Topic 2: Algorithms

Professor Anna Nagurney

John F. Smith Memorial Professor
and
Director – Virtual Center for Supernetworks
Isenberg School of Management
University of Massachusetts
Amherst, Massachusetts 01003

SCH-MGMT 825
Management Science Seminar
Variational Inequalities, Networks, and Game Theory
Spring 2014

Algorithms

The development of efficient algorithms for the numerical computation of equilibria is a topic as important as the qualitative analysis of equilibria.

The complexity of equilibrium problems, coupled with their increasing scale, is precluding their resolution via closed form analytics.

Also, the growing influence of policy modeling is stimulating the construction of frameworks for the accessible evaluation of alternatives.

Algorithms

Variational inequality algorithms resolve the VI problem into, typically, a series of optimization problems. Hence, usually, variational inequality algorithms proceed to the equilibrium iteratively and progressively via some procedure.

Specifically, at each iteration of a VI algorithm, one encounters a linearized or relaxed substitute of the original system, which can, typically, be rephrased or reformulated as an optimization problem and, consequently, solved using an appropriate nonlinear programming algorithm.

Algorithms

In the case where the problem exhibits an underlying structure (such as a network structure), **special-purpose algorithms may, instead, be embedded within the variational inequality algorithms to realize further efficiencies**.

Examples of VI Algorithms

The General Iterative Scheme of Dafermos, which induces such algorithms as:

- The Projection Method and
- The Relaxation Method, plus

The Modified Projection Method of Korpelevich which converges under less restrictive conditions than the general iterative scheme.

A variety of Decomposition Algorithms, both serial and parallel.

Additional Algorithms

There are also algorithms based on the general iterative scheme of Dupuis and Nagurney (1993) for the solution of projected dynamical systems (PDSs). These can also be applied to compute solutions to variational inequality problems of concern here.

Special cases of this general iterative scheme include such algorithms as the Euler method and the Heun method.

The General Iterative Scheme of Dafermos

We now present a general iterative scheme for the solution of the variational inequality problem defined in (1) (Dafermos (1983)). The iterative scheme induces, as special cases, such well-known algorithms as the projection method, linearization algorithms, and the relaxation method, and also induces new algorithms.

In particular, we seek to determine $x^* \in K \subset R^n$, such that

$$F(x^*)^T \cdot (x - x^*) \geq 0, \quad \forall x \in K, \tag{1}$$

where F is a given continuous function from K to R^n and K is a given closed, convex set. K is also assumed to be compact and $F(x)$ continuously differentiable.

The General Iterative Scheme of Dafermos

Assume that there exists a smooth function

$$g(x,y) : K \times K \mapsto R^n \tag{2}$$

with the following properties:

(i) $g(x,x) = F(x)$, for all $x \in K$,

(ii) for every fixed $x, y \in K$, the $n \times n$ matrix $\nabla_x g(x,y)$ is symmetric and positive-definite.

The General Iterative Scheme of Dafermos

Any function $g(x, y)$ with the above properties generates the following:

Algorithm

Step 0: Initialization

Start with an $x^0 \in K$. Set $t := 1$.

Step 1: Construction and Computation

Compute x^t by solving the variational inequality subproblem:

$$g(x^t, x^{t-1})^T \cdot (x - x^t) \geq 0, \quad \forall x \in K. \tag{3}$$

Step 2: Convergence Verification

If $|x^t - x^{t-1}| \leq \epsilon$, for some $\epsilon > 0$, a prespecified tolerance, then stop; otherwise, set $t := t + 1$ and go to Step 1.

The General Iterative Scheme of Dafermos

Since $\nabla_x g(x,y)$ is assumed to be symmetric and positive definite, the line integral $\int g(x,y)dx$ defines a function $f(x,y): K \times K \mapsto R$ such that, for fixed $y \in K$, $f(\cdot, y)$ is strictly convex and

$$g(x,y) = \nabla_x f(x,y). \tag{4}$$

Hence, variational inequality (3) is equivalent to the strictly convex mathematical programming problem

$$\min_{x \in K} f(x, x^{t-1}) \tag{5}$$

for which a unique solution x^t exists. The solution to (5) may be computed using any appropriate mathematical programming algorithm.

Exploiting Problem Structure

If there is, however, a special-purpose algorithm that takes advantage of the problem's structure, then such an algorithm is usually preferable from an efficiency point of view. Of course, (3) should be constructed in such a manner so that, at each iteration t, this subproblem is easy to solve.

