Solving Stochastic Equilibria: EMP, SELKIE, and Optimal Value Functions

Michael C. Ferris

University of Wisconsin, Madison

(Joint work with Olivier Huber and Youngdae Kim)

Oberwolfach, Germany
August 24, 2018

The issue

- My problem has a solution, why does your solver fail to find it?
- Inappropriate application of solver
- Reality: Poor implementations of nice problems are much harder
- Reality: Solver cannot detect structure of problem to exploit computationally
- Without loss of generality, we can assume our problem is in the following standard form...
- Reality: Real problems are messy
- Our algorithm solves these (nonstandard) problems well - how can we make it available?

Let modelers formulate their problem naturally, with appropriate mathematical constructs, convey known problems structures, and automate problem transformations for solution engines

Equilibrium problems (GNEPs)

- Generalized Nash equilibrium problems (GNEPs)

$$\begin{array}{lll} \text{find} & (x_1^*, \dots, x_N^*) & \text{satisfying,} \\ x_i^* \in & \mathrm{argmin}_{x_i} & \theta_i(x_i, x_{-i}^*), \\ & \mathrm{s.t.} & h_i(x_i, x_{-i}^*) = 0, \\ & & g_i(x_i, x_{-i}^*) \leq 0, \end{array}$$

  where

$$x_{-i} = (x_1, \dots, x_{i-1}, x_{i+1}, \dots, x_N).$$

- If interactions occur only in objectives, then it becomes a NEP.

Equilibrium problems (MOPECs)

- GNEP + VI agent: MOPEC

$$\begin{array}{lll} \text{find} & (x_1^*, \ldots, x_N^*, \pi^*) & \text{satisfying,} \\ x_i^* \in & \mathrm{argmin}_{x_i} & \theta_i(x_i, x_{-i}^*, \pi^*), \\ & \mathrm{s.t.} & h_i(x_i, x_{-i}^*, \pi^*) = 0, \\ & & g_i(x_i, x_{-i}^*, \pi^*) \leq 0, \end{array}$$

$$\pi^* \in \mathrm{SOL}(K(x^*), F(\cdot; x^*))$$

- Add an additional VI agent that solves a variational inequality, i.e. market clearing conditions

$$0 \leq \text{supply} - \text{demand} \quad \perp \quad \pi \geq 0$$

Trading risk: Philpott et al. [2016]

$$\text{CP:} \quad \min_{d^1, d^2_\omega \geq 0, t^C} \quad \sigma t^C + p^1 d^1 - W(d^1) + \rho_C \left[p^2_\omega d^2_\omega - W(d^2_\omega) - t^C_\omega \right]$$

$$\text{TP:} \quad \min_{v^1, v^2_\omega \geq 0, t^T} \quad \sigma t^T + C(v^1) - p^1 v^1 + \rho_T \left[C(v^2_\omega) - p^2_\omega v^2(\omega) - t^T_\omega \right]$$

$$\text{HP:} \quad \min_{\substack{u^1, x^1 \geq 0 \\ u^2_\omega, x^2_\omega \geq 0, t^H}} \quad \sigma t^H - p^1 U(u^1) + \rho_H \left[-p^2(\omega) U(u^2_\omega) - V(x^2_\omega) - t^H_\omega \right]$$

$$\text{s.t.} \quad x^1 = x^0 - u^1 + h^1,$$

$$x^2_\omega = x^1 - u^2_\omega + h^2_\omega$$

$$0 \leq p^1 \perp U(u^1) + v^1 \geq d^1$$

$$0 \leq p^2_\omega \perp U(u^2_\omega) + v^2_\omega \geq d^2_\omega, \forall \omega$$

$$0 \leq \sigma_\omega \perp t^C_\omega + t^T_\omega + t^H_\omega \geq 0, \forall \omega \quad \sigma = (\sigma_\omega)$$

Issues with specifying equilibrium problems

- How to specify equilibrium problems in modeling languages?
  - Abstractly, a model is defined by a set of variables and equations.
    - ex) a GNEP model: $m := \{(x_i, \theta_i, h_i, g_i)\}_{i=1}^N$
  - No constructs exist to specify equilibrium problems or variational inequalities:
    - `solve m using lp min obj` will not work.
  - Existing way: formulate an MCP(B, F)
    - Specify MCP using `.` in GAMS or `complements` in AMPL.
    - However, we'll lose agent information.
    - Modeler has to compute derivatives (KKT).
  - A new set of constructs are needed to specify *agent information* (who controls what).

