

1 Introduction

This repository contains files to simulate results for the paper *Procurement Mechanisms for Differentiated Products*, by Daniela Saban and Gabriel Weintraub.

I implemented it in the programming language Julia, because of its capabilities to formulate and solve optimization problems. In particular, I use the library JuMP, which makes the formulation of problems much easier. In addition, to have Julia and JuMP, the other requirements are installing the solvers Gurobi and Knitro:

- Gurobi: general solver for linear and mixed integer problems. I use it to solve the centralized problem. It can be easily installed from the Gurobi website, and they have academic licenses for free. Once you install it, you have to also install a Julia package to use it.
- Knitro: general solver for non-linear problems. I use it to solve the decentralized problem. This solver can also be installed and there is a trial license for free (which lasts around 6 months). An important feature of this solver is the number of starting points used during the optimization, since this helps to avoid local optimal solutions. So far I implemented it with 500 starting points and it is working well with that; however, it takes some time to solve. You may want to decrease that to speed up, but you may get worst results.

2 Repository

The repository Julia contains four main files:

- HM.jl: implementation of the Hotelling model.
- LDM.jl: implementation of the general affine demand model.

- SimulationsHM.jl: implementation to simulate different instances of the Hotelling model.
- SimulationsLM.jl: implementation to simulate different instances of the general affine demand model.
- utilities.jl: contains a series of methods to compute demands, virtual costs, and check assumptions.
- procesingresults.jl: contains a series of methods to process the results of the simulations.
- exmaples.jl: tests solutions with close forms to contrast with the results from optimization.

In what follows I describe some details that are relevant in each file.

3 HM.jl

This file implements the Hotelling demand model. Given $\Theta = (\theta_1, \dots, \theta_m)$ and n suppliers, the structure of the variables is

$$x(\theta) = [x_1(\theta_1), \dots, x_n(\theta_1), x_1(\theta_2), \dots, x_n(\theta_2), \dots, x_1(\theta_m), \dots, x_n(\theta_m)]\,.$$

The same holds for the other variables, i.e. t, p, and also for the dual variables u and v.

The formulation of the centralized problem follows directly from the paper, considering as objective function

$$\max \mathbb{E}_\theta \left[\sum_{i=1}^{n} -\frac{\delta}{2} \left[\left(l_i - \sum_{j=1}^{i-1} x_j(\theta) \right)^2 + \left(\sum_{j=1}^{i} x_j(\theta) - l_i \right)^2 \right] \right]$$

In the case with two suppliers in the extremes of [0,1] this reduces to

$$\max \mathbb{E}_\theta \left[-\frac{\delta}{2} \left[x_1(\theta)^2 + x_2(\theta)^2 \right] - t_1(\theta) - t_2(\theta) \right]$$

To implement the decentralized problem, we incorporate the KKT conditions as additional constraints to the centralized problem. In the two-supplier case these are: for each $\theta \in \Theta$ and $i \in \{1, 2\}$

$$\begin{aligned} p_i(\theta) + \delta x_i(\theta) - u_i(\theta) + v(\theta) &= 0 \\ x_i(\theta) \cdot u_i(\theta) &= 0 \\ u_i(\theta) &\geq 0 \\ t_i(\theta) - x_i(\theta) \cdot p_i(\theta) &= 0 \end{aligned}$$

where $\theta = (\theta_i, \theta_{-i})$. Primal feasibility constraints are omitted because they are part of the centralized problem. Moreover, when I say that ex-post IR constraints are considered, I mean that the following constraints are added:

$$t_i(\theta_i, \theta_{-i}) - \theta_i x_i(\theta_i, \theta_{-i}) \geq 0, \ \forall \theta_i, \theta_{-i}.$$

When dealing with the decentralized problem, these constraints can be formulated as

$$\left(p_i(\theta_i, \theta_{-i}) - \theta_i\right) x_i(\theta_i, \theta_{-i}) \geq 0, \ \forall \theta_i, \theta_{-i}.$$

This file has the following methods:

- GenerateInputs: returns the matrices and vectors to construct the constraints and objective of the optimization problem. Some special elements of this method are:
  - line 12: generates the set of types Θ, i.e. all possible combinations of types of suppliers.
  - line 27-34: represents computation of $f_{-i}(\theta_{-i})$.