The General Iterative Scheme of Dafermos

Note that if the sequence $\{x^t\}$ is convergent, i.e., $x^k \longrightarrow x^*$, as $t \longrightarrow \infty$, then because of the continuity of $g(x, y)$, (3) yields

$$F(x^*)^T \cdot (x - x^*) = g(x^*, x^*)^T \cdot (x - x^*) \geq 0, \quad \forall x \in K \quad (6)$$

and, consequently, x^* is a solution to (1).

A condition on $g(x, y)$, which guarantees that the sequence $\{x^k\}$ is convergent, is now given.

Theorem 1 (Convergence of General Iterative Scheme) *Assume that*

$$|||\nabla_x g^{-\frac{1}{2}}(x^1, y^1) \nabla_y g(x^2, y^2) \nabla_x g^{-\frac{1}{2}}(x^3, y^3)||| < 1 \qquad (7)$$

for all $(x^1, y^1), (x^2, y^2), (x^3, y^3) \in K$, *where* $||| \cdot |||$ *denotes the standard norm of an* $n \times n$ *matrix as a linear transformation on* R^n. *Then the sequence* $\{x^k\}$ *is Cauchy in* R^n.

The General Iterative Scheme of Dafermos

A necessary condition for (7) to hold is that $F(x)$ is strictly monotone.

Hence, the general iterative scheme was shown to converge by establishing contraction estimates that allow for the possibility of adjusting the norm at each iteration of the algorithm. This flexibility will, in general, yield convergence, under weaker assumptions.

The Projection Method

The projection method resolves variational inequality (1) into a sequence of subproblems (3) (cf. also (5)) which are equivalent to quadratic programming problems. Quadratic programming problems are usually easier to solve than more highly nonlinear optimization problems, and effective algorithms have been developed for such problems.

The Projection Method

In the framework of the general iterative scheme, the projection method corresponds to the choice

$$g(x, y) = F(y) + \frac{1}{\rho} G(x - y), \quad \rho > 0 \tag{8}$$

where G is a fixed symmetric positive definite matrix. At each step k of the projection method, the subproblem that must be solved is given by:

$$\min_{x \in K} \frac{1}{2} x^T \cdot Gx + (\rho F(x^{t-1}) - Gx^{t-1})^T \cdot x. \tag{9}$$

The Projection Method

In particular, if G is selected to be a diagonal matrix, then (9) is a separable quadratic programming problem.
Condition (7) for convergence of the projection method takes the form:

Theorem 3 (Convergence)
Assume that

$$|||I - \rho G^{-\frac{1}{2}} \nabla_x F(x) G^{-\frac{1}{2}}||| < 1, \quad \forall x \in K \tag{10}$$

where $\rho > 0$ and fixed. Then the sequence generated by the projection method (9) converges to the solution of variational inequality (1).

The Relaxation Method

The relaxation (sometimes also called diagonalization) method resolves variational inequality (1) into a sequence of subproblems (3) which are, in general, nonlinear programming problems.

In the framework of the general iterative scheme, the relaxation method corresponds to the choice

$$g_i(x, y) = F_i(y_1, \ldots, y_{i-1}, x_i, y_{i+1} \ldots, y_n), \quad i = 1, \ldots, n. \tag{11}$$

The Relaxation Method

The assumptions under which the relaxation method converges are now stated.

Theorem 4

Assume that there exists a $\gamma > 0$ such that

$$\frac{\partial F_i(x)}{\partial x_i} \geq \gamma, \quad i = 1, \ldots, n, \quad x \in K \tag{12}$$

and

$$|||\nabla_y g(x, y)||| \leq \lambda\gamma, \quad 0 < \lambda < 1, \quad x, y \in K \tag{13}$$

then condition (7) of Theorem 1 is satisfied.

The Modified Projection Method

Note that a necessary condition for convergence of the general iterative scheme is that $F(x)$ is strictly monotone. In the case that such a condition is not met by the application under consideration, a modified projection method may still be appropriate.

This algorithm requires, instead, only monotonicity of F, but with the Lipschitz continuity condition holding, with constant L. The G matrix (cf. the projection method) is now the identity matrix I.

The algorithm is now stated.

The Modified Projection Method

Step 0: Initialization

Start with an $x^0 \in K$. Set $t := 1$ and select ρ, such that $0 < \rho \leq \frac{1}{L}$, where L is the Lipschitz constant for function F in the variational inequality problem.