The EMP framework for equilibrium problems

- The EMP framework: Ferris et al. [2009], Kim and Ferris [2018b]
  - Annotate agent information in a separate file, called the `empinfo` file, using *symbols* of the model.
    - ★ Define a model in the usual way:

      ```
      variables obj(i), x(i);
      equations defobj(i), defh(i), defg(i);
      ...
      model / defobj, defh, defg /;
      ```

    - ★ Write annotations in the `empinfo` file:

      ```
      min obj(i) s.t. x(i), defobj(i), defh(i), defg(i)
      ```

  - Identify the problem structure by parsing the `empinfo` file.
  - Verification could be performed by checking the ownership.

Representing sophisticated expressions: shared constraints

- Shared constraints

$$\min_{x_i} \quad \theta_i(x_i, x_{-i}) \quad \text{s.t.} \quad g(x_i, x_{-i}) \le 0, \quad (\perp \;\; \mu_i).$$

  - `empinfo` file:

    ```
    min obj(i) s.t. x(i), defg
    ```

  - Switching to different solution concepts is easy:

    ```
    visol defg
    min obj(i) s.t. x(i), defg
    ```

    - ★ `visol` computes a variational equilibrium, where we force use of a single g and a single μ:

$$\min_{x_i} \quad \theta_i(x_i, x_{-i}) - \mu^T g(x_i, x_{-i}),$$

$$0 \quad \ge g(x_i, x_{-i}) \quad \perp \quad \mu \le 0$$

Representing sophisticated expressions: shared variables

- Shared variables (with their defining constraints)

$$\min_{x_i, y} \quad \theta_i(x_i, y, x_{-i}),$$
$$\text{s.t.} \quad h(x_i, y, x_{-i}) = 0.$$

  - `empinfo` file:

```
implicit y, defh
min obj(i) s.t. x(i), y, defobj(i)
```

Several different uses of shared variables

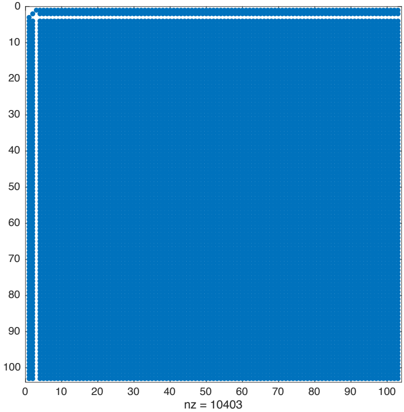
- Improve sparsity:

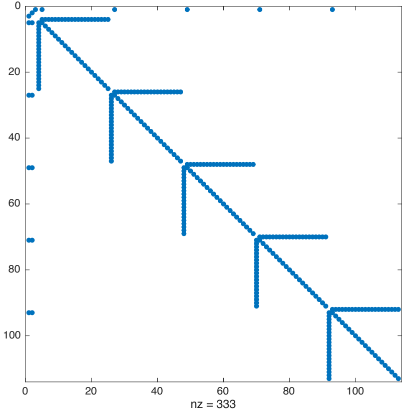
$$\max_{x_i \geq 0} \quad x_i p\left(\sum_{j=1}^{N} x_j\right) - c_i(x_i) \quad (\Rightarrow) \qquad \max_{x_i \geq 0, y = \sum_{i=1}^{N} x_i} x_i p(y) - c_i(x_i)$$

  - Computational performance could be significantly improved.

original			$y = \sum_{i=1}^{N} x_i$		
size	density	time (secs)	size	density	time (secs)
2,502	99.92%	57.78	2,508	0.20%	1.30
5,002	99.96%	420.92	5,008	0.10%	5.83
10,002	99.98%	-	10,008	0.05%	22.01
-	-	-	25,008	0.02%	148.08
-	-	-	50,008	0.01%	651.14

The sparsity patterns





Jacobian nonzero pattern $n = 100, \; N_a = 20$

Several different uses of shared variables

- Modeling mixed-behavior of agents: price-makers/price-takers

$$\max_{x_i, y=p(x)} \quad x_i y - c_i(x_i) \quad \text{or} \quad \max_{x_i} \quad x_i y - c_i(x_i).$$

  - empinfo file: only include y if agent i has control of it.

```
implicit y, defp
max obj(i) s.t.  x(i), y, defobj(i)
```

Profit	Competitive	Oligo1	Oligo12	Oligo123	Oligo1234	Oligo12345
Firm 1	123.834	125.513	145.591	167.015	185.958	199.934
Firm 2	195.314	216.446	219.632	243.593	264.469	279.716
Firm 3	257.807	278.984	306.174	309.986	331.189	346.590
Firm 4	302.863	322.512	347.477	373.457	376.697	391.279
Firm 5	327.591	344.819	366.543	388.972	408.308	410.357
Total profit	1207.410	1288.273	1385.417	1483.023	1566.621	1627.875
Social welfare	39063.824	39050.191	39034.577	39022.469	39016.373	39015.125

- Other usage: general economic conditions, shared objective variables, etc.