  The outputs of this method are:
  - nsupp: number of suppliers
  - ntypes: number of types for each supplier;
  - nvars: length of each vector of variables, i.e. $n \times m$
  - sts: number of scenarios, i.e. $|\Theta|$
  - bGx: matrix multiplying x in the inequality IC and IR constraints. This matrix is organized as follows:
    - The first nsupp $\times$ ntypes rows are IR constraints, ordered lexicographically by (i, θ_i) in increasing order, where i is the supplier and θ_i is his type. For example, in a 2 supplier-2 types example, the first row corresponds to supplier 1 with low type, then supplier 1 with high type, then supplier 2 with low type, and finally supplier 2 with high type.
    - The last nsupp $\times$ ntypes $\times$ (ntypes-1) rows correspond to IC constraints, ordered lexicographically by (i, θ_i, θ_i') in increasing order.
  - bGt: matrix multiplying t in the inequality IC and IR constraints
  - bh: right-hand side of IR-IC constraints (equal to a vector of zeros)
  - bA: matrix multiplying x in the feasibility constraints
  - bb: right-hand side of feasibility constraints (equal to a vector of ones)
  - wqt: vector with the probabilities of each scenario in Θ to compute the expected value for objective function
  - f: joint distribution of types
  - Theta: equivalent to Θ, i.e. the set of all types.

- GenerateExPostInputs: receives as input the list of types, the set of type combinations, the number of suppliers and the number of variables. Generates as output the matrices and vectors required to write ex-post IR constraints.
- CheckFeasibility: receives the inputs of the problem and solutions x_0, t_0 and checks whether this solution is feasible for the decentralized problem. To accomplish this, I fix $x = x_0, t = t_0$ and solve the decentralized problem with objective value equal to 1.
- SolveOptimization: formulates and solves the optimization problem using as inputs the parameters of the problem and version, which is equal to centralized or decentralized.

4 LDM.jl

This file implements the general affine linear demand model. The structure of the variables is equivalent to the previous case. The centralized problem follows directly from the paper, while the decentralized problem is implemented by adding the following KKT conditions as constraints (in matrix form):

$$\begin{aligned} p(\theta) - c + Dx(\theta) - u(\theta) + v(\theta) &= 0 \\ x(\theta) \cdot u(\theta) &= 0 \\ u(\theta) &\geq 0 \\ t(\theta) - x(\theta) \cdot p(\theta) &= 0 \end{aligned}$$

As in the Hotelling model, to add ex-post IR constraints we consider the extra constraints

$$\left(p_i(\theta_i, \theta_{-i}) - \theta_i\right) x_i(\theta_i, \theta_{-i}) \geq 0, \ \forall \theta_i, \theta_{-i}.$$

The methods in this file are the same as in the previous case adapted to the general affine demand model. In particular, these methods are:

- GenerateInputs: same as before. Actually, these methods are exactly equivalent to that in the Hotelling case.
- GenerateExPostInputs: same as before. Actually, these methods are exactly equivalent to that in the Hotelling case.
- InputsObjectiveLM: these method generates special inputs for the general affine demand model. In particular, it computes

  $$D = \Gamma^{-1}, \quad c = D\alpha$$

  and also generates the inputs for the objective function of the affine model.
- CheckFeasibility: given a solution to the affine model, it checks whether the solution is feasible.

- SolveOptimization: formulates and solves the optimization problem using as inputs the parameters of the problem and version, which is equal to centralized or decentralized. In addition, it receives the following additional inputs:
  - elastic: boolean variable that is true if we want to solve elastic model, and false if we want to solve inelastic model.
  - expostir: boolean variable that is true if we want to incorporate ex-post IR constraints, and false if we want to consider only interim constraints.
  - x0: vector with initial solution - allocation
  - t0: vector with initial solution - transfers

  Note: the solvers considered are two:
  - Gurobi: used to solve the centralized problem.
  - Knitro: used to solve the decentralized problem. This solver is needed because the decentralized problem is not linear, so Gurobi cannot handle it. When setting the solver, one parameter that is relevant is **msmaxsolves**, which is the number of multi-start points considered. Considering several starting points is important to avoid local optima solutions that are not globally optimal.
- FormulateAndSolve: considers the inputs of the problem to solve both the centralized and the decentralized models, and returns:
  - objective function of the centralized model
  - expected transfer of the centralized model
  - vector of allocation of the centralized model, i.e. $x(\theta),\ \theta \in \Theta$
  - vector of transfers of the centralized model, i.e. $t(\theta),\ \theta \in \Theta$
  - objective function of the decentralized model
  - expected transfer of the decentralized model
  - vector of allocation of the decentralized model
  - vector of transfers of the decentralized model

5 SimulationsHM.jl

The implementation of the Hotelling model considers only two suppliers and any potential number of types. Implementing more suppliers is tricky because the the KKT conditions are messy.

The methods considered in this file are:

- SimulateTwoSuppliers: this method solves the Hotelling model for a list of different δ. The inputs are:
  - Types: list of types Θ_i

- Dictionary of marginal distributions
- Vector of locations for Hotelling model
- Vector of δs to perform simulation.
- elastic: boolean that is true for elastic demand, false for inelastic.
- expostir: boolean that is true for expost IR constraints, false for including only interim IR constraints.