Step 1: Construction and Computation

Compute $\bar{x}^{t-1}$ by solving the variational inequality subproblem:

$$\left[\bar{x}^{t-1} + (\rho F(x^{t-1}) - x^{t-1})\right]^T \cdot \left[x' - \bar{x}^{t-1}\right] \geq 0, \quad \forall x' \in K. \tag{14}$$

The Modified Projection Method

Step 2: Adaptation

Compute x^t by solving the variational inequality subproblem:

$$\left[x^t + (\rho F(\bar{x}^{t-1}) - x^{t-1})\right]^T \cdot [x' - x^t] \geq 0, \quad \forall x' \in K. \quad (15)$$

Step 3: Convergence Verification

If $|x^t - x^{t-1}| \leq \epsilon$, for $\epsilon > 0$, a prespecified tolerance, then, stop; otherwise, set $t := t + 1$ and go to Step 1.

The Modified Projection Method

The modified projection method converges to the solution of $\mathrm{VI}(F, K)$, where K is assumed to be nonempty, but not necessarily compact, under the following conditions.

Theorem 5 (Convergence)
Assume that $F(x)$ is monotone, that is,

$$(F(x^1) - F(x^2))^T \cdot (x^1 - x^2) \geq 0, \quad \forall x^1, x^2 \in K,$$

and that $F(x)$ is also Lipschitz continuous, that is, there exists a constant $L > 0$ such that

$$\|F(x^1) - F(x^2)\| \leq L\|x^1 - x^2\|, \quad \forall x^1, x^2 \in K.$$

Then the modified projection method converges to a solution of variational inequality (1).

Decomposition Algorithms

Now it is assumed that the feasible set K is a Cartesian product, that is,

$$K = \prod_{i=1}^{m} K_i$$

where each $K_i \subset R^{n_i}$, $\sum_{i=1}^{m} n_i = n$, and x_i now denotes, without loss of generality, a vector $x_i \in R^{n_i}$, and $F_i(x) : K \mapsto R^{n_i}$ for each i.

Many equilibrium problems are defined over a Cartesian product set and, hence, are amenable to solution via variational inequality decomposition algorithms.

Decomposition Algorithms

The appeal of decomposition algorithms lies in their particular suitability for the solution of large-scale problems. Furthermore, parallel decomposition algorithms can be implemented on parallel computer architectures and further efficiencies realized.

Decomposition Algorithms

For example, in the case of multicommodity problems, in which there are m commodities being produced, traded, and consumed, a subset K_i might correspond to constraints for commodity i. On the other hand, in the case of intertemporal problems, K_i might correspond to the constraints governing a particular time period i.

Decomposition Algorithms

Moreover, a given equilibrium problem may possess alternative variational inequality formulations over distinct Cartesian products; each such formulation, in turn, may suggest a distinct decomposition procedure.

Numerical testing of the algorithms, on the appropriate architecture(s), subsequent to the theoretical analysis, can yield further insights into which algorithm(s) performs in a superior (or satisfactory) manner, as mandated by the particular application.

Decomposition Algorithms

An important observation for the Cartesian product case is that the variational inequality now decomposes into m coupled variational inequalities of smaller dimensions, which is formally stated as:

Proposition 1

A vector $x^* \in K$ solves variational inequality (1) where K is a Cartesian product if and only if

$$F_i(x^*)^T \cdot (x_i - x_i^*) \geq 0, \quad \forall x_i \in K_i, \quad \forall i.$$

Decomposition Algorithms

The linearized variational inequality decomposition algorithms are now presented, both the serial version, and then the parallel version.

The former is a Gauss-Seidel method in that it serially updates the information as it becomes available.

The latter is a Jacobi method in that the updating is done simultaneously, and, hence, can be done in parallel. For both linearized methods, the variational inequality subproblems are linear.

Linearized Decomposition Algorithm - Serial Version

Step 0: Initialization

Start with an $x^0 \in K$. Set $t := 1$; $i := 1$.

Step 1: Linearization and Computation

Compute the solution $x_i^t = x_i$ to the variational inequality subproblem:

$$\Big[F_i(x_1^t, \ldots, x_{i-1}^t, x_i^{t-1}, \ldots, x_m^{t-1})$$

$$+A_i(x_1^t, \ldots, x_{i-1}^t, x_i^{t-1}, \ldots, x_m^{t-1}) \cdot (x_i - x_i^{t-1})\Big]^T$$

$$\cdot [x_i' - x_i] \geq 0, \; \forall x_i' \in K_i.$$

Set $i := i + 1$. If $i \leq m$, go to Step 1; otherwise, go to Step 2.

Linearized Decomposition Algorithm - Serial Version

Step 2: Convergence Verification

If $|x^t - x^{t-1}| \leq \epsilon$, for $\epsilon > 0$, a prespecified tolerance, then stop; otherwise, set $t := t + 1$; $i = 1$, and go to Step 1.

Linearized Decomposition Algorithm - Parallel Version

Step 0: Initialization

Start with an $x^0 \in K$. Set $t := 1$.