Optimal Value Functions

Problem type

Objective function or Constraint

$$\min_{x\in X} \theta(x) + \rho(F(x)) \qquad \min_{x\in X} \theta(x) \text{ s.t. } \rho(F(x)) \leq \alpha$$

- Special case is a Quadratic Support Function (Aravkin et al. [2013])

$$\rho(y) = \sup_{u\in U} \langle u, By + b\rangle - \frac{1}{2}\langle u, Mu\rangle$$

- Dual representation (of coherent r.m.) in terms of risk sets

$$\rho(Z) = \sup_{\mu\in\mathcal{D}} \mathbb{E}_\mu[Z]$$

- If $\mathcal{D} = \{p\}$ then $\rho(Z) = \mathbb{E}[Z]$
- If $\mathcal{D}_{\alpha,p} = \{\lambda \in [0, p/(1-\alpha)] : \langle \mathbb{1}, \lambda\rangle = 1\}$, then $\rho(Z) = \overline{CVaR}_\alpha(Z)$

The transformation to MOPEC

- EMP allows any Quadratic Support Function to be defined and facilitates model transformations to tractable forms for solution
- `empinfo` file: OVF cvarup F(x) rho .9

$$\min_{x \in X} \theta(x) + \rho(F(x))$$

$$\rho(y) = \sup_{u \in U} \left\{ \langle u, y \rangle - \frac{1}{2} \langle u, Mu \rangle \right\}$$

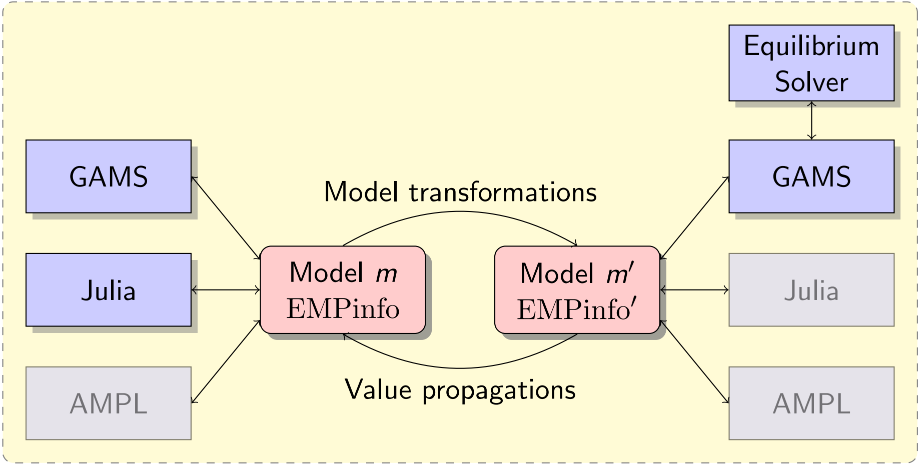
$$0 \in \partial\theta(x) + \nabla F(x)^T \partial\rho(F(x)) + N_X(x)$$

$$0 \in \partial\theta(x) + \nabla F(x)^T u + N_X(x)$$

$$0 \in -u + \partial\rho(F(x)) \iff 0 \in -F(x) + Mu + N_U(u)$$

- This is a MOPEC, and we have a copy of this construct for each agent

EMP framework



The model representation inside the EMP solver is independent of any model language

Solution methods for equilibrium problems

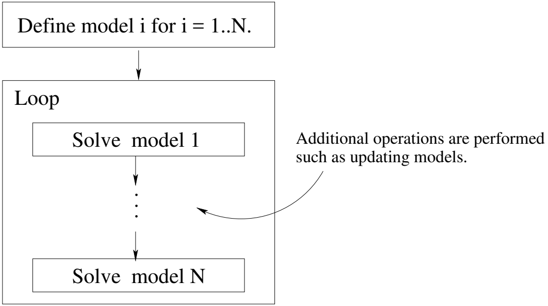
- MCP using PATH
  - Form an MCP(B, F) by concatenating the KKT conditions of agents.

$$F_i(x,\lambda,\mu) = \begin{bmatrix} \nabla_{x_i}\theta(x) - \nabla_{x_i} g(x)\lambda_i - \nabla_{x_i} h(x)\mu_i \\ g_i(x) \\ h_i(x) \end{bmatrix} \perp \begin{bmatrix} x_i \\ \lambda_i \\ \mu_i \end{bmatrix} \in B$$

  - Solve the resulting MCP using the PATH solver.
- Decomposition using (group) diagonalization
  - Repeat for $i = 1, \ldots, N$ agent i solves its problem while keeping x_{-i} fixed until convergence.
  - Jacobi: agents use the same values for other agents' variables.
  - Gauss-Seidel: each agent uses the most recent values.