The output generated by this method is a file that contains, for each δ,

- value of δ considered
- objective function of the centralized model
- expected transfer of the centralized model
- vector of allocation of the centralized model, i.e. $x(\theta),\ \theta \in \Theta$
- vector of transfers of the centralized model, i.e. $t(\theta),\ \theta \in \Theta$
- objective function of the decentralized model
- expected transfer of the decentralized model
- vector of allocation of the decentralized model
- vector of transfers of the decentralized model

The execution starts in line 44. There you can write the parameters related to the types of suppliers, the distribution of types, the parameters for transportation cost and location of suppliers, as well as what we want to vary during simulation.

6 SimulationsLM.jl

For the general affine demand case, we can solve the problem for different number of supplier and for several number of types. The methods included in this file are:

- SimulateTwoSuppliersAsymmetric: this method solves the problem for both the centralized and decentralized cases considering two suppliers, for different parameters of quality and demand functions. This method considers all possible combinations of parameters provided as inputs. The inputs are:
  - types: vector of types for each supplier Θ_i
  - fm: marginal distributions of types
  - aj: list of potential values for the quality of suppliers
  - gammaii: list of potential values for the elements on the diagonal of the matrix Γ
  - gammaij: list of potential values for the elements off the diagonal of the matrix Γ

  - elastic: boolean that is true for elastic demand, false for inelastic
  - expostir: boolean that is true for expost IR constraints, false for including only interim IR constraints
  - outdir: directory to store outputs of simulation

- SimulateNSupplierSymmetric: this method solves the symmetric problem for any number of suppliers and types, i.e. it assumes that all suppliers provide the same quality and that the demand matrix is symmetric and all elements in its diagonal are the same. The inputs for this function are the same as before, with an additional one, N, which is the desired number of suppliers.

The output generated by this method is a file that contains, for each combination of parameters (in the 2 suppliers case) $(\alpha_1, \alpha_2, r_1, r_2, \gamma)$,

- values of $\alpha_1, \alpha_2, r_1, r_2, \gamma$ considered
- objective function of the centralized model
- expected transfer of the centralized model
- vector of allocation of the centralized model, i.e. $x(\theta),\ \theta \in \Theta$
- vector of transfers of the centralized model, i.e. $t(\theta),\ \theta \in \Theta$
- objective function of the decentralized model
- expected transfer of the decentralized model
- vector of allocation of the decentralized model
- vector of transfers of the decentralized model

In the symmetric case the outputs look the same, but instead each row starts with α, r, γ (remember $r_1 = r_2 = r, \alpha_1 = \alpha_2 = \alpha$).

The execution starts in line 44. There you can write the parameters related to the types of suppliers, the distribution of types, the parameters for transportation cost and location of suppliers, as well as what we want to vary during simulation.

7 utilities.jl

This file contains a series of methods that are used in the previous files. These methods are:

- combine and combwithrep: used to generate all potential profiles in Θ.
- ComputeCumulativeDistribution: for a given dictionary of marginal distributions, returns a disctionary of marginal cumulative distributions.

- virtualcost: computes, for a given supplier i and type θ, the virtual cost.
- demandLM: computes the demand faced by supplier i in the general affine model with two suppliers (obtained from close form solution).
- assortmentHM: returns the set of suppliers in the assortment of the Hotelling model for a given price and δ.
- demandHM: computes the demand faced by supplier i in the Hoteeling model with two suppliers.
- checkvcincreasing: checks whether the virtual costs are increasing for each supplier.
- checkconditionsHM: check whether the conditions in Theorem 4.1 are satisfied; returns a list with two booleans, one for each condition.
- checkconditionsLM: check whether the conditions in Theorem 5.2 are satisfied; returns a list with two booleans, one for each condition. To find d^*, we use the following heuristic:

  1. Divide Θ in two subsets Θ^S and Θ^N, such that

  $$\Theta^S = \{\theta \in \Theta : \ x_i(\theta) > 0, \ \forall i\}$$
  $$\Theta^N = \{\theta \in \Theta : \ \exists i \ st. \ x_i(\theta) = 0\}$$

  2. For each $\theta \in \Theta^S$, compute

  $$d(\theta) = \max\left\{\left|v_i(\theta) - v_i(\theta')\right| : \theta' \in \Theta^N, \ i = 1, \ldots, n\right\}$$

  3. Define

  $$d^* = \min\left\{d(\theta) : \ \theta \in \Theta^S\right\}$$

8 processingresults.jl

This file contains function that allow to read the outputs generated during simulation, and analyze different statistics or results.

9 examples.jl

This file tests the methods described previously.