Step 1: Linearization and Computation

Compute the solutions $x_i^t = x_i$; $i = 1, \ldots, m$, to the m variational inequality subproblems:

$$\left[F_i(x^{t-1}) + A_i(x^{t-1}) \cdot (x_i - x_i^{t-1})\right]^T \cdot [x_i' - x_i] \geq 0,$$

$$\forall x_i' \in K_i, \; \forall i.$$

Step 2: Convergence Verification

If $|x^t - x^{t-1}| \leq \epsilon$, for $\epsilon > 0$, a prespecified tolerance, then stop; otherwise, set $t := t + 1$, and go to Step 1.

Linearized Decomposition Algorithm - Parallel Version

Possible choices for $A_i(\cdot)$ are as follows.

If $A_i(x^{t-1}) = \nabla_{x_i} F_i(x^{t-1})$, then a Newton's method is obtained.

If $A_i(x^{t-1}) = D_i(x^{t-1})$, where $D_i(\cdot)$ denotes the diagonal part of $\nabla_{x_i} F_i(\cdot)$, then a linearization method is induced.

If $A_i(\cdot) = G_i$, where G_i is a fixed, symmetric and positive-definite matrix, then a projection method is obtained.

Linearized Decomposition Algorithm - Parallel Version

Note that the variational inequality subproblems should be easier to solve than the original variational inequality since they are smaller variational inequality problems, defined over smaller feasible sets.

In particular, if each $A_i(\cdot)$ is selected to be diagonal and positive definite, then each of the subproblems is equivalent to a separable quadratic programming problem with a unique solution.

Convergence Results

Theorem 6 (Convergence of Linearized Decomposition Schemes)

Suppose that the variational inequality problem (1) has a solution x^* and that there exist symmetric positive definite matrices G_i and some $\delta > 0$ such that $A_i(x) - \delta G_i$ is positive semidefinite for every i and $x \in K$, and that there exists a $\gamma \in [0, 1)$ such that

$$\|G_i^{-1}(F_i(x) - F_i(y) - A_i(y) \cdot (x_i - y_i))\|_i \leq \delta\gamma \max_j \|x_j - y_j\|_j,$$

$$\forall x, y \in K,$$

where $\|x_i\|_i = (x_i^T G_i x_i)^{\frac{1}{2}}$. Then both the parallel and the serial linearized decomposition algorithms with $A_i(x)$ being diagonal and positive definite, converge to the solution x^* geometrically.

The Nonlinear Decompisition Algorithms

The nonlinear analogues of the above Linearized Decomposition Algorithms are now presented.

Nonlinear Decomposition Algorithm - Serial Version

Step 0: Initialization

Start with an $x^0 \in K$. Set $t := 1$; $i := 1$.

Step 1: Relaxation and Computation

Compute the solution $x_i^t = x_i$ by solving the variational inequality subproblem:

$$F_i(x_1^t, \ldots, x_{i-1}^t, x_i, x_{i+1}^{t-1}, \ldots, x_m^{t-1})^T \cdot [x_i' - x_i] \geq 0, \quad \forall x_i' \in K_i.$$

Set $i := i + 1$. If $i \leq m$, go to Step 1; otherwise, go to Step 2.

Step 2: Convergence Verification

If $|x^t - x^{t-1}| \leq \epsilon$, for $\epsilon > 0$, a prespecified tolerance, then stop; otherwise, set $t := t + 1$; $i = 1$, and go to Step 1.

The Nonlinear Decompisition Algorithms

The parallel analogue is now given.

Nonlinear Decomposition Algorithm - Parallel Version

Step 0: Initialization

Start with an $x^0 \in K$. Set $t := 1$.

Step 1: Relaxation and Computation

Compute the solutions $x_i^t = x_i$; $i = 1, \ldots, m$, to the variational inequality subproblems:

$$F_i(x_1^{t-1}, \ldots, x_{i-1}^{t-1}, x_i, x_{i+1}^{t-1}, \ldots, x_m^{t-1})^T \cdot [x_i' - x_i] \geq 0,$$

$$\forall x_i' \in K_i, \forall i.$$

Step 2: Convergence Verification

If $|x^t - x^{t-1}| \leq \epsilon$, for $\epsilon > 0$, a prespecified tolerance, then stop; otherwise, set $t := t + 1$, and go to Step 1.