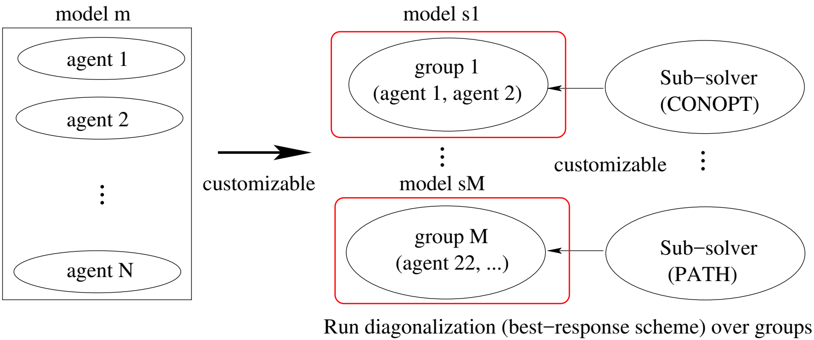
Implementing decomposition methods in modeling languages

- Typical steps (unless the underlying solver directly supports it)
  1. Define some number of submodels.
     - ex) master and subproblems in the case of Dantzig-Wolfe or Benders
  2. Define a sequence of operations to be performed over these submodels iteratively.



Selkie

- Selkie
  - Performs a model transformation: generate submodels for decomposition.
  - Supports various decomposition methods.
  - Can compute a solution in an adaptable and flexible way.
  - ex) Selkie on equilibrium problems:



An example of using Selkie for group diagonalization

- An oligopolistic market equilibrium problem:

$$\underset{q_i \geq 0}{\text{maximize}} \quad q_i p\left(\sum_{j=1, j\neq i}^{5} q_j + q_i\right) - c_i(q_i), \text{ for } i = 1, \ldots, 5.$$

Group	Iterations			
	Jacobi	GS	GSW	GS(RS)
{{1},{2},{3},{4},{5}}	155	45	28	50
{{1,2},{3,4},{5}}	57	21	22	30
{{1..3},{4,5}}	28	14	14	18
{{1..4},{5}}	22	12	12	16
{{1..5}}	1			

  - GS: Gauss-Seidel
  - GSW: Gauss-Southwell
  - GS(RS): Gauss-Seidel with random sweep

- An automatic detection of independent groups is supported.

Example: solving a dynamic economic model using value function iteration (VFI)

- A Bellman equation
  - For each state x (assuming continuous values),

$$V(x) := \max_{a \in A} \quad [c(x,a) + \beta V(x')],$$
$$\text{s.t.} \qquad x' = h(x,a).$$

  - Find a fixed point V: $V(\cdot) = T(V(\cdot))$ where T is an operator of the right-hand side of the above.
- Value function iteration
  - Create a grid of possible states: $(x_1, \ldots, x_n) \quad \Rightarrow \quad V \in \mathbb{R}^n$
  - Starting with $V^0 \in \mathbb{R}^n$, repeat
    - ★ For $i = 1, \ldots, n$,

      solve $V^k(x_i) = \max_{a_i \in A}[c(x_i, a_i) + \beta V^{k-1}(x')]$ s.t. $x' = h(x_i, a_i)$.
    - ★ Stop if $\|V^k - V^{k-1}\| \leq \epsilon$.

Solving a dynamic economic model using VFI (cont)

- We may need to evaluate $V(x)$ at x not in the grid points.
  - e.g., $x' = h(x_i, a)$ may not be in the grid points.
  - Use an approximation $\tilde{V}(x) := \sum_{j=1}^{m} \alpha_j \phi_j(x)$, where $\phi_j(\cdot)$ is a basis function.
- The problem becomes an equilibrium problem (Chang et al. [2018]):

$$
\begin{array}{lll}
\text{find} & (\alpha^*, a_1^*, \ldots, a_n^*) & \text{satisfying,} \\
\alpha^* \in & \mathrm{argmin}_{\alpha} & \displaystyle\sum_{i=1}^{n} \Big(V(x_i) - \tilde{V}(x_i) \Big)^2, \\
a_i^* \in & \mathrm{argmax}_{a_i \in A} & \Big[c(x_i, a_i) + \beta \tilde{V}(x') \Big], \\
 & \text{s.t.} & x' = h(x_i, a_i).
\end{array}
$$

Using EMP and SELKIE

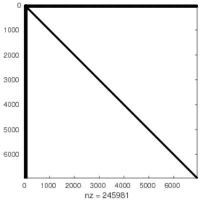
- empinfo file:

```
equilibrium
min objl s.t.  alpha(j) defobjl
max obj(i) s.t.  a(i) x(i) defobj(i) defh(i)
```

- selkie.opt file:

```
agent_group {1,{2..390}:jacobi}
parallel_jacobi yes
```

  - Interpretation
    - ★ There are two groups of agents: least squares and Bellman
    - ★ When solving Bellman, apply a parallel Jacobi method.