Convergence Results

Theorem 7 (Convergence of Nonlinear Decomposition Schemes)
Suppose that the variational inequality problem (1) has a solution x^* *and that there exist symmetric positive-definite matrices* G_i *and some* $\delta > 0$ *such that* $A_i(x) - \delta G_i$ *is positive semidefinite for every* i *and* $x \in K$*, and that there exists a* $\gamma \in [0, 1)$ *such that*

$$\|G_i^{-1}(F_i(x) - F_i(y) - A_i(y) \cdot (x_i - y_i)\|_i \leq \delta\gamma \max_j \|x_j - y_j\|_j,$$

$$\forall x, y \in K,$$

where $\|x_i\|_i = (x_i^T G_i x_i)^{\frac{1}{2}}$. *Then both the parallel and the serial nonlinear decomposition algorithms converge to the solution* x^* *geometrically.*

Equilibration Algorithms

Recall that variational inequality algorithms proceed to the equilibrium iteratively and progessively via some "equilibration" procedure, which involves the solution of a linearized or relaxed substitute of the system at each step.

If the equilibration problem encountered at each step is an optimization problem (which is usually the case), then, in principle, any appropriate optimization algorithm may be used for the solution of such embedded problems.

Equilibration Algorithms

However, since the overall efficiency of a variational inequality algorithm will depend upon the efficiency of the procedure used at each step, an algorithm that exploits problem structure, if such a structure is revealed, is usually preferable if efficiency is mandated by the application.

Equilibration Algorithms

Since many equilibrium problems of interest have a network structure, we now describe equilibration algorithms that exploit network structure.

Equilibration algorithms were introduced by Dafermos and Sparrow (1969) for the solution of traffic assignment problems, both user-optimized and system-optimized problems, on a general network.

Congested Transportation Networks

Equilibration Algorithms

In a **user-optimized problem**, each user of a network system seeks to determine his/her cost-minimizing route of travel between an origin/destination pair, until an equilibrium is reached, in which no user can decrease his/her cost of travel by unilateral action.

In a **system-optimized network problem**, users are allocated among the routes so as to minimize the total cost in the system. Both classes of problems, under certain imposed assumptions, possess optimization formulations.

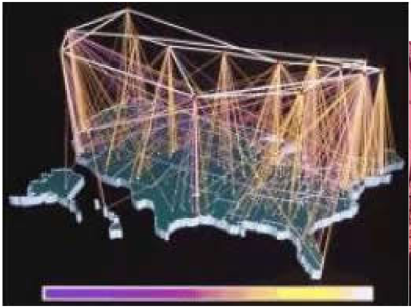
Behavior on Congested Networks

Flows are routed on individual cost-minimizing routes.

Decentralized		Selfish		User-Optimized or Equilibrium U–O
vs.	⇔	vs.	⇔	vs.
Centralized		Unselfish		S–O System-Optimized

Flows are routed so as to minimize the total cost.

Other Networks that Behave like Traffic Networks



The Internet, electric power networks, supply chains, and even multitiered financial networks!

Equilibration Algorithms

In particular, the user-optimized, or equilibrium problem was shown to be characterized by equilibrium conditions which, under certain symmetry assumptions on the user cost functions, were equivalent to the Kuhn-Tucker conditions of an optimization problem (albeit artificial).

Equilibration Algorithms

The first equilibration algorithms assumed that the demand associated with an origin/destination (O/D) pair was known and fixed. In addition, for networks of special structure, specifically, those with linear user cost functions and paths connecting an O/D pair having no links in common, a special-purpose algorithm could be used to compute an O/D pair's equilibrium path flows and associated link flows exactly and in closed form. This approach is sometimes referred to as "exact equilibration."

Equilibration Algorithms

Later, the algorithms were generalized to the case where the demands are unknown and have to be computed as well.

General Equilibration Algorithms

Equilibration algorithms were devised for the computation of user and system-optimized flows on general networks. They are, in principle, “relaxation” methods in that they resolve the solution of a nonlinear network flow problem into a series of network problems defined over a smaller and, hence, a simpler feasible set. Equilibration algorithms typically proceed from origin/destination (O/D) pair to O/D pair, until the entire system is solved or “equilibrated.”

General Equilibration Algorithms

We now present equilibration algorithms for the solution of network equilibrium problems with separable and linear "user" cost functions on the links. We begin with the equilibration algorithm for the single O/D pair problem with fixed demand, and then generalize it to J O/D pairs.

Classical Network Equilibrium Problem

Consider a general network $G = [N, L]$, where N denotes the set of nodes, and L the set of directed links. Let a denote a link of the network connecting a pair of nodes, and let p denote a path consisting of a sequence of links connecting an O/D pair. P_w denotes the set of paths connecting the O/D pair of nodes w.

Let x_p represent the flow on path p and f_a the load on link a. The following conservation of flow equation must hold:

$$f_a = \sum_{p \in P} x_p \delta_{ap},$$

where $\delta_{ap} = 1$, if link a is contained in path p, and 0, otherwise. This expression states that the load on a link a is equal to the sum of all the path flows on paths p that contain (traverse) link a.

Classical Network Equilibrium Problem

Moreover, if we let d_w denote the demand associated with O/D pair w, then we must have that

$$d_w = \sum_{p \in P_w} x_p,$$

where $x_p \geq 0$, $\forall p$, that is, the sum of all the path flows between an origin/destination pair w must be equal to the given demand d_w.

Let c_a denote the user cost associated with traversing link a, and C_p the user cost associated with traversing the path p. Then

$$C_p = \sum_{a \in L} c_a \delta_{ap}.$$

In other words, the cost of a path is equal to the sum of the costs on the links comprising the path.

Classical Network Equilibrium Problem

The network equilibrium conditions are then given by: For each path $p \in P_w$ and every O/D pair w:

$$C_p \begin{cases} = \lambda_w, & \text{if} \quad x_p^* > 0 \\ \geq \lambda_w, & \text{if} \quad x_p^* = 0 \end{cases}$$

where λ_w is an indicator, whose value is not known a priori. These equilibrium conditions state that the user costs on all used paths connecting a given O/D pair will be minimal and equalized.

The equilibration algorithms for general networks and fixed demands first identify the most expensive used path for an O/D pair, and then the cheapest path, and equilibrate the costs for these two paths, by reassigning a portion of the flow from the most expensive path to the cheapest path. This process continues until the equilibrium is reached to a prespecified tolerance.

In the case of linear user cost functions, that is, where the user cost on link a is given by

$$c_a(f_a) = g_a f_a + h_a,$$

with $g_a > 0$ and $h_a > 0$, this reassignment or reallocation process can be computed in closed form.

Assume, for the time being, that there is only a single O/D pair w_i on a given network. An equilibration algorithm is now presented for the computation of the equilibrium path and link flows satisfying the conditions, where the feasibility conditions (conservation of flow equations) are also satisfied by the equilibrium pattern. Cost functions of the simple, separable, and linear form above are considered.

Single O/D Pair U-O Equilibration

Step 0: Initialization

Construct an initial feasible flow pattern x^0, which induces a feasible link flow pattern. Set $k := 1$.

Step 1: Selection and Convergence Verification

Determine

$$r = \{p | \max_p C_p \quad \text{and} \quad x_p^{k-1} > 0\}$$

$$q = \{p | \min_p C_p\}.$$

If $|C_r - C_q| \leq \epsilon$, with $\epsilon > 0$, a prespecified tolerance, then stop; otherwise, go to Step 2.

Single O/D Pair U-O Equilibration

Step 2: Computation

Compute

$$\Delta' = \frac{[C_r - C_q]}{\sum_{a \in L} g_a(\delta_{aq} - \delta_{ar})^2}$$

$$\Delta = \min\{\Delta', x_r^{k-1}\}.$$

Set

$$x_r^k = x_r^{k-1} - \Delta$$

$$x_q^k = x_q^{k-1} + \Delta$$

$$x_p^k = x_p^{k-1}, \quad \forall p \neq q \cup r.$$

Let $k := k + 1$, and go to Step 1.

In the case that a tie exists for the selection of path r and/or q, then any such selection is appropriate.

Convergence

Convergence of this procedure is established by constructing an associated optimization problem, the Kuhn-Tucker conditions of which are equivalent to the equilibrium conditions. This problem is given by:

$$\text{Minimize} \quad \sum_{a \in L} \frac{1}{2} g_a f_a^2 + h_a f_a$$

subject to conservation of flow equations and the nonnegativity assumption on the path flows.

One then demonstrates that a reallocation of the path flows as described above decreases the value of the appropriate function until optimality, equivalently, equilibrium conditions are satisfied, within a prespecified tolerance.

Multiple O/D Pairs

On a network in which there are now J O/D pairs, the above single O/D pair equilibration procedure is applicable as well.

We term Step 1 above (without the convergence check) and Step 2 of the above as the equilibration operator E_{w_i} for a fixed O/D pair w_i. Now two possibilities for equilibration present themselves.

Equilibration I

Let $E^1 \equiv E_{w_J} \circ \ldots \circ E_{w_1}$.

Step 0: Initialization

Construct an initial feasible flow pattern which induces a feasible link flow pattern. Set $k := 1$.

Step 1: Equilibration

Apply E^1.

Multiple O/D Pair U-O Equilibration

Step 2: Convergence Verification

If convergence holds, stop; otherwise, set $k := k + 1$, and go to Step 1.

Equilibration II

Let $E^2 = (E_{w_J} \circ (\ldots \circ (E_{w_J}))) \circ \ldots \circ (E_{w_1} \circ (\ldots \circ (E_{w_1})))$.