Performance comparisons

- Model statistics

# rows	# cols	# nnz	# grid
1,946	2,724	156,768	389

# nonlinear code	# nonlinear nonzeros
25,752,580	154,433

- Experimental results

# iter	time (mins)	
	original	SELKIE
5	9	2
10	17	3
20	32	4
⋮	⋮	⋮
583	$\gg 240$	63

Dantzig-Wolfe decomposition for VIs (Luna et al. [2012])

- Given $\mathrm{VI}(K, F)$ where $K = \{x \mid g_i(x_i) \leq 0, h(x) \leq 0\}$,
  - $h(x)$ is assumed to be a coupling constraint: $Ax = b$ in LP case.
  - Follow a similar mechanism like LP:
    - ★ Master problem $\mathrm{VI}(K_m, F_m)$

$$K_m = \left\{ \lambda \mid h\left(\sum_{i=1}^{l} \lambda_i y^i\right) \leq 0, \sum_{i=1}^{l} \lambda_i = 1, \lambda_i \geq 0 \right\},$$

$$(F_m(\lambda))_i = F\left(\sum_{i=1}^{l} \lambda_i y^i\right)^T y^i$$

    - ★ Subproblem $\mathrm{VI}(K_s, F_s)$

$$K_s = \{x \mid g_i(x_i) \leq 0\},$$
$$F_s(x) = F(x) - \nabla h(x)\mu_m$$

Using EMP and SELKIE

- `empinfo` file:

```
vi F x cons
```

- `selkie.opt`:

```
decomposition_method dantzig_wolfe
coupling_constraints nameofcons
```

- Performance comparisons:

# cols	time (secs)	
	original	SELKIE
102	1	0.2
502	6	0.9
1,002	29	4
2,502	273	26
5,002	2,040	124

Conclusions and future work

- EMP facilitates easier formulation of stochastic equilibrium problems
- The EMP framework and SELKIE automate the implementation of decomposition methods in modeling languages
  - Equilibrium problems with diagonalization, SJM
  - Variational inequalities with Dantzig-Wolfe decomposition
- They enable an efficient and flexible deployment of decomposition methods.
  - Different group decomposition
- Fast running time is achieved using efficient model generation and parallel solve of submodels.
- Need better algorithms that exploit structure
- Need to interface to more modeling languages
- Need feedback on what is difficult to do
- How to specify stochastic processes within modeling systems

References

A. Aravkin, J. V. Burke, and G. Pillonetto. Sparse / robust estimation and kalman smoothing with nonsmooth log-concave densities: Modeling, computation, and theory. *Journal of Machine Learning Research*, 14(1):2689–2728, 2013.

W. Chang, M. C. Ferris, Y. Kim, and T. F. Rutherford. Solving stochastic dynamic programming problems: a mixed complementarity approach. *Computational Economics*, 2018. Submitted.

M. C. Ferris, S. P. Dirkse, J.-H. Jagla, and A. Meeraus. An Extended Mathematical Programming Framework. *Computers and Chemical Engineering*, 33:1973–1982, 2009.

Y. Kim and M. C. Ferris. Selkie: a model transformation and distributed solver for structured equilibrium problems. *In preparation.*, 2018a.

Y. Kim and M. C. Ferris. Solving equilibrium problems using extended mathematical programming. *Mathematical Programming C*, June 2018b. Submitted.

Y. Kim, O. Huber, and M. C. Ferris. A Structure-Preserving Pivotal Method for Affine Variational Inequalities. *Mathematical Programming*, 168(1):93–121, Mar. 2017.

J. P. Luna, C. Sagastizábal, and M. Solodov. A class of dantzig–wolfe type decomposition methods for variational inequality problems. *Mathematical Programming*, 143(1-2):177–209, oct 2012.

A. B. Philpott, M. C. Ferris, and R. J. B. Wets. Equilibrium, uncertainty and risk in hydro-thermal electricity systems. *Mathematical Programming B*, 157(2):483–513, Jan. 2016.