Step 0: Initialization (as above).

Step 1: Equilibration

Apply E^2.

Step 2: Convergence Verification (as above).

The distinction between E^1 and E^2 is as follows. E^1 equilibrates only one pair of paths for an O/D pair before proceeding to the next O/D pair, and so on, whereas E^2 equilibrates the costs on all the paths connecting an O/D pair using the 2-path procedure above, before proceeding to the next O/D pair, and so on.

The Elastic Demand Version

The elastic demand situation, where the demand d_w is no longer known a priori but needs to be computed as well, is now briefly described. For the elastic demand model assume as given a disutility function $\lambda_w(d_w)$, for each O/D pair w, that is monotonically decreasing. One may then transform the elastic model into one with fixed demands as follows. For each O/D pair w we determine an upper bound on the demand $\bar{d}_w$ and construct an overflow arc w connecting the O/D pair w. The user cost on such an arc is $c_w \equiv \lambda_w(\bar{d}_w - f_{a_w})$, where f_w denotes the flow on arc a_w. The fixed demand for O/D pair w then is set equal to $\bar{d}_w$.

Modeling Extensions

Although there exist algorithms to compute the equilibrium in the elastic demand problem, the elastic demand problem can be transformed into a **fixed demand** problem, for which efficient algorithm exists. Recall the equilibration algorithms that we studied earlier.

Reformulation of Elastic Demand Problems as Fixed Demand Problems

The Excess Overflow Reformulation

For each O/D pair $w \in W$, we first construct a path connecting O/D pair w consisting of a single link which we denote by w. Given an upper bound $\bar{d}_w > d_w(\lambda_w)$, for every w, we then construct associated cost functions on the excess overflow links thus:

$$c_w = \lambda_w(\bar{d}_w - f_w).$$

An Example

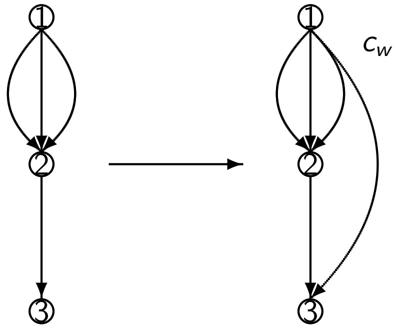


Figure: Fixed demand reformulation of elastic demand problem

If $\lambda_w = -d_w + 100$, then $\bar{d}_w = 100$, and
$c_w = \lambda(\bar{d}_w - f_w) = -(100 - f_w) + 100 = f_w$.

The System-Optimized Problem

The above discussion focused on the user-optimized problem. We now turn to the system-optimized problem in which a central controller, say, seeks to minimize the total cost in the network system, where the total cost is expressed as

$$\sum_{a \in L} \hat{c}_a(f_a)$$

where it is assumed that the total cost function on a link a is defined as:

$$\hat{c}_a(f_a) \equiv c_a(f_a) \times f_a,$$

subject to the conservation of flow equations, and the nonnegativity assumption on the path flows. Here separable link costs have been assumed, for simplicity, and other total cost expressions may be used, as mandated by the particular application.

The System-Optimized Problem

Under the assumption of strictly increasing user link cost functions, the optimality conditions are: For each path $p \in P_w$, and every O/D pair w:

$$\hat{C}'_p \begin{cases} = \mu_w, & \text{if} \quad x_p > 0 \\ \geq \mu_w, & \text{if} \quad x_p = 0, \end{cases}$$

where $\hat{C}'_p$ denotes the marginal cost on path p, given by:

$$\hat{C}'_p = \sum_{a \in L} \frac{\partial \hat{c}_a(f_a)}{\partial f_a} \delta_{ap}.$$

The System-Optimized Problem

Under the assumption of linear user cost functions as above, one may adapt the Equilibration Algorithm above to yield the solution to the system-optimized problem. Indeed, in the case of a single O/D pair, the restatement would be:

Single O/D Pair Optimization

Step 0: Initialization

Construct an initial feasible flow pattern x^0, which induces a feasible link load pattern. Set $k := 1$.

Step 1: Selection and Convergence Verification

Determine

$$r = \{p | \max_p \hat{C}'_p \quad \text{and} \quad x_p^{k-1} > 0\}.$$

$$q = \{p | \min_p \hat{C}'_p\}.$$

The System-Optimized Problem

If $|\hat{C}'_r - \hat{C}'_q| \leq \epsilon$, with $\epsilon > 0$, a prespecified tolerance, then stop; otherwise, go to Step 2.

Step 2: Computation

Compute

$$\Delta' = \frac{\left[\hat{C}'_r - \hat{C}'_q\right]}{\sum_{a \in L} 2g_a(\delta_{aq} - \delta_{ar})}$$

$$\Delta = \min\{\Delta', x_r^{k-1}\}.$$

Set

$$x_r^k = x_r^{k-1} - \Delta$$

$$x_q^k = x_q^{k-1} + \Delta$$

$$x_p^k = x_p^{k-1}, \forall p \neq q \cup r.$$

Let $k := k + 1$, and go to Step 1.

The Equilibration Schemes E^1 and E^2 can then be adapted accordingly. One should note that the system-optimized solution corresponds to the user-optimized solution on a congested network, i.e., one with increasing user link cost functions, only in highly stylized networks.

Nevertheless, one does have access to policy interventions in the form of tolls, which will make the system-optimized flows pattern, a user-optimized one. We will discuss this policy in the framework of transportation network equilibrium problems in the next lecture.

Summary

We have overviewed some of the fundamental algorithms for the computation of solutions to variational inequality problems.

We have emphasized the importance of selecting an algorithm for computational efficiency and the need to exploit problem structure.

In this seminar, we will be discussing many different network problems and will show how a spectrum of these computational schemes work in practice.

References

Below are references cited in the lecture as well as additional ones.

Ahuja, R. K., Magnanti, T. L., and Orlin, J. B., **Network Flows: Theory, Algorithms, and Applications**, Prentice-Hall, Upper Saddle River, New Jersey, 1993.

Avriel, M., **Nonlinear Programming: Analysis and Methods**, Prentice - Hall, Englewood Cliffs, New Jersey, 1976.

Beckmann, M., McGuire, C. B., and Winsten, C. B., **Studies in the Economics of Transportation**, Yale University Press, New Haven, Connecticut, 1956.

References

Bertsekas, D. P., and Tsitsiklis, J. N., **Parallel and Distributed Computation - Numerical Methods**, Prentice - Hall, Englewood Cliffs, New Jersey, 1989.

Dafermos, S., "An extended traffic assignment model with applications to two-way traffic," *Transportation Science* **5** (1971) 366-389.

Dafermos, S., "An iterative scheme for variational inequalities," *Mathematical Programming* **26** (1983) 40-47.

Dafermos, S., and Nagurney, A., "Supply and demand equilibration algorithms for a class of market equilibrium problems," **23** (1989) 118-124.

Dafermos, S. C., and Sparrow, F. T., "The traffic assignment problem for a general network," *Journal of Research of the National Bureau of Standards* **73B** (1969) 91-118.

References

Dupuis, P., and Nagurney, A., “Dynamical systems and variational inequalities,” *Annals of Operations Research* **44** (1993) 9-42.

Eydeland, A., and Nagurney, A., “Progressive equilibration algorithms: the case of linear transaction costs,” *Computer Science in Economics and Management* **2** (1989) 197-219.

Gartner, N. H., “Optimal traffic assignment with elastic demands: a review; part II: Algorithmic approaches,” *Transportation Science* **14** (1980) 192-208.

Hearn, D. W., Lawphongpanich, S., and Ventura, J. A., “Restricted simplicial decomposition: computation and extensions,” *Mathematical Programming Study* **31** (1987) 99-118.

References

Korpelevich, G. M., "The extragradient method for finding saddle points and other problems," *Matekon* **13** (1977) 35-49.

Leventhal, T., Nemhauser, G., and Trotter, L., Jr., "A column generation algorithm for optimal traffic assignment," *Transportation Science* **7** (1973) 168-176.

Nagurney, A., "An equilibration scheme for the traffic assignment problem with elastic demands," *Transportation Research* **22B** (1988) 73-79.

Nagurney, A., editor, **Advances in Equilibrium Modeling, Analysis, and Computation**, *Annals of Operations Research* **44**, J. C. Baltzer AG Scientific Publishing Company, Basel, Switzerland, 1993.

Nagurney, A., and Zhang, D., **Projected Dynamical Systems and Variational Inequalities with Applications**, Kluwer Academic Publishers, Boston, Massachusetts, 1996.

References

Pang, J. S., and Chan, D., "Iterative methods for variational and complementarity problems," *Mathematical Programming* **24** (1982) 284-313.

Patriksson, M., **The Traffic Assignment Problem**, VSP, Utrecht, The Netherlands, 1994.

Sheffi, Y., **Urban Transportation Networks - Equilibrium Analysis with Mathematical Programming Methods**, Prentice-Hall, Englewood Cliffs, New Jersey, 1985.

Zangwill, W. I., **Nonlinear Programming: A Unified Approach**, Prentice - Hall, Englewood Cliffs, New Jersey, 1